

# SRR: An $O(1)$ Time Complexity Packet Scheduler for Flows in Multi-Service Packet Networks

Chuanxiong Guo, *Member, IEEE*

## *Abstract*

In this paper, we present a novel fair queueing scheme, which we call Smoothed Round Robin (SRR). Ordinary round robin schedulers are well known for the burstiness of their scheduling output. In order to overcome this problem, SRR codes the weights of the flows into binary vectors to form a Weight Matrix, and then uses a Weight Spread Sequence (WSS), which is specially designed to distribute the output more evenly, to schedule packets by scanning a Weight Matrix. By using the WSS and the Weight Matrix, SRR emulates the Generalized Processor Sharing (GPS) well. SRR possesses better short-term fairness and scheduling delay properties in comparison with various existing round robin schedulers. At the same time, SRR preserves  $O(1)$  time complexity by avoiding the time-stamp maintenance employed in various fair queueing schedulers. Simulation and implementation experiments show that SRR provides good mean end-to-end delay for soft real-time services. SRR can be implemented in high-speed networks to provide quality of service (QoS) due to its simplicity and low time complexity.

## *Index Terms*

Packet scheduling, fair queueing, time complexity, end-to-end delay, high-speed networks.

## I. INTRODUCTION

With the expansion of the Internet, more and more services besides the traditional best effort services are added into the network. Video and audio conferencing, remote medical caring are some examples. It is expected that more services to be introduced in the near future. Different types of services have different characteristics, and generally have different requirements. For example, video conferencing is a kind of VBR service that requires broad bandwidth and low end-to-end delay bound, while traditional data services do not have explicit quality of service (QoS) requirements. Even with the rapid increasing rate of the transmission medium, certain kind of isolation is needed to satisfy the QoS requirements of the

---

Manuscript received May 25, 2002; revised November 27, 2002. This work was supported by the 863 Program of China under contract number 863-300-02-04-99. This paper was presented in part at the ACM SIGCOMM 2001, San Diego, CA, August 2001.

The author was with the Institute of Communications Engineering, Nanjing 210016 China. He is now with the Wireless and Networking Group of Microsoft Research Asia, Beijing 100080 China (email: xguo@ieee.org).

competing *flows* (as defined in [32], here a flow is a stream of packets that traverse the same route from the source to the destination, and that require the same grade of transmission service. Flows can be further aggregated into *classes*). Many mechanisms on how to provide QoS support for packet networks have been proposed [3], [7], [12], [19], [32], [33]. One of the most important parts of these mechanisms is a packet scheduler. A packet scheduler's task is to decide which packet to be transmitted when the output link is idle. Traditional routers use First Come First Serve (FCFS) to schedule packets. FCFS does not distinguish different flows. Hence, it does not provide any kind of isolation among different flows. We only consider schedulers that distinguish different flows or classes in this paper.

Generally, a packet scheduler is expected to have the following properties:

1. Has low time complexity to select and forward a packet;
2. Treats different flows fairly;
3. Provides low worst case delay and delay variation;
4. Is simple enough to implement efficiently.

The simplicity and time complexity properties always collide with the fairness and delay bound properties. Schedulers with short-term fairness and strict delay bound generally have high time complexity and are hard to be implemented.  $O(1)$  time complexity schemes are easy to be implemented, but they generally fail to provide short-term fairness and low scheduling delay bound.

In time-stamp based schedulers (one of the two types of well studied work-conserving scheduling algorithms), a virtual time clock is maintained to emulate the ideal Generalized Processor Sharing (GPS [19]). Traditional Weighted Fair Queueing (WFQ) [10] (PGPS [19]) has low local delay bound and good fairness, but its time complexity is  $O(N)$  ( $N$  is the number of the active flows). Variants of WFQ such as Virtual-Clock [32], WF<sup>2</sup>Q [1], Start-time FQ [15], FFQ, SPFQ [25], Time-shift FQ [8] use different methods to calculate the time-stamp, but still have at least  $O(\log N)$  time complexity. Since the best known algorithm to insert a number into a sorted array needs  $O(\log N)$ , it is unlikely that a time-stamp based scheduler with  $O(1)$  time complexity can be found. However, an  $O(\log N)$  scheduler is not good enough for a high-speed link. For instance, it takes approximate 0.08us to transmit a 100 bytes packet for a 10Gbps link. That means an  $O(\log N)$  scheduler must finish the packet selection in 0.08us regardless of the number of flows. The situation becomes even worse when the capacity of the output link is 40Gbps or higher.

On the other hand, the other type of work-conserving schedulers, the round robin schemes are simple to implement and have  $O(1)$  time complexity, but they are well known for their output burstiness and short-term unfairness. Deficit Round Robin (DRR) [23] and Carry-Over Round Robin (CORR) [22] are typical round robin schedulers. In these schemes, the schedulers serve a flow for a continuous period of time in proportion to the weight of the flow, resulting in a highly bursty scheduling output for each flow. Therefore, these schemes are considered not suitable to provide QoS in packet networks.

In this paper, we present a Smoothed Round Robin (SRR) scheduler to overcome the shortcomings of the ordinary round robin schedulers. SRR has short-term fairness and certain scheduling delay bound, as well as  $O(1)$  time complexity. A Weight Spread Sequence (WSS) and a Weight Matrix are introduced as two key data structures of the scheduler. The weights of the

flows are coded into binary vectors to form a Weight Matrix, and then SRR uses the corresponding WSS to scan the Weight Matrix. WSS is a specially designed sequence that distributes the output traffic of each flow evenly. Thus, SRR emulates GPS as the various time-stamp based schedulers do. Since it does not need to maintain any tags or states, SRR achieves  $O(1)$  time complexity, short-term fairness, and certain delay bound simultaneously.

In the following, an example is used to illustrate how SRR works. The definitions of WSS and Weight Matrix are given in Section II.

Assume that there are four flows with fixed packet size, named  $f_1, f_2, f_3, f_4$ , with rates  $r_1 = 64kbps$ ,  $r_2 = 256kbps$ ,  $r_3 = 512kbps$ ,  $r_4 = 192kbps$ , respectively. All the packet sizes of the flows are 512 bytes and all the four flows are assumed to be backlogged. The bandwidth of the output link  $C$  is  $1Mbps$ . The corresponding weights of the flows are  $w_1 = 1$ ,  $w_2 = 4$ ,  $w_3 = 8$ ,  $w_4 = 3$ . By coding the weights into binary vectors, we have  $WV_1 = \{0,0,0,1\}$ ,  $WV_2 = \{0,1,0,0\}$ ,  $WV_3 = \{1,0,0,0\}$ ,  $WV_4 = \{0,0,1,1\}$  for the four flows, respectively. According to the binary vectors, the Weight Matrix corresponding to flow  $f_1, f_2, f_3, f_4$  is,

$$WM = \begin{bmatrix} WV_1 \\ WV_2 \\ WV_3 \\ WV_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

We number the columns of this WM from left to right as  $column_3$ ,  $column_2$ ,  $column_1$ , and  $column_0$ .

The corresponding WSS (which will be defined in Section II-A) to this Weight Matrix is,

$$1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1.$$

SRR then scans this WSS sequence term by term, when the value of the term is  $i$ ,  $column_{4-i}$  of the WM is chosen. In this column, the scheduler will scan the terms from top to bottom. When the term is not 0, the scheduler will serve the corresponding flow. That is, the flows will be served according to the following service sequence

$$f_3, f_2, f_3, f_4, f_3, f_2, f_3, f_1, f_4, f_3, f_2, f_3, f_4, f_3, f_2, f_3.$$

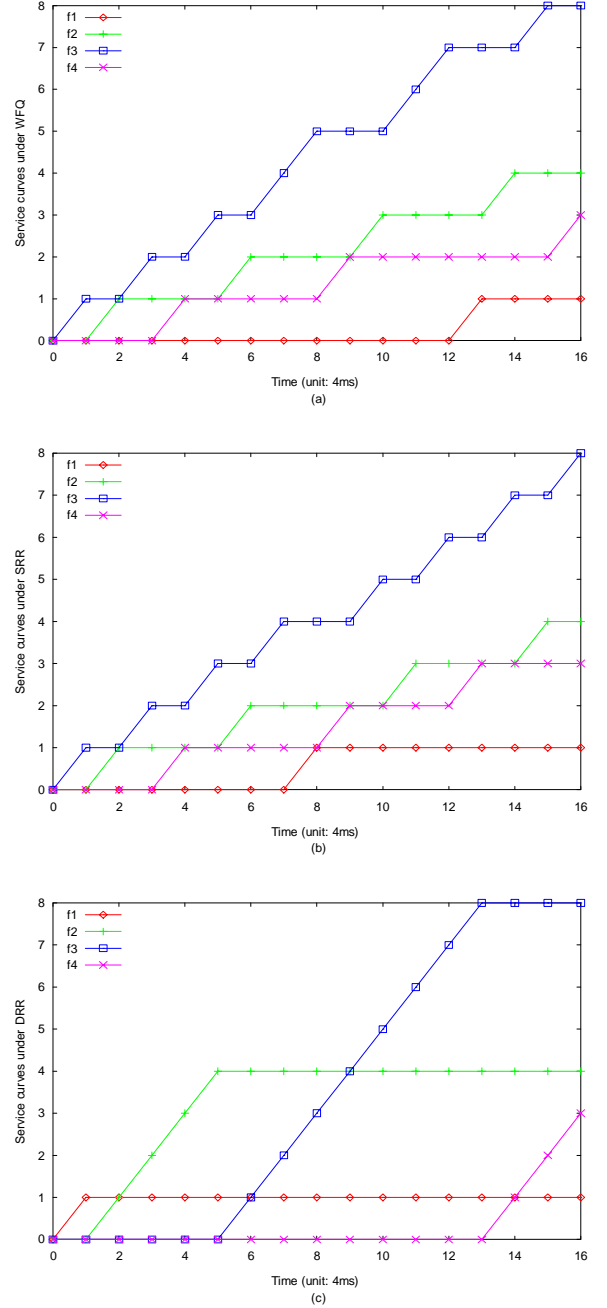


Fig. 1. Service curves of the three schedulers. (a) WFQ. (b) SRR. (c) DRR.

The service curves of the flows in SRR are shown in Fig. 1 together with that in WFQ and DRR. From the above example, we observe that SRR emulates the GPS quite well. Both WFQ and SRR perform much better than DRR in that they have much better delay bound and short-term fairness. It is also easy to observe that the outputs of  $f_2$ ,  $f_3$ , and  $f_4$  in DRR are very bursty.

The rest of this paper is organized as follows. In Section II, the definition of a set of Weight Spread Sequences (WSS) and their properties are presented first, and then the definition of the Weight Matrix is given. In Section III, the formal description of SRR is described. The fairness property, scheduling delay bound, scalability, and space and time complexities of SRR are analyzed in Section IV. In Section V, simulation experiments are designed to compare the end-to-end delay property of SRR with that of WFQ and DRR. Section VI concludes the paper.

## II. THE WEIGHT SPREAD SEQUENCE AND THE WEIGHT MATRIX

### A. The Weight Spread Sequence

*Definition 1:* A set of Weight Spread Sequences (WSS) is defined recursively as,

- 1) The first WSS  $S^1 = 1$ ;
- 2) The  $k$ th WSS is

$$S^k = \{a_i^k\} = S^{k-1}, k, S^{k-1} \quad (1)$$

where  $k > 1$  and  $1 \leq i \leq 2^k - 1$ .

The set corresponding to sequence  $S^k$  is  $\{1, 2, 3, \dots, k\}$ . Let  $len_k$  be the total number of terms of the  $k$ th WSS. It is obvious to see that  $len_k = 2^k - 1$ . The terms of the  $k$ th WSS can be arranged in a circle so that term  $a_{2^k-1}^k$  is next to term  $a_1^k$ .

The *distance* between two terms  $a_m^k$  and  $a_n^k$  of  $S^k$  is defined to be

$$\min[(n - m) \bmod (2^k - 1), (m - n) \bmod (2^k - 1)].$$

Two terms  $a_m^k$  and  $a_n^k$  ( $m > n$ ) are called two adjacent occurrences of element  $i$  ( $1 \leq i < k$ ) if

- 1)  $a_m^k = a_n^k = i$ , and

- 2)  $a_j^k \neq i$  for  $j \in (n, m)$  or  $a_j \neq i$  for  $j \in (m, 2^k - 1] \cup [1, n)$ . For the former case, the *chain* between two adjacent occurrences  $a_m^k$  and  $a_n^k$  for element  $i$  is  $\{a_{n+1}^k, a_{n+2}^k, \dots, a_{m-1}^k\}$ . For the latter case, the *chain* is  $\{a_{m+1}^k, a_{m+2}^k, \dots, a_{2^k-1}^k, a_1^k, a_2^k, \dots, a_{n-1}^k\}$ .

For example, from Definition 1, the 5th WSS is

$$1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1.$$

For this sequence,  $len_5 = 31$ , and the *distance* between two adjacent occurrences of element 3 is 7 or 8.

The following facts about WSS can be derived from its definition.

*Fact 1:* The first  $(2^i - 1)$  terms of a  $k$ th WSS forms an  $i$ th WSS ( $1 \leq i < k$ ).

*Fact 2:* The number of the occurrences of element  $i$  in  $S^k$  ( $1 \leq i \leq k$ ) is  $2^{k-i}$ .

*Fact 3:* A  $(j+k)$ th WSS can be constructed from a  $j$ th and a  $(k+1)$ th WSS by using the following procedure:

for each term  $a_i^j$  of  $S^j$

if ( $a_i^j = 1$ )

replace  $a_i^j$  with  $S^{k+1}$ ;

else

$a_i^j = a_i^j + k$ ;

The proof of Fact 3 is provided in Appendix A. As a special case of Fact 3, a  $(2k)$ th WSS can be constructed from a  $k$ th and a  $(k+1)$ th WSS.

By applying Fact 3,  $S^k$  can be constructed from  $S^{k-i+1}$  and  $S^i$ . Therefore, the following property of WSS holds.

*Fact 4:* The *chain* between two adjacent occurrences  $a_m^k$  and  $a_n^k$  ( $m > n$ ) of element  $i$  in  $S^k$  is

$$\begin{cases} S^{i-1}, S^{i-1} & \text{if } m-n > (n+2^k-1)-m \\ S^{i-1}, x, S^{i-1} & \text{if } m-n < (n+2^k-1)-m \end{cases} \quad (2)$$

where  $1 < i < k$ , and  $i < x \leq k$ . The following property is obvious in view of Fact 4.

*Fact 5:* The *distance* between two adjacent occurrences of element  $i$  ( $1 \leq i < k$ ) in  $S^k$  is either  $2^i$  or  $2^i - 1$ .

Since there is only one occurrence of element  $k$  in  $S^k$ , the distance of element  $k$  is defined to be  $2^k - 1$ .

The definition of WSS is very similar to the Hanoi tower problem in spirit. The intuition behind WSS is to interleave the output of different flows evenly according to their weights, thereby enabling SRR to emulate GPS well even in short time intervals.

## **B. The Weight Matrix**

In various fair queueing schedulers, each flow is assigned a weight in proportion to its reserved rate. In this paper, we assume that the set of weights is  $\{1, 2, 3, \dots, 2^k - 1\}$ . By adjusting the value of  $k$ , different rate allocation ranges can be accommodated. For example, for  $k = 16$ , if the granularity of rate is  $1\text{bps}$ , the set of rates corresponding to the set of weights is  $\{1\text{bps}, 2\text{bps}, 3\text{bps}, \dots, 64\text{kbps}\}$ . For  $k = 32$ , the set of rates is  $\{1\text{bps}, 2\text{bps}, 3\text{bps}, \dots, 4\text{Gbps}\}$ .

The weight of  $flow_f$  can be coded as

$$w_f = \sum_{n=0}^{k-1} a_{f,n} 2^n \text{ where } a_{f,n} \in \{0,1\}.$$

*Definition 2:* The binary coefficients  $a_{f,n}$  of  $w_f$  form a Weight Vector of  $flow_f$ , which is defined as

$$WV_f = \{a_{f,(k-1)}, a_{f,(k-2)}, \mathbb{K}, a_{f,0}\}. \quad (3)$$

*Definition 3:* The Weight Matrix corresponding to flows  $f_1, f_2, \dots, f_N$  is defined as

$$WM = \begin{bmatrix} WV_1 \\ WV_2 \\ \mathbb{M} \\ WV_N \end{bmatrix} = \begin{bmatrix} a_{1,(k-1)} & a_{1,(k-2)} & \mathbb{K} & a_{1,0} \\ a_{2,(k-1)} & a_{2,(k-2)} & \mathbb{K} & a_{2,0} \\ \mathbb{M} & \mathbb{M} & \mathbb{O} & \mathbb{M} \\ a_{N,(k-1)} & a_{N,(k-2)} & \mathbb{K} & a_{N,0} \end{bmatrix}, \quad (4)$$

where  $a_{i,j} \in \{0,1\}$  for  $1 \leq i \leq N$  and  $0 \leq j \leq (k-1)$ . We number the column of the Weight Matrix from left to right as  $column_{k-1}, column_{k-2}, \dots, column_0$ , respectively.

### III. THE SMOOTHED ROUND ROBIN SCHEDULER

We combine the  $k$ th WSS and the  $N \times k$  Weight Matrix to form the Smoothed Round Robin (SRR) scheduler. The basic idea of SRR is based on scanning the WSS and the corresponding Weight Matrix. The WSS is scanned term by term. When the current term is element  $i$ ,  $column_{k-i}$  of the Weight Matrix is selected. For each occurrence of  $a_{f,(k-i)} = 1$  in this column, packet from flow corresponding to the row of  $a_{f,(k-i)}$  is scheduled.

In the following, the formal description of SRR is given.

#### A. Formal Description of SRR

In this work, we focus on packet scheduler. We assume that packets are classified into different flows by a *packet classifier*, the weights of flows are assigned by an *admission controller*, and all input packets are queued to their corresponding queues by a *packet enqueueer*. The tasks of the packet scheduler are to select and forward packets and to maintain related data structures of the scheduler.

In a packet network, if the packet length of a flow is greater than the Maximum Transmission Unit (MTU) of the output link, the system will fragmentize the packet into small pieces. Therefore, we assume that the maximum packet length of all the flows is the MTU of the output link, and denote it as  $L_{\max}$ .

In SRR, we assume that the maximum order of WSS is  $K_{\max}$ . When  $K_{\max} = 32$ , if the bandwidth assignment granularity is  $1bps$ , the set of rates that can be provided by SRR is  $\{1bps, 2bps, 3bps, \dots, 4Gbps\}$ ; if the granularity is  $1kbps$ , the set of rates is  $\{1kbps, 2kbps, 3kbps, \dots, 4Tbps\}$ .

We assume that a flow can be deleted explicitly by a command (i.e., by certain signaling protocols) or implicitly by SRR when the queue corresponding to that flow is empty.

We adopt the following notation for the scheduler:

$K_{\max}$	The maximum order of the WSS used by SRR;
$M$	Weight Matrix of all the active flows;
$S^k$	The $k$ th WSS currently used by the scheduler;
$k$	The order of the current WSS used by SRR;
$P_c$	Index of the current scanning position of the WSS, ranging from 1 to $2^k - 1$ ;
$queue_f$	Queue of the received packets of $flow_f$ , which is a FIFO;
$P_f$	Packet that is at the head of $queue_f$ ;
$L_f$	Length of $P_f$ ;
$w_f$	Weight of $flow_f$ , which is a normalized value according to the bandwidth assignment granularity;
$deficit_f$	A register to memorize how many bytes $flow_f$ should bring to the next round;
$DL_i$	The $i$ th double link, $0 \leq i < K_{\max}$ . There are $K_{\max}$ double links in SRR;
$P_{dl}$	Pointer to a node of a double link;
$L_{\max}$	The upper bound of packet's length of the output link;
$C$	Normalized (according to the bandwidth assignment granularity) bandwidth of the output link.

We use the following 3 pieces of pseudo C code in Fig. 2 to describe the scheduler. There are 3 asynchronous *actions*, namely, **Schedule**, **Add\_flow**, **Del\_flow**. Each action is triggered by some events. **Schedule** is the main part of the scheduler, which is invoked whenever the output link enters a *busy-period*. **Add\_flow** is invoked when a new flow arrives. **Del\_flow** is the action taking place when the flow is deleted explicitly or dead (i.e., the queue of the corresponding flow is empty).

#### Schedule{

local variable:  $f, col$ ; /\* $f$  and  $col$  are the current row and column number of  $M$ , respectively\*/

$P_c=1$ ;  $P_{dl} = head_{k-1} \rightarrow next$ ; /\*initialization\*/

while(in busy-period){

$f = P_{dl} \rightarrow fid$ ;

$deficit_f = deficit_f + L_{\max}$ ;

while( $deficit_f > 0$ ){

if( $L_f \leq deficit_f$ ){

dequeue( $P_f$ );

send( $P_f$ );

$deficit_f = deficit_f - L_f$ ;

if( $queue_f$  is empty){

Del\_flow( $f$ );

#### Add\_flow( $f$ ) { /\* $f$ is the flow ID of this new flow \*/

local variable:  $w_f$ ; /\*  $w_f$  is the weight of  $f$ \*/

assign  $deficit_f, queue_f$ ; /\*  $deficit_f = 0, queue_f$  is empty \*/

use  $w_f$  to form a Weight Vector as shown in Equation (3);

insert nodes at the tails of  $DL_0, DL_1, \dots, DL_{K_{\max}-1}$  according to the coefficients of  $w_f$ ;

if(new columns are added into  $M$ ){

$P_c = P_c \times 2^{j-k}$ ; /\*  $j$  and  $k$  are new and old orders of WSS respectively. \*/

$k = j$ ;

}



flow, the system enters idle state waiting for the next busy-period. Since **Add\_flow** and **Del\_flow** need to update at most  $k$  double links, their time complexities are in proportion to  $k$ , which is the current order of WSS used by SRR.

Since  $M$  is adjusted dynamically according to the weights of flows in SRR, SRR has the following property.

*Fact 6:* The double link  $DL_{k-1}$  is not empty in SRR, where  $k$  is the order of the WSS currently in use in SRR.

In the following section, properties of SRR such as the long-term and short-term fairness, scheduling delay bound, scalability, and time and space complexity are analyzed.

#### IV. PROPERTIES OF SRR

Since SRR always forwards packets when there are active flows in the system, it is therefore work-conserving.

SRR finishes a *round* when it starts from the first term of the  $k$ th WSS, and after visiting all the  $2^k - 1$  terms, back to the beginning of the sequence again.

*Theorem 1:*  $flow_f$  will be visited  $w_f$  times by SRR in a round, where  $w_f$  is the weight of  $flow_f$ .

*Proof:* From Fact 2 and the description of SRR,  $column_i$  (and therefore all the terms belong to  $column_i$ ) of the Weight Matrix  $M$  will be visited  $2^i$  times in a round. Since  $w_f = \sum_{i=0}^{k-1} a_{f,i} 2^i$ , where  $a_{f,i}$  belongs to  $column_i$ ,  $flow_f$  thus will be visited  $w_f$  times in a round. □

Therefore, each flow gets its share in a round according to its weight.

The following lemma is obvious according to the working procedure of SRR.

*Lemma 1:* Suppose  $flow_f$  is backlogged, and has been visited by SRR  $x$  times from time 0 to  $t$ , and  $S_f(0, t)$  denotes the bytes served by SRR of  $flow_f$ , then

$$(x-1)L_{\max} < S_f(0, t) \leq xL_{\max}. \quad (5)$$

##### A. Fairness of the Scheduler

Let  $V_f(0, t)$  be the number of times that  $flow_f$  is visited by SRR from time 0 to  $t$ , and  $\tau$  be the time the scheduler finishes a round. From *Theorem 1*, it is easy to observe that at the end of a round, for any pair of active flows  $f$  and  $g$ , the following lemma holds:

*Lemma 2:* For any pair of backlogged flows  $f$  and  $g$ , at the end of a round in SRR, we have

$$\left| \frac{V_f(0, \tau)}{w_f} - \frac{V_g(0, \tau)}{w_g} \right| = 0. \quad (6)$$

Lemma 2 shows the long-term fairness of SRR. However, SRR can provide more than this. For any pair of active flows  $f$  and  $g$ , we have the following theorem.

*Theorem 2:* For any pair of backlogged flows  $f$  and  $g$  in SRR, at any time instance  $t$ , we have

$$\left| w_f V_g(0, t) - w_g V_f(0, t) \right| \leq \frac{k}{2} \max(w_f, w_g) \quad (7)$$

where  $k$  is the order of the current WSS used by SRR. The proof of Theorem 2 is given in Appendix B.

From Theorem 2, the following corollary can be easily derived.

*Corollary 1:* For any pair of backlogged flows  $f$  and  $g$  in SRR, we have

$$\left| \frac{S_f(0, t)}{w_f} - \frac{S_g(0, t)}{w_g} \right| < \frac{(k+2)L_{\max}}{2 \min(w_f, w_g)} \quad (8)$$

where  $S_f(0, t)$  and  $S_g(0, t)$  denote the service received by flows  $f$  and  $g$  from time 0 to  $t$ , respectively.

*Proof:* From Lemma 1, the following 2 inequalities hold

$$\begin{aligned} (V_f(0, t) - 1)L_{\max} < S_f(0, t) \leq V_f(0, t)L_{\max} \\ (V_g(0, t) - 1)L_{\max} < S_g(0, t) \leq V_g(0, t)L_{\max}. \end{aligned}$$

Hence

$$\begin{aligned} \frac{S_g(0, t)}{w_g} - \frac{S_f(0, t)}{w_f} &< \frac{V_g(0, t)L_{\max}}{w_g} - \frac{(V_f(0, t) - 1)L_{\max}}{w_f} = \left( \frac{V_g(0, t)}{w_g} - \frac{V_f(0, t)}{w_f} \right) L_{\max} + \frac{L_{\max}}{w_f} \\ \frac{S_f(0, t)}{w_f} - \frac{S_g(0, t)}{w_g} &< \frac{V_f(0, t)L_{\max}}{w_f} - \frac{(V_g(0, t) - 1)L_{\max}}{w_g} = \left( \frac{V_f(0, t)}{w_f} - \frac{V_g(0, t)}{w_g} \right) L_{\max} + \frac{L_{\max}}{w_g}. \end{aligned}$$

Thus

$$\begin{aligned} \left| \frac{S_f(0, t)}{w_f} - \frac{S_g(0, t)}{w_g} \right| &< \left| \frac{V_f(0, t)}{w_f} - \frac{V_g(0, t)}{w_g} \right| L_{\max} + \frac{L_{\max}}{\min(w_f, w_g)} \\ &\leq \frac{kL_{\max}}{2 \min(w_f, w_g)} + \frac{L_{\max}}{\min(w_f, w_g)} = \frac{(k+2)L_{\max}}{2 \min(w_f, w_g)}. \end{aligned}$$

□

It is worthy to note that, when  $w_f = 2^m$  and  $w_g = 2^n$  ( $m \geq n$ ) for the backlogged flows  $f$  and  $g$ , the following inequality holds

$$\left| w_f V_g(0, t) - w_g V_f(0, t) \right| \leq \frac{w_f}{2} = 2^{m-1}.$$

Thus, if the weights of all the flows are in the diagonal of the weight matrix, the fairness of SRR will be quite good.

## B. Scheduling Delay Bound of SRR

Suppose  $T_a^p$  is the time a packet becomes the head of  $queue_f$ , and  $T_d^p$  is the time that the scheduler finishes transmitting the packet. We denote the scheduling delay for this packet,  $D_f^p$ , then  $D_f^p = T_d^p - T_a^p$ . We further denote the maximum value of  $D_f^p$ ,  $D_f$ , the scheduling delay bound of  $flow_f$ , that is

$$D_f = \max(D_f^p), \text{ where } p \in flow_f.$$

Regarding  $D_f$ , we have the following theorem.

*Theorem 3:* Suppose there are  $N$  flows, numbered from 1 to  $N$  in SRR. The weight assigned to  $flow_f$  is  $w_f$ , and  $\sum_{f=1}^N w_f \leq C$ ,  $w_f = \sum_{n=0}^i a_{f,n} 2^n$ , where  $a_{f,i} = 1$ , and  $i \leq k-1$ . The scheduling delay bound of  $flow_f$  is

$$D_f < \frac{2L_{\max}}{w_f} + (N-1) \frac{2L_{\max}}{C}. \quad (9)$$

*Proof:* According to SRR, a packet becomes the head of a flow if it is the head of a new flow or the packets before it have left the system. When a packet becomes the head of a flow, it will be served when SRR visits the flow again. A flow is visited when one of its coefficients  $a_{f,n}$  ( $a_{f,n} \neq 0$ ) is visited by SRR. Therefore the delay bound of a flow is the maximum value of the intervals between two adjacent visits by SRR. Let  $V_{cnt}$  denote the sum of times that all the non-zero terms of  $M$  being visited by SRR during this time interval. Based on different values of  $w_f$ , there are two cases.

$$1) 2^i \leq w_f < 2^{i+1} - 1$$

From Fact 4, there must exist a  $y$ , where  $y < i$  and  $a_{f,y} = 0$ . The chain between two terms of element  $(k-i)$  that includes element  $(k-y)$  is  $S^{k-i-1}, (k-y), S^{k-i-1}$ . In this case,  $flow_f$  will be visited again after SRR visits the columns mapped by  $S^{k-i-1}, (k-y), S^{k-i-1}$  and  $column_i$ . Thus

$$\begin{aligned} V_{cnt} &= 2 \sum_{m=1}^N a_{m,i+1} + 2^2 \sum_{m=1}^N a_{m,i+2} + \dots + 2^{k-i-1} \sum_{m=1}^N a_{m,k-1} + \sum_{m=1}^N a_{m,y} + \sum_{m=1}^N a_{m,i} \\ &= \sum_{n=i}^{k-1} (2^{n-i} \sum_{m=1}^N a_{m,n}) + \sum_{m=1}^N a_{m,y} \\ &= \frac{1}{2^i} \sum_{n=i}^{k-1} (2^n \sum_{m=1}^N a_{m,n}) + \sum_{m=1}^N a_{m,y} \\ &= \frac{1}{2^i} (\sum_{n=0}^{k-1} 2^n \sum_{m=1}^N a_{m,n} - \sum_{n=0}^{i-1} 2^n \sum_{m=1}^N a_{m,n}) + \sum_{m=1}^N a_{m,y} \\ &\leq \frac{1}{2^i} (C - \sum_{n=0}^{i-1} 2^n \sum_{m=1}^N a_{m,n}) + \sum_{m=1}^N a_{m,y}. \end{aligned}$$

$$2) w_f = 2^{i+1} - 1$$

In this case, the chain with the maximum length between two adjacent occurrences of element  $(k-i)$  is  $S^{k-i-1}, S^{k-i-1}$ . Similarly, the following inequality holds

$$V_{cnt} \leq \frac{1}{2^i} (C - \sum_{n=0}^{i-1} 2^n \sum_{m=1}^N a_{m,n}).$$

Therefore

$$\begin{aligned} D_f^P &= \max\left(\frac{V_{cnt} \cdot L_{\max}}{C}\right) + \frac{(N-1)L_{\max}}{C} \\ &< \frac{L_{\max}}{C} \left[ \frac{1}{2^i} (C - \sum_{n=0}^{i-1} 2^n \sum_{m=1}^N a_{m,n}) + \sum_{m=1}^N a_{m,y} \right] + \frac{(N-1)L_{\max}}{C} \\ &< \frac{2L_{\max}}{w_f} + \frac{(\sum_{m=1}^N a_{m,y} + (N-1))L_{\max}}{C} \\ &< \frac{2L_{\max}}{w_f} + \frac{2L_{\max}(N-1)}{C}. \end{aligned}$$

where  $(N-1)L_{\max}$  is the maximum deficit the other  $(N-1)$  flows can bring into this time interval.

□

Therefore  $D_f$  is not only in inverse proportion to the weight of the flow, but also in direct proportion to the total number of active flows in SRR. Thus it cannot provide a strictly rate-proportional delay bound. However, the delay bound is still much better than that of DRR.

We have  $D_f < \frac{2L_{\max}}{w_f}$  if  $\frac{1}{2^i} \sum_{n=0}^{i-1} 2^n \sum_{m=1}^N a_{m,n} - \sum_{m=1}^N a_{m,y} \leq 0$  and no flows bring deficit to the next round.

Therefore, in general case, the delay bound is only in inverse proportion to the weight of the flow.

It should be noted that  $D_f$  is different from the local delay bound concept used in WFQ and its variants, where the departure time of a packet is compared with the departure time under GPS [19].  $D_f$  is similar to the concept of WFI of [1], [2]. It has been shown in [1] that  $D_f$  (or WFI) of WFQ is in proportion to  $N$ , where  $N$  is the number of active flows. Thus  $D_f$  (or WFI) of SRR is similar to that of WFQ.

We also note that SRR cannot provide the inequality  $F'-F \leq \frac{L_{\max}}{C}$  as WFQ does<sup>2</sup>. For instance, suppose the packet length is 1, and  $C=16$ , there are 8 flows numbered as  $f_1, f_2, \dots, f_8$  with weight 1. When SRR is serving  $f_1$ , a new flow  $f_9$  with weight 8 comes. In this case, for the first packet of  $f_9$  in SRR,  $F'-F = \frac{N-1}{C} + \frac{1}{C} - \frac{1}{w_9} = \frac{7}{16} > \frac{L_{\max}}{C} = \frac{1}{16}$ .

### C. Scalability of SRR

The scalability of SRR is illustrated in the following aspects.

1. Different rate ranges can be accommodated by WSS of the same order by adjusting the rate granularity. For example, when the granularity of rate is  $1kbps$ , and  $K_{\max} = 16$  (i.e., the order of the WSS is 16), the set of rates is  $\{1kbps, 2kbps, 3kbps, \dots, 64Mbps\}$ . When the rate granularity is  $1Mbps$ , the corresponding set of rate is  $\{1Mbps, 2Mbps, 3Mbps, \dots, 64Gbps\}$ . Therefore, similar WSS can be used in both core routers (switches) and edge routers (switches).

2. SRR can be used in output links with variable bandwidth capacity (e.g., wireless links). According to its working procedure, SRR provides fairness among competing flows even when the bandwidth of the output link varies from time to time.

3. SRR works well regardless of the number of flows. Since the time complexity of SRR is strictly  $O(1)$  (which will be proven in the next subsection), SRR works well even with very large number of flows. This makes SRR attractive for high-speed networks where time complexity is the most critical factor.

### D. Complexity of SRR

From Fact 1, we observe that the WSSs with order  $\{1, 2, 3, \dots, K_{\max} - 1\}$  are contained in the  $K_{\max}$ th WSS. Therefore, only one  $K_{\max}$ th WSS is needed in SRR. When  $K_{\max} = 16$ , the space needed to store the corresponding WSS is  $2^{16}$  bytes (each term of WSS occupies one byte). However, since the length of the WSS increases exponentially with the order of the WSS, it becomes impractical to store the whole sequence statically when  $K_{\max}$  becomes very large. This problem can be overcome by applying Fact 3. By constructing a  $(2k)$ th WSS from a  $k$ th and a  $(k+1)$ th WSS, the space needed can be reduced from  $2^{2k}$  to  $2^{k+1}$ , since the  $k$ th WSS is contained in the  $(k+1)$ th WSS.

We believe that a 32th WSS is enough for current and future packet networks (it can provide up to  $4Tbps$  rate with granularity of  $1kbps$ ). Thus, under this condition, the space complexity of SRR is  $2^{17} + O(N \times K_{\max})$ , where  $K_{\max} \leq 32$ .  $2^{17}$  is the space needed to store a 17th WSS,  $O(N \times K_{\max})$  is the space needed to store the  $K_{\max}$  double links.

We have the following theorem for the time complexity of SRR.

*Theorem 4:* The SRR packet Scheduler needs  $O(1)$  time to choose a packet for transmission,  $O(k)$  time to add or delete a flow, where  $k$  is the order of WSS currently used by SRR.

---

<sup>2</sup> The inequality is Theorem 1.1 of reference [19] (in page 25).

*Proof:* SRR uses the **Schedule** action in Fig. 2 to choose a packet for transmission. It takes the scheduler  $O(1)$  time to choose the flow  $f$ . Then since  $L_{\max} \geq L_f$ , **schedule** will transmit at least one packet for flow  $f$ . After serving  $f$ , **schedule** will update  $P_{dl}$ . If the end of  $DL_{col}$  is not reached, one sentence is needed to update  $P_{dl}$ . If the end of  $DL_{col}$  is met, **schedule** will update  $P_c$  to get the new column number of  $M$ , it may enter the *loop* code. However,  $column_{k-1}$  of  $M$  will be visited at least once in every 2 times according to the construction of WSS. According to Fact 6,  $DL_{k-1}$  is not empty. As a result, the *loop* code can be executed at most 2 times. Hence, the code that updates  $P_{dl}$  and  $P_c$  needs  $O(1)$  time. Therefore, SRR needs  $O(1)$  time to choose a packet for transmission.

Since **Add\_flow** and **Del\_flow** need to update the  $k$  double links in the worst case when flows come and leave SRR, their time complexities are  $O(k)$

□

It should be noted that if a flow is always not backlogged, the **Add\_flow** and **Del\_flow** will be invoked once per packet. Though in SRR (which uses a fixed number of weights,  $2^{K_{\max}}$ ) the time complexities of **Add\_flow** and **Del\_flow** are constant values (at most  $O(K_{\max})$ ), it does introduce a burden that may be comparable to the  $O(\log N)$  incremental step in the time-stamp based schemes.

In [17], we propose to use a timer to delay the deletion of an inactive flow. However, such a mechanism will make SRR not a strictly  $O(1)$  scheme to forward a packet. We also show in [17] that it is difficult to choose the time-out value of the timer. It is a topic for further investigation.

## V. SIMULATION

In this paper, we use simulation to compare the end-to-end delay property of SRR with that of WFQ and DRR. For more experiments (such as experiments on fairness and local scheduling delay), please refer to [17].

### A. Simulation Configuration

The tool we used in our simulation experiment is NS2 [27], to which we added WFQ, SRR scheduling classes, and revised the DRR<sup>3</sup> scheduling class. Though scheduling schemes are always combined with the buffer management schemes tightly, we focus on the effect of scheduling schemes only in the following simulation. In all the following experiments, the buffer size for each queue is set to 50 packets.

---

<sup>3</sup> In ns2.1.b5, the implementation of DRR does not interpret the algorithm in a right way. It deletes a flow when it *deque*s its last packet. However, a flow should be deleted only when the last bit of the last packet leaves the transmission interface.

The network topology is depicted in Fig. 3. There are 12 hosts ( $N_0$ - $N_3$ ,  $M_0$ - $M_3$ , and  $N_4$ - $N_7$ ), and 5 routers ( $R_0$ - $R_4$ ). The propagation delays and bandwidth capacities of the links are shown in TABLE I.

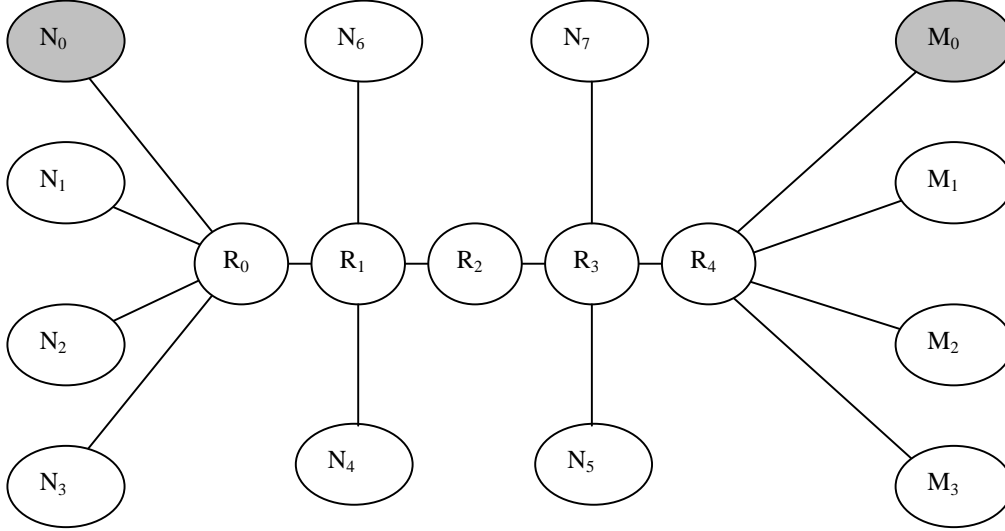


Fig. 3. Network topology of the simulation experiment.

TABLE I

PROPAGATION DELAY AND BANDWIDTH PARAMETERS OF THE LINKS

Links	Propagation delay (ms)	Bandwidth (Mbps)
$N_{[0,3]} - R_0$	0.03	10
$R_0 - R_1$	0.1	6
$R_1 - R_2$	3	15.5
$R_2 - R_3$	3	100
$R_3 - R_4$	0.1	10
$R_4 - M_{[0,3]}$	0.03	10
$N_{[4,6]} - R_1$	0.03	10
$N_{[5,7]} - R_3$	0.03	10

In this simulation,  $R_0$ ,  $R_4$  are edge routers,  $R_1$ - $R_3$  are core routers. A packet from  $N_0$  to  $M_0$  will traverse 6 links.

The following traffic traces are used in the following experiments,

1. There are 2 real-time video streams numbered as flow 11 and 12 between  $N_1$  and  $M_1$ . The total rates of the 2 streams are 1.1Mbps. The video streams are obtained from [21]. One is a cartoon movie named *simpsons* (with average rate 464kbps), and the other is a movie named *golden finger* (with average rate 608kbps). The videos are compressed using an MPEG-1 compliant encoder. The quantization values are: I=10, P=14, and B=18 using the pattern *IBBPBBPBBPBB*, which gives a group of picture (GOP) size of 12.
2. There are 10 flows numbered from 13 to 22 with *Pareto* distribution between  $N_2$  and  $M_2$ . The rate of each flow is 200kbps. These flows are mapped to a same queue in the following experiments, and are used to simulate services with long-range dependency.
3. There are 2 *ftp* flows between  $N_3$  and  $M_3$ . These 2 flows are best effort streams. The rate assigned to the 2 flows is the surplus capacity of link  $R_0R_1$ .
4. There is a *ftp* stream between  $N_5$  and  $N_6$ , and a *telnet* stream between  $N_4$  and  $N_7$ . These flows are best effort services used to consume the redundant bandwidth of the network. All the best effort streams in these experiments are mapped to flow 0.

In the following 3 experiments, we vary the weights of the CBR flows from  $N_0$  to  $M_0$ , and measure the corresponding end-to-end delays of the CBR flows under different scheduling schemes (i.e., WFQ, SRR, and DRR). The packet size of each CBR packet is 250 bytes. The  $L_{\max}$  in SRR is set to 250 bytes, and the minimum *quantum* in DRR is 250 bytes too. CBR flows simulate the real-time audio/video services.

## ***B. Simulation Results***

### *1) The Weights of the CBR Flows Are in the Diagonal*

In the first experiment, there are 7 CBR flows numbered from 1 to 7 between  $N_0$  and  $M_0$ . The rates of the 7 flows are 10kbps, 20kbps, 40kbps, 80kbps, 160kbps, 320kbps, and 640kbps, respectively. In this case, the weights of the CBR flows are in the diagonal of the weight matrix. When all the flows are in the diagonal of the weight matrix, SRR has better fairness property and should perform better than the general case. The mean and maximum end-to-end delays versus the rate of the CBR flows are shown in Fig. 4(a) and Fig. 4(b).

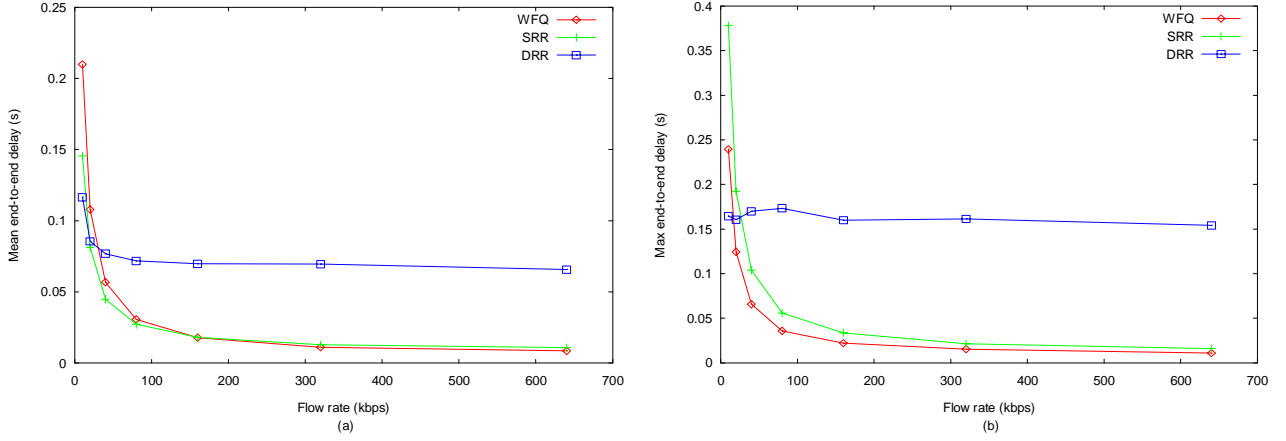


Fig. 4. (a) The mean delays of the CBR flows. (b) The maximum delays of the CBR flows.

### 2) The Weights of the CBR Flows Are Randomly Chosen

In the second experiment, there are 10 CBR flows numbered from 1 to 10 between  $N_0$  and  $M_0$ . The rates of the 10 flows are 10kbps, 10kbps, 20kbps, 20kbps, 40kbps, 80kbps, 80kbps, 160kbps, 260kbps, and 320kbps, respectively. In this case, the weights of the flows are distributed in the weight matrix more randomly than the former experiment. Fig. 5(a) and Fig. 5(b) illustrate the mean and maximum end-to-end delays of the 10 flows.

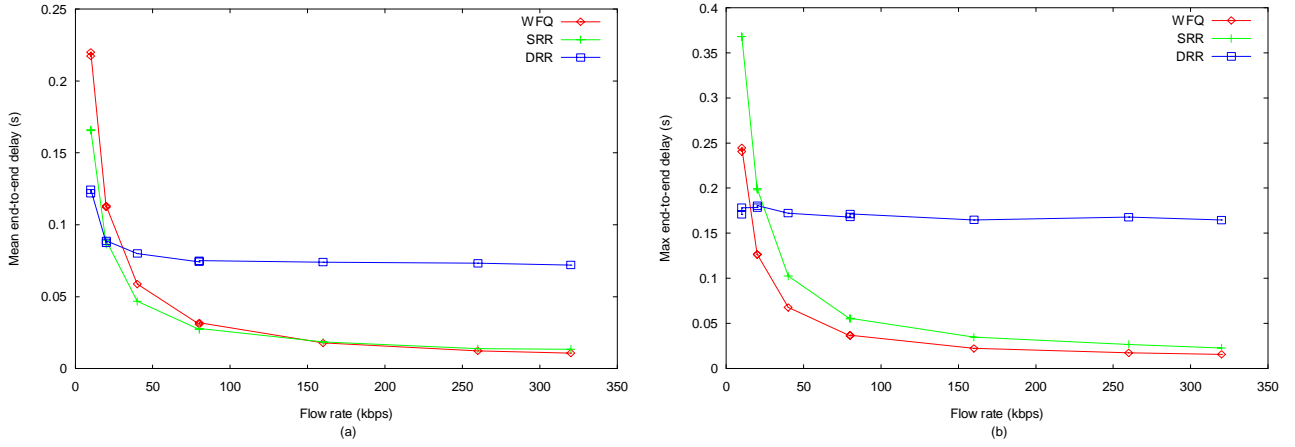


Fig. 5. (a) The mean delays of the CBR flows. (b) The maximum delays of the CBR flows.

### 3) The Weights of the CBR Flows Are Equal

In the third experiment, there are also 10 CBR flows numbered from 1 to 10 between  $N_0$  and  $M_0$ . This time, all the rates of the 10 flows are 100kbps. In this case, SRR retrogresses to DRR. The weights of the CBR flows are in the same column of the weight matrix. We show the end-to-end delays of the 10 flows under SRR (DRR) and WFQ in TABLE II.

TABLE II

THE MAX AND MEAN DELAYS OF THE 10 CBR FLOWS WITH RATE 100KBPS

Flow number	WFQ		SRR (DRR)	
	Max (ms)	Mean (ms)	Max (ms)	Mean (ms)
1	27.2	23.7	31.2	22.2
2	27.8	24.3	31.6	22.7
3	28.6	24.9	31.9	23.2
4	29.2	25.5	33.0	24.1
5	29.7	26.1	32.7	24.8
6	30.4	26.7	33.1	25.3
7	30.7	27.3	33.3	25.8
8	31.4	28.0	33.8	26.2
9	31.8	28.7	34.3	26.7
10	32.7	29.3	34.8	27.3

These experiments show that the end-to-end delay property of SRR is very similar to that of WFQ in all the three cases. The worst-case and mean end-to-end delays of SRR and WFQ decrease with the increasing of the flow rate. These experiments also show that the worst-case end-to-end delay property of SRR is worse than that of WFQ (which conforms to Theorem 3 of this paper), and the mean delay property of SRR is a little better than that of WFQ. For example, as to the flow 4 (80kbps) in the first experiment, the worst-case end-to-end delays under WFQ, SRR, and DRR are 35.9ms, 54.5ms and 173.3ms, and the mean delays are 30.8ms, 25.8ms, and 71.7ms, respectively. Therefore, under this condition, we have

$$D_{\max}^{WFQ} : D_{\max}^{SRR} : D_{\max}^{DRR} = 1 : 1.52 : 4.83$$

$$D_{\text{mean}}^{WFQ} : D_{\text{mean}}^{SRR} : D_{\text{mean}}^{DRR} = 1 : 0.84 : 2.33$$

As to the corresponding flow (flow 6) with the same rate, the worst-case end-to-end delays under WFQ, SRR, and DRR in experiment 2 are 36.2ms, 55.2ms and 167.9ms, and the mean delays are 30.8ms, 25.9ms, and 74.2ms, respectively. Thus, as to flow 6, we have

$$D_{\max}^{WFQ} : D_{\max}^{SRR} : D_{\max}^{DRR} = 1 : 1.52 : 4.64$$

$$D_{\text{mean}}^{WFQ} : D_{\text{mean}}^{SRR} : D_{\text{mean}}^{DRR} = 1 : 0.84 : 2.41$$

In the third experiment, the end-to-end delays of SRR (DRR) and WFQ are very similar. Therefore, the end-to-end delay of SRR is similar to that of WFQ under various conditions. Both WFQ and SRR perform much better than DRR when the weights of flows are not equal. If the flow (flow 4 in the first experiment, flow 6 in the second experiment) is a real-time multimedia stream, it will work well under both WFQ and SRR. However, it will suffer under DRR due to the large worst-case and mean end-to-end delays.

Experiments 1 and 2 also show that the maximum and mean end-to-end delays of DRR do not change much for different flow rates. The mean delay is about 80ms, and the maximum delay is about 170ms for DRR under both experiments. These

experiments show that the worst-case and mean delays of DRR make it not suitable for services with certain delay bound requirements, such as video conferencing.

To summarize, the above simulation results demonstrate that SRR is a qualified scheduler for services that do not have strict end-to-end delay requirements, such as IP telephony and adaptive real-time multimedia streaming services.

## VI. CONCLUSION

We have presented SRR and examined its properties in this paper. A Weight Spread Sequence and a Weight Matrix are introduced as two main data structures of SRR. With the use of the WSS and the Weight Matrix, the output of SRR is distributed more evenly than that of the ordinary round robin schedulers. SRR provides strictly  $O(1)$  time complexity, short-term fairness, and certain scheduling delay bound at the same time.

SRR is attractive for its low time complexity and simplicity. It only needs to store a static WSS, to maintain  $K_{\max}$  double links, and to assign a deficit counter for each flow. It can be implemented in high-speed links at low cost, where efficiency and time complexity are the most important factors. It should be noted that SRR cannot provide a strict scheduling delay bound. As a result, it is not suitable for those applications where strict end-to-end delay bound is needed (i.e., guaranteed services). However, simulation results show that SRR provides good mean and certain worst-case end-to-end delay bounds, thus it is an appropriate scheduler for services where strict delay bound is not required (such as IP telephony and adaptive real-time services).

Though it is now proved that the ideal packet scheduler with strict rate-proportional delay bound and  $O(1)$  time complexity does not exist [31], this paper introduces a new idea to avoid the  $O(\log N)$  limits of various time-stamp based schedulers while still maintaining short-term fairness and certain scheduling delay bound.

We have implemented and tested SRR in the Linux Kernel 2.2.5, the implementation indicates that SRR introduces little cost to the TCP/IP stack<sup>4</sup>, and the experiment results are consonant with that of the simulation results. Our experiments also show that SRR is a suitable scheduling algorithm for the AF PHB of DiffServ and real-time audio/video streaming servers. SRR can also be used as the process (thread) scheduler for interactive time-sharing operating systems.

## APPENDIX A

### ***Proof of Fact 3***

*Proof.* Fact 3 is proved by induction.

1. It is obvious to see that a  $(j + 1)$ th WSS can be constructed by a  $j$ th WSS and a 2nd WSS from the definition of WSS.
2. Suppose a  $(j + k)$ th WSS can be constructed by a  $j$ th and a  $(k + 1)$ th WSS using the procedure.

For a  $(j+k+1)$ th WSS,

---

<sup>4</sup> In our implementation, when there is only one flow in the system, the sums of the time costs for *enque* and *deque* under FIFO, SRR, and WF<sup>2</sup>Q<sup>+</sup> are <1us, 1-2us, and 5-6us respectively with a PIII 500MHZ CPU and 256M DRAM PC. More results can be found in [28].

$$\begin{aligned}
S^{j+k+1} &= S^{j+k}, (j+k+1), S^{j+k} \\
&= S^{k+1}, (k+2), S^{k+1}, \dots, S^{k+1}, (k+2), S^{k+1}, (j+k+1), S^{k+1}, (k+2), S^{k+1}, \dots, S^{k+1}, (k+2), S^{k+1} \\
&= S^{k+2}, (k+3), S^{k+2}, \dots, (j+k+1), \dots, S^{k+2}, (k+3), S^{k+2}
\end{aligned}$$

Therefore, a  $(j+k+1)$ th WSS can be constructed from a  $j$ th WSS and a  $(k+2)$ th WSS.

Thus, Fact 3 follows by induction.

## APPENDIX B

### *Proof of Theorem 2*

*Proof.* We observe that the value of  $|w_f V_g(0, t) - w_g V_f(0, t)|$  only relates to the weights of flows  $f$  and  $g$  in SRR. Though other flows may change the time distribution of  $V_f$  and  $V_g$ , they will not affect the relative service sequence of flows  $f$  and  $g$ . Therefore, only the service sequence includes  $f$  and  $g$  is relevant to the value of  $|w_f V_g(0, t) - w_g V_f(0, t)|$ .

From Lemma 2, we know that at the end of each round,  $|w_f V_g(0, \tau) - w_g V_f(0, \tau)| = 0$ . Thus, we only need to prove Theorem 2 in its first round. Under this condition, we have  $V_f \leq w_f$  and  $V_g \leq w_g$ .

We then prove this theorem by induction:

1. It is obvious that the theorem is correct for  $k = 1$  and  $2$ .
2. Suppose that the inequality is correct using a  $k$ th WSS, i.e., for any pair of  $w_f, w_g$ ,

$$|w_f V_g(0, t) - w_g V_f(0, t)| \leq \frac{k}{2} \max(w_f, w_g).$$

For any pair of flows  $f'$  and  $g'$  using a  $(k+1)$ th WSS,  $w_{f'}$  and  $w_{g'}$  can be expressed as,

$$w_{f'} = 2w_f + a_{f',0}, \quad w_{g'} = 2w_g + a_{g',0}, \quad \text{where } w_{f'} > 1, w_{g'} > 1, a_{f',0}, a_{g',0} \in \{0,1\}.$$

Therefore, the service sequence of flow  $f'$  and  $g'$  can be expressed as,

$$S_{(k+1)}(f', g') = S_k(f, g), \{a_{f',0} \cdot f, a_{g',0} \cdot g\}, S_k(f, g).$$

With different values of  $a_{f',0}$  and  $a_{g',0}$ , there are 4 cases: 1)  $a_{f',0} = 0, a_{g',0} = 0$ ; 2)  $a_{f',0} = 1, a_{g',0} = 0$ ; 3)  $a_{f',0} = 0, a_{g',0} = 1$ ; 4)  $a_{f',0} = 1, a_{g',0} = 1$ .

These 4 cases can be proven with similar method. In what follows, we only show the proof of the last case.

As to this case, when  $V_{f'} = V_f$  and  $V_{g'} = V_g$ ,

$$|w_f V_{g'} - w_g V_{f'}| = |(2w_f + 1)V_g - (2w_g + 1)V_f| \leq |2w_f V_g - 2w_g V_f| + |V_g - V_f| < \frac{k+1}{2} \max(w_{f'}, w_{g'}).$$

When  $V_{f'} = w_f + 1$  and  $V_{g'} = w_g$ ,

$$|w_f V_{g'} - w_g V_{f'}| = |(2w_f + 1)w_g - (2w_g + 1)(w_f + 1)| = |w_g + w_f + 1| < \frac{k+1}{2} \max(w_{f'}, w_{g'}).$$

When  $V_{f'} = w_f + 1$  and  $V_{g'} = w_g + 1$ ,

$$|w_f V_{g'} - w_g V_{f'}| = |(2w_f + 1)(w_g + 1) - (2w_g + 1)(w_f + 1)| = |w_g - w_f| < \frac{k+1}{2} \max(w_{f'}, w_{g'}).$$

When  $V_{f'} = w_f + 1 + V_f$  and  $V_{g'} = w_g + 1 + V_g$ ,

$$\begin{aligned} |w_f V_{g'} - w_g V_{f'}| &= |(2w_f + 1)(w_g + 1 + V_g) - (2w_g + 1)(w_f + 1 + V_f)| = |2w_g V_f - 2w_f V_g + w_g - V_g - w_f + V_f| \\ &\leq |2w_g V_f - 2w_f V_g| + |w_g - V_g - w_f + V_f| < |2w_g V_f - 2w_f V_g| + \max(w_{f'}, w_{g'}) \leq \frac{k+1}{2} \max(w_{f'}, w_{g'}). \end{aligned}$$

Hence, for different combinations of  $V_{f'}$  and  $V_{g'}$  in the last case, we have

$$|w_f V_{g'} - w_g V_{f'}| \leq \frac{k+1}{2} \max(w_{f'}, w_{g'}).$$

Therefore, Theorem 2 follows by induction.

### ACKNOWLEDGMENT

The author would like to thank Prof. Wangdong Qi for introducing the names of SRR and WSS and for his constructive comments and generous help. The author thanks Jun Wang and Yu Sun for implementing SRR in the Linux kernel, Juan Chen and Cong Ling for their valuable comments. We thank the anonymous reviewers and Prof. Roch Guerin for their valuable comments and suggestions. Lastly we would like to thank Dr. Zhensheng Zhang for carefully proofreading this manuscript.

### REFERENCES

- [1] J. Bennet and H. Zhang, "WF<sup>2</sup>Q: worst-case fair weighted fair queueing," in *Proc. INFOCOM'96*, 1996.
- [2] J. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proc. SIGCOMM'96*, 1996.
- [3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services," *RFC 2475*, Dec. 1998.
- [4] J. Bolot and T. Turletti, "Experience with control mechanisms for packet video," in *Proc. SIGCOMM'97*, 1997.
- [5] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, Aug. 1998.

- [6] D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. SIGCOMM'88*, 1988.
- [7] D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," in *Proc. SIGCOMM'92*, 1992.
- [8] J. Cobb, M. Gouda, and A. El-Nahas, "Time-shift scheduling—fair scheduling of flows in high-speed networks," *IEEE/ACM Trans. Networking*, vol. 6, June 1998.
- [9] J. A. Cobb and M. G. Gouda, "Flow theory," *IEEE/ACM Trans. Networking*, vol.5, Oct. 1997.
- [10] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. SIGCOMM'89*, 1989.
- [11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, Aug. 1993.
- [12] S. Floyd and V. Jacobson, "Link-share and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, Aug. 1995.
- [13] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Networking*, vol. 7, Aug. 1999.
- [14] L. Georgiadis, R. Guerin, and R. Rajan, "Efficient support of delay and rate Guarantees in an Internet," in *Proc. SIGCOMM'96*, 1996.
- [15] P. Goyal, H. M. Vin, and H Cheng, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Networking*, vol. 5, 1997.
- [16] P. Goyal and H. Vin, "Generalized guaranteed rate scheduling algorithms: a framework," *IEEE/ACM Trans. Networking*, vol. 5, 1997.
- [17] C. Guo, "A SRR packet scheduler for flows in multi-service packet networks," Ph.D. thesis, Inst. of Comm. Eng. of China, Apr. 2000.
- [18] S. R. McCanne, "Scalable compression and transmission of Internet multicast video," Ph.D. thesis, UC. Berkeley, Dec. 1996.
- [19] A. Parekh, "A generalized processor sharing approach to flow control in integrated services network," Ph.D. thesis, Dept. Elect. Eng. and Comput. Sci., M.I.T., Feb. 1992.
- [20] V. Paxson and S. Floyd, "Why we don't know how to simulate the Internet," from: <ftp://ftp.ee.lbl.gov/papers/wsc97.ps>.
- [21] O. Rose, "Traffic modeling of variable bit rate MPEG video and its impacts on ATM networks," Ph.D. thesis, Wurezbunger, Bericht, Feb. 1997.

- [22] D. Saha, S. Mukherjee, and S. Tripathi, "Carry-over round robin: a simple cell scheduling mechanism for ATM networks," *IEEE/ACM Trans. Networking*, vol.6, Dec. 1998.
- [23] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *Proc. SIGCOMM'95*, 1995.
- [24] D. Stiliadis and A. Varma, "Rate-proportional servers: a design methodology for fair queueing algorithms," *IEEE/ACM Trans. Networking*, vol. 6, Apr. 1998.
- [25] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Trans. Networking*, vol. 6, Apr. 1998.
- [26] A. Varma and D. Stiliadis, "Hardware implementation of fair queuing algorithms for asynchronous transfer mode networks," *IEEE Commun. Mag.*, vol. 35, Dec. 1997.
- [27] The VINT Project, "NS notes and documentation", from <http://www-mash.cs.berkeley.edu/ns/nsDoc.ps.gz>.
- [28] J. Wang, "Study and implementation of packet schedulers for DiffServ," Master thesis, Inst. of Comm. Eng. of China, Mar. 2001.
- [29] W. Weiss, "QoS with differentiated services," *Bell-labs Technical Journal*, Oct.-Dec. 1998.
- [30] W. Willingers and V. Paxson, "Where Mathematics meets the Internet," *Notes of the American Mathematical Society*, vol.45, Aug.1998.
- [31] J. Xu and R. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms," in *Proc. SIGCOMM'02*, 2002.
- [32] L. Zhang, "A new architecture for packet switching network protocols," Ph.D. thesis, Dept. Elect. Eng. and Comput Sci., M.I.T., Aug. 1989.
- [33] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new Resource ReReservation Protocol," *IEEE Network*, 1993.

## Author Biographies

Chuanxiong Guo [S'99-M'01] received a Ph.D. degree in communications and information systems in 2000 from the Institute of Communications Engineering, Nanjing, China. In 2001 he spent the year to run a research project sponsored by the National High Technology Research and Development Program of China (the 863 Program). He is now at Microsoft Research Asia as a researcher. His current research interests include multi-service support for wired and wireless networks, mobility management, networking support in operating systems, networked worm/virus defense.

## List of Figure Captions

Fig. 1. Service curves of the three schedulers. (a) WFQ. (b) SRR. (c) DRR.

Fig. 2. Description of SRR.

Fig. 3. Network topology of the simulation experiment.

Fig. 4. (a) The mean delays of the CBR flows. (b) The maximum delays of the CBR flows.

Fig. 5. (a) The mean delays of the CBR flows. (b) The maximum delays of the CBR flows.

## List of Table Captions

TABLE I PROPAGATION DELAY AND BANDWIDTH PARAMETERS OF THE LINKS

TABLE II THE MAX AND MEAN DELAYS OF THE 10 CBR FLOWS WITH RATE 100KBPS