Rate Proportional SCFQ Algorithm for High-Speed Packet-Switched Networks

Byung-Hwan Choi and Hong-Shik Park

Self-Clocked Fair Queueing (SCFQ) algorithm has been considered as an attractive packet scheduling algorithm because of its implementation simplicity, but it has unbounded delay property in some input traffic conditions. In this paper, we propose a Rate Proportional SCFQ (RP-SCFQ) algorithm which is a rate proportional version of SCFQ. If any fair queueing algorithm can be categorized into the rate proportional class and input is constrained by a leaky bucket, its delay is bounded and the same as that of Weighted Fair Queueing (WFQ) which is known as an optimal fair queueing algorithm. RP-SCFQ calculates the timestamps of packets arriving during the transmission of a packet using the current value of system potential updated at every packet departing instant and uses a starting potential when it updates the system potential. By doing so, RP-SCFQ can have the rate proportional property. RP-SCFQ is appropriate for high-speed packet-switched networks since its implementation complexity is low while it guarantees the bounded delay even in the worst-case input traffic conditions.

Manuscript received January 27, 2000 ; revised July 15, 2000.
The authors are with Information And Communications University, Taejon , Korea.
Byung-Hwan Choi (e-mail: bhchoi33@icu.ac.kr)
Hong-Shik Park (phone: +82 42 866 6120, e-mail: hspark@icu.ac.kr)

I. INTRODUCTION

Traffic scheduling is a critical component of future integrated-services packet networks that will provide a broad range of Quality-of-Service (QoS) guarantees. These guarantees are usually in the form of bounds on end-to-end delay, bandwidth, delay jitter (variation in delay), packet loss rate, or a combination of these parameters. The function of a traffic scheduling algorithm is to select, for each outgoing link of the switch, a packet to be transmitted in the next cycle from the available packets belonging to the flows sharing the same outgoing link. Design of a scheduling algorithm involves an inevitable tradeoff among its delay, complexity of implementation, and fairness.

As regards fairness and end-to-end delay performance are concerned, Generalized Processor Sharing (GPS) [1], where the input traffic is treated as a fluid-flow, has been thought as the most ideal concept, but it is not feasible to be implemented in the real environment. Weighted Fair Queueing (WFQ), or interchangeably Packet-by-packet Generalized Processor Sharing (PGPS) [1], [2] can be the best approximation of it. Though WFQ can guarantee the performance close to that of GPS, it requires very high implementation complexity and so it is not appropriate for the high-speed network. Self-Clocked Fair Queueing (SCFQ) [3] was proposed to improve the complexity of WFQ, but it has worse delay performance than that of WFQ. Its delay bound directly depends on the number of sessions since all packets that arrive during a packet's transmission always have larger timestamps than that of the packet being transferred.

On the other hand, a framework called Rate Proportional Server (RPS) [4] has been defined by D.Stiliadis. As far as any scheduling algorithm belongs to the category of RPS, its delay bound is identical to that of WFQ, although its fairness and im-

plementation complexity may be different. For example, WFQ and Virtual Clock [5] belong to the RPS class. So their delay bounds are same. The fairness of WFQ looks like that of GPS closely because it simulates the algorithm with very high complexity, while the fairness of Virtual Clock is unbounded. Due to its intrinsic property of the system potential, SCFQ can not be classified into the RPS category. Its delay bound increases linearly with the number of connections it serves.

Starting Potential-based Fair Queueing (SPFQ) [6] proposed by D. Stiliadis performs the re-calibration at packet boundaries, resulting in a fairness bound that is equal to that of WFQ, still maintaining the O(1) timestamp computation, but its system potential function has complexity of O(log V).

Aiming to improve the delay property of SCFQ, we propose a new scheduling algorithm named Rate Proportional Self-Clocked Fair Queueing (RP-SCFQ), which provides a simple mechanism to maintain the system potential without simulation of the GPS server. As the name implies, RP-SCFQ can be categorized into the RPS class, so it guarantees the end-to-end delay performance even in the heterogeneous network environment. The main focus of our idea is to improve the delay bound of SCFQ while maintaining its implementation complexity still at the lower level.

Organization of the paper is as follows. In Section II, the comparative survey and analysis on fair queueing algorithms are briefly given. Then, the concept and operational principle of the proposed algorithm are presented in Section III, and the performance criteria of the algorithm are also discussed in Section III. The simulation results concerning the delay and fairness performance of the algorithm are given in Section IV. Other algorithms such as WFQ, SCFQ, and SPFQ are taken into account for the performance comparison. Finally, the conclusive words are presented in Section V.

II. FAIR QUEUEING ALGORITHMS

1. Background

In general packet scheduling algorithms can be classified into work-conserving or non work-conserving [7] from the viewpoint of the mode of server operation. For the work-conserving algorithm, a server can never be idle as long as there are packets waiting for service in the system. On the other hand, non work-conserving algorithms allow the server to stay in the idle state until predefined conditions (eligibility) are met even though there are the packets waiting in the system. So, non work-conserving algorithms can guarantee the tighter bound for delay and delay jitter, but they experience inefficient utilization of server's capacity. Due to this fact, non work-conserving algorithms are not suitable for the high-speed packet network. Even though server utilization is comparably high in work-conserving algorithms, a proper algorithm should be considered to guarantee minimum bandwidth per each traffic flow as contracted and to fairly allocate the unused capacity of the server to all backlogged flows. In this section, we focus on the work-conserving algorithms and look into the representative algorithms pertinent to this category.

The GPS algorithm is based on the fluid-flow traffic model. All traffic flows are modeled as fluid-flows, so it is assumed that all backlogged traffic flows are serviced simultaneously by the server and traffic is considered to be infinitely divisible. It provides ideal fairness among backlogged traffic flows and optimal delay bound. These characteristics are used as the reference for all other realizable work-conserving algorithms. It is not possible, however, to implement the GPS algorithm in the real network environment because its basic assumptions are not practical. In reality, each packet network defines its own packet size as a minimum traffic unit it can deliver and a server can only serve one packet from one flow at a time. Therefore, many packet-based algorithms have been proposed so far to approximate the performance of GPS with practical applicability in real networks.

Most GPS-like algorithms depend on the timestamp allocated per packet on its arrival to decide the transmission order for packets waiting in queue. In other words, a packet is provided with the timestamp before it is located at queue for service and the server chooses a packet whose timestamp is minimum among all Head of Line (HOL) packets from backlogged sessions at the moment it becomes idle. As a reference value for the calculation of timestamp, each algorithm maintains its own system potential (or virtual time), which tracks the amount of work that is done by the server. The system potential is updated at every event such as a packet arrival or a departure during the server busy period. Let TS_i^k be the timestamp of the k-th packet of session i and $P(a_i^k)$ be the system potential at the moment the k-th packet arrives from session i. General procedure for the timestamp calculation is shown below:

$$TS_i^0 = 0, \text{ for all } i,$$
$$TS_i^k = \max\{TS_i^{k-1}, P(a_i^k)\} + \frac{L_i^k}{\rho_i}, \quad (1)$$

where L_i^k is the length of the k-th packet from session i and ρ_i is the rate allocated to session i.

WFQ [2] is the packet-based version of the GPS algorithm. To simulate the GPS server, at every event of a packet arrival or a packet departure, the system potential is updated. As (1), upon arrival of a packet, the timestamp is calculated based on the system potential as well. Let $P(t)$ be the system potential when the j-th event happens and τ be the time period between

any two adjacent events. Then,

$$P(0) = 0,$$
$$P(t) = P(\tau) + \frac{(t-\tau)}{\sum_{i \in B_j} \rho_i}, \quad (2)$$

where B_j represents the set of backlogged sessions. From (2), note that the system potential is the function of the number of backlogged sessions during an observed period. WFQ can approximate the GPS algorithm closely, but has the complexity of O(V) because it is required to update the system potential V times during the transmission of one packet in the worst case, where V means the number of sessions. WFQ algorithm is not easy to be implemented for the high-speed network due to its calculation complexity of the system potential.

As an important trial to improve the limitation of WFQ, Golestani proposed a novel algorithm called SCFQ. As its name implies, the timestamp of the packet currently being served is set as the system potential. It decreases the complexity dramatically because the system potential is updated only once during the transmission of a packet. Let $\hat{s}_j^l$ and d_j^l be the service start time and service finish time of the packet p_j^l that is being serviced at any time, and F_j^l be the timestamp of p_j^l. Then the system potential when a packet arrives at time t is given as follows:

$$P(a_i^k) = \hat{F}_j^l, \quad \hat{s}_j^l < t \le \hat{d}_j^l. \quad (3)$$

The timestamp for the new packet is calculated according to (1) using the system potential. Even though the complexity is greatly improved, the delay bound of SCFQ becomes dependent on the number of sessions. With this scheme, maximum V packets may have the same timestamp as that of the packet being served, and even a packet that arrives to an empty queue can not be served because its timestamp becomes always bigger than that of the packet being served. If there are multiple packets whose timestamps are same as that of the packet being served, the newly arrived packet is delayed until all those packets are transmitted. Therefore, the delay bound of SCFQ linearly depends on the number of the backlogged connections.

2. Virtual Time Function of WFQ Algorithm

In order to explain the explicit difference between WFQ and SCFQ, Hui Zhang presented an extreme example in [7]. We found that there was misunderstanding for the service order of WFQ due to the wrong computation of virtual time.

There are 11 sessions as shown in Fig. 1(a), whose allocated rates are 0.5 for session 0 and 0.05 for other 10 sessions, respectively. For convenience, the server rate and packet length are assumed as 1. All sessions keep their traffic contract, so packets arrive at every 2 time slots from session 0 and at every 20 time slots from session 1-10. Figure 1(b) shows the comparison of the service order of WFQ and SCFQ misled in [7]. H. Zhang explained that the packets from session 0 are assigned with the timestamp spaced by 2 exactly. But this is wrong since the virtual time of WFQ depends on the accumulated sum of backlogged session rates during any time period, the increase of system potential becomes larger than 2 as session 1 to 10 complete service.

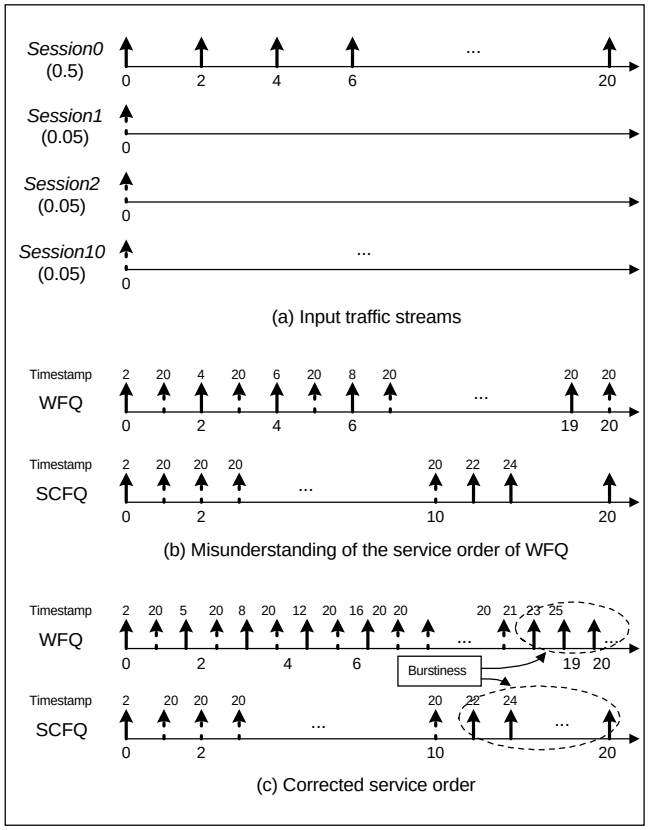


Fig. 1. Comparison of service order.

Let's explain the correct service order of WFQ. All packets that arrive at time 0 are assigned with the timestamps according to their rates because the system potential is initially zero. Among all those first packets, one from session 0 is selected for service because its timestamp is smallest at that time. Since it takes 1 time unit for service completion of the packet, the first packet of session 0 completes service at time 1 and the virtual time is set to 1. Because there is no packet waiting in session 0 and the server operates in the work-conserving mode, one of packets from session 1-10 is chosen for next service. At time 2, a new packet arrives from session 0, and the system potential is computed as 3 (i.e., 1+1/0.5) because the accumulated rate between time slot 1 and 2 is 0.5. The timestamp of the second packet of session 0 is assigned as 5 (i.e., 3 + 1/0.5) so this pack-

et is located in service. Similarly, at time 3, one of packets from session 2-10 is chosen for service, and a new packet arrives in session 0 at time 4. As the sessions complete the service, the virtual time increases quicker than the real time. The 6th packet from session 0 arrives at time 10 and its timestamp exceeds 20, so it is delayed until all other packets whose timestamps equal 20 complete service. As shown in Fig. 1(c), the 6th and following packets from session 0 are delayed until all packets from session 5-10 complete service. Therefore, WFQ as well as SCFQ yields some level of burstiness marked with the dotted circle in Fig. 1(c) for this type of input pattern.

3. Rate Proportional Server

In this sub-section, we review the theory of RPS formulated by Stiliadis and Varma in [4], which we use in the next section to evaluate the delay property of RP-SCFQ. As mentioned in Section I, it is important for scheduling algorithms to guarantee the end-to-end delay bound specially in the heterogeneous network environment. It was proven that the algorithms categorized into RPS guarantee the same delay bound as that of WFQ.

A fluid-flow service discipline is an RPS if its system potential satisfies the following two conditions:

(a) For any interval $(t_1, t_2]$ during a system busy period (i.e., a period of activity of the server)

$$P(t_2) - P(t_1) \geq t_2 - t_1 . \tag{4}$$

(b) At any time t, the system potential is not higher than the minimum potential of all backlogged sessions;

$$P(t) \leq \min_{i \in B(t)} P_i(t), \tag{5}$$

where $B(t)$ is the set of backlogged sessions, and $P_i(t)$ is the potential of session i.

In an RPS, the evolution of $P_i(t)$ is completely specified by the following rule:

(a) $P_i(t)$ is constant as long as session *i* is not backlogged.

(b) If session *i* becomes backlogged at time τ, then

$$P_i(\tau) = \max\{P_i(\tau^-), P(\tau^-)\}. \tag{6}$$

(c) For any time $t > \tau$ such that session i remains backlogged during the whole time interval $(\tau, t]$,

$$P_i(t) = P_i(\tau) + \frac{W_i(\tau, t)}{\rho_i}, \tag{7}$$

where $W_i(\tau, t)$ is the total amount of service received by session i during $(\tau, t]$.

$P_i(t)$ regulates the access of session *i* to the server since only sessions with the minimum potential are serviced. Every serviced session gets an instantaneous amount of service that is proportional to the reserved rate, so that all the potentials of the serviced sessions increase at the same rate. The methodology that maintains the system potential to meet the basic conditions and achieve the bounded short-term unfairness has been suggested in [4].

III. RATE PROPORTIONAL SELF-CLOCKED FAIR QUEUEING

1. Description of the Algorithm

In SCFQ, the first packet of an empty session is tagged with the timestamp based on the service tag of the packet being transmitted. Since the timestamp of this packet shall always be larger than that of the packet being serviced regardless of its session rate, the new packet has to be delayed until all other packets with the less service tag are serviced. Due to this property, the delay bound of SCFQ linearly depends on the number of connections. On the other hand, SCFQ provides great benefit of simplicity for implementation because it reduces the computational complexity for the system potential. As mentioned earlier, there is a trade-off relationship between performance and complexity. We argue that the delay performance of SCFQ can greatly be improved if we replace its system potential with the one whose property inherits the rate proportionality. As an alternative, we choose the minimum starting potential, originally proposed in SPFQ, among all backlogged sessions as the system potential at the departing moment of a packet. It can describe well the state of the hypothetical system without complex GPS simulation. No matter how many packets whose timestamps are identical to that of the packet being served are waiting for service when a new packet arrives at an empty session, it can be served in proportion with its allocated rate. Followings are the detailed description of the RP-SCFQ algorithm.

A. When the *k*-th packet p_i^k from session i arrives, the timestamp TS_i^k is calculated and tagged.

$$TS_i^k = \max\{TS_i^{k-1}, P(a_i^k)\} + \frac{L_i^k}{\rho_i}, \tag{8}$$

where $P(a_i^k)$ is the system potential at arrival time a_i^k of the packet P_i^k. In addition, L_i^k and ρ_i is the length of the packet P_i^k and the allocated rate of session i, respectively.

B. When the *m*-th packet departs at time d^m, the system potential is updated and a new packet is chosen for the next transmission as follows:

(a) Calculate the temporal system potential $P'(d^m)$ by adding up the actual transmission time of the *m*-th packet onto the previous system potential $P(d^{m-1})$ updated at

$(m-1)$-th packet's departure.

$$P'(d^m) = p(d^{m-1}) + \frac{L^m}{r}, \tag{9}$$

where L^m and r are the length of the m-th packet and the server rate, respectively.

(b) Calculate the minimum starting potential among all backlogged sessions and set it as the base potential $B^P(d^m)$.

$$B^P(d^m) = \min_{j \in B(d^m)} SP_j, \tag{10}$$

where SP_j means the starting potential of session j and $B^P(d^m)$ is the set of backlogged sessions at departing time d^m.

(c) Update the system potential.

$$P(d^m) = \max\{p'(d^m), B^P(d^m)\}. \tag{11}$$

(d) Choose the packet whose timestamp is minimum among all backlogged sessions for the next transmission.

$$NextPacket(d^m) = \min_{j \in B(d^m)} TS_j, \tag{12}$$

where TS_j is the timestamp of the HOL packet of session j and $B(d^m)$ is the set of backlogged sessions at departure time d^m.

Now we are going to show that the system potential function of RP-SCFQ has the rate-proportional property, so it can be classified into the RPS class. As presented in Section II, there are two conditions for a scheduling algorithm to be classified into the RPS class.

Theorem 1. RP-SCFQ is a Rate Proportional Server.

Proof. As mentioned in Section II, a scheduling algorithm must satisfy two main properties of RPS for both the session and system potential.

A) Property of session potential $P_i(t)$

The session potential remains constant in the case that relevant session is not backlogged. If a session becomes backlogged at time τ, the potential $P_i(\tau)$ of session i is decided as

$$P_i(\tau) = \max\{P(\tau), P_i(\tau)\}, \tag{13}$$

where $P(\tau)$ is the system potential at time τ and $P_i(\tau)$ is the potential of relevant session i, which was updated at the previous session busy period during the same system busy period. Session potential increases by the normalized amount of serviced traffic during the time period $(\tau, t]$ that the session is continuously backlogged.

$$P_i(t) = P_i(\tau) + \frac{W_i(\tau, t)}{\rho_i}, \tag{14}$$

where $W_i(\tau, t)$ is the total amount of service received by session i during $(\tau, t]$. So the session potential $P_i(\tau)$ satisfies the properties of RPS.

B) Property of system potential $P(t)$

B.1 For any time period $(t_1, t_2]$ during the system busy period, the change of the system potential must be more than the increase of real time. That is,

$$P(t_2) - P(t_1) \geq (t_2 - t_1).$$

[*Proof for B.1*]

1) If the server is idle, the system potential remains zero:

$$P(0) = 0.$$

2) In the fluid model, the system potential of RP-SCFQ is updated as follows for any period $(t_1, t_2]$ in the system busy period, as explained in Section III-1. $W(t_1, t_2)$ means the total amount of service provided during $(t_1, t_2]$ by the server.

$$P(t_2) = \max\{B^P(t_2), P'(t_2)\}, \tag{15}$$

where $B^P(t_2) = \min_{j \in B(t_2)} SP_j$ and $P'(t_2) = P(t_1) + \frac{W(t_1, t_2)}{r}$.

We have to consider two possible cases here:

[Case 1] $B^P(t_2) < P'(t_2)$

$$P(t_2) = P(t_1) + \frac{W(t_1, t_2)}{r},$$

$$P(t_2) - P(t_1) = \frac{W(t_1, t_2)}{r} = (t_2 - t_1),$$

or equivalently,

$$P(t_2) - P(t_1) = (t_2 - t_1),$$

since $W(t_1, t_2) = r(t_2 - t_1)$.

[Case 2] $B^P(t_2) \geq P'(t_2)$

$$P(t_2) = B^P(t_2) \geq P(t_1) + \frac{W(t_1, t_2)}{r},$$

$$P(t_2) - P(t_1) \geq \frac{W(t_1, t_2)}{r},$$

or equivalently, $P(t_2) - P(t_1) \geq (t_2 - t_1)$.

Therefore, we can conclude that RP-SCFQ satisfies the first requirement $[B.1]$.

B.2 For any time *t* in the system busy period, the system potential must satisfy the following property that its value never exceeds the minimum potential of backlogged connections in the corresponding fluid server;

$$P(t) \leq \min_{j \in B(t)}(P_j(t)).$$

[*Proof for B.2*]

We can easily prove the requirement by contradiction. Let i be a connection with the minimum potential in the fluid server at time *t*. If the potential $P_i(t)$ of session i in the fluid server is less than the base potential $B^P(t)$ that is the minimum starting potential among all backlogged sessions, and if connection *i* has received more service until time *t* in the packet-by-packet server than in the fluid server, due to the property of work-conserving server, there is another connection *k* with the potential $P_k(t) \leq P_i(t)$ that has received less service in the packet-by-packet server than in the fluid-server. This means that the packet most recently serviced from the connection *k* in the fluid-server has not yet finished service in the packet-by-packet server. Let SP_k be the starting potential of this packet. Then,

$$SP_k \leq P_k(t) \leq P_i(t). \tag{16}$$

Since $P_i(t) < B^P(t)$ from the original assumption, we must have $SP_k < B^P(t)$. This result contradicts with the definition of the base potential that is the minimum starting potential among all backlogged sessions. Therefore, (5) is satisfied for any *t* during the server busy period.

At this point, we return to the case presented in Fig. 1 to see how RP-SCFQ treats such input condition. Figure 2 shows the service order of RP-SCFQ for the same input traffic pattern in Fig. 1(a). In RP-SCFQ, the system potential function is based on the minimum starting potential among all backlogged sessions. The timestamps of the packets from session 0 increase by 2 based on that of the previous packet from the same session rather than the system potential as long as there are backlogged sessions whose starting potentials are smaller than that of any packet from session 0. Even for this extreme case, RP-SCFQ doesn't generate any burst of the packets at the output link.

2. Delay Bound of RP-SCFQ

Delay bound refers to the maximum transmission delay that a packet from a session can experience. It depends on the reference model of the source of the packets and the latency of the server. The latency is the worst-case transmission delay seen by the first packet coming from session i after an idle period.

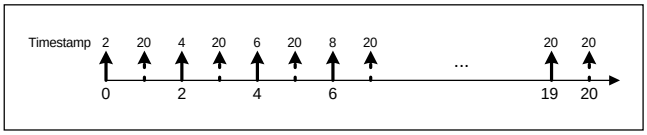


Fig. 2. The service order of RP-SCFQ.

Theorem 2. The end-to-end delay bound of RP-SCFQ is equal to that of WFQ. Let D_i be the delay bound of session *i*, and traffic sources are assumed to be constrained by the leaky bucket with the parameter (σ_i, ρ_i) where σ_i and ρ_i are the allowable burstiness and allocated rate of session *i*, respectively. Then,

$$D_i \leq \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{k} \Theta_i^{S_i} - \frac{L_i}{\rho_i}, \tag{17}$$

where $\Theta_i^{S_i} = \frac{L_{\max}}{r} + \frac{L_i}{\rho_i}$, which is the latency of RP-SCFQ for session i at the *j*-th node along the route.

Proof. Since we have shown that RP-SCFQ belongs to the RPS category in **Theorem 1**, its delay bound basically follows the result of RPS. If the *k*-th packet P_i^k from session i initiates the session backlogged period at time τ, its timestamp becomes

$$TS_i^k = P(\tau) + \frac{L_i^k}{\rho_i}, \tag{18}$$

where $P(\tau)$ is the system potential at time τ. The worst case delay of the packet P_i^k can be considered as the latency of session i since it arrives into an empty queue. And, as the system potential $P(\tau)$ is the minimum starting potential among all backlogged sessions, the packet P_i^k is privileged to share the smallest starting potential. Since it is the first packet of a new backlogged period of session *i*, all packets whose timestamps are less than that of the packet P_i^k complete service before the packet P_i^k receives service. At the worst case, if a packet with the maximum length L_{max} starts to receive service when the packet P_i^k arrives at the system, then the overall delay (or equivalently the latency) Θ_i^k of the packet P_i^k can be calculated as follows:

$$\begin{aligned}\Theta_i^k &\leq \frac{L_{\max}}{r} + \frac{1}{r}(L_i + \sum_{j \neq i} \frac{L_i}{\rho_i}\rho_j) \\ &\leq \frac{L_{\max}}{r} + \frac{1}{r}(L_i + \frac{(r-\rho_i)}{\rho_i}L_i) \leq \frac{L_{\max}}{r} + \frac{L_i}{\rho_i}.\end{aligned} \tag{19}$$

Therefore, the latency of RP-SCFQ is same as that of WFQ and this result is sufficient to argue that RP-SCFQ has the same delay bound as WFQ.

On the other hand, the end-to-end delay bound of session i can be given as follows [8], [9]:

$$D_i \le \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{n} \Theta_i^{S_j} - \frac{L_i}{\rho_i}, \tag{20}$$

where $\Theta_i^{S_j}$ is the latency of connection i in the j-th server in the network. Finally, we can get the end-to-end delay bound of RP-SCFQ by substituting (19) into (20).

$$D_i \le \frac{\sigma_i}{\rho_i} + (n-1)\frac{L_i}{\rho_i} + n\frac{L_{\max}}{r}, \tag{21}$$

where n is the number of nodes cascaded to build up an end-to-end route. (21) is actually common for all RPS class algorithms including RP-SCFQ.

3. Implementation Complexity

The most advantageous feature of SCFQ algorithm is its simplicity in maintaining the system potential, since it utilizes the timestamp of the packet being serviced instead of the approximation of GPS. RP-SCFQ also shares the same system potential during the transmission of a packet while it increases virtually as time passes. So, the computation of system potential in RP-SCFQ requires the complexity of O(1) because it is updated only once at every packet departure. Similar to other GPS class algorithms, the complexity of O($logV$) is added for sorting of the timestamps of HOL packets to be transmitted next. RP-SCFQ has the additional complexity of O(log$V$) compared to SCFQ for the computation and sorting of the starting potentials at every packet departure. Therefore, RP-SCFQ can provide better delay performance with slightly higher complexity than SCFQ.

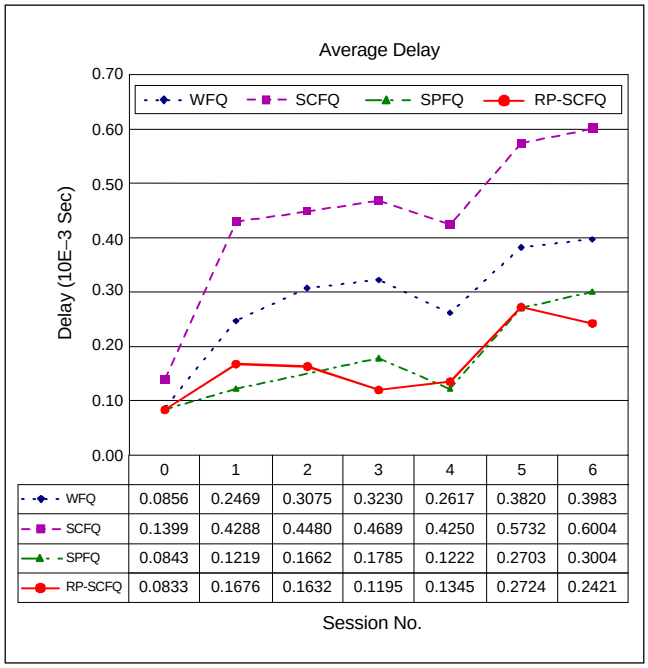
IV. SIMULATION STUDY

1. Simulation Scenarios and Input Conditions

In this section, we present the simulation results to verify the delay bound of RP-SCFQ and compare its average and maximum delay performances with those of WFQ, SCFQ and SPFQ. We used Block Oriented Network Simulator (BONeS) commercial network simulation tool [10]–[12]. The simulation model consists of 8 connections sharing the same outgoing link, and Table 1 shows the rate allocated to each connection. While the allocated rate is the contracted rate, the actual rate is the actual packet generation rate in the simulation. There is a misbehaving connection that sends more packets than the allocated rate to verify the session isolation property of each algorithm. We exclude misbehaving connection from the statistics deliberately. We adjust the rate of the outgoing link (or equivalently server rate) as 10 Mbps and use the fixed packet length of 424 bits per packet.

Table 1. Rate allocation in the simulation.

Connection No.	Allocated rate	Actual rate
0	0.5	0.48
1~4	0.078125	0.076
5~6	0.0625	0.062
7	0.0625	0.1



	0	1	2	3	4	5	6
WFQ	0.0856	0.2469	0.3075	0.3230	0.2617	0.3820	0.3983
SCFQ	0.1399	0.4288	0.4480	0.4689	0.4250	0.5732	0.6004
SPFQ	0.0843	0.1219	0.1662	0.1785	0.1222	0.2703	0.3004
RP-SCFQ	0.0833	0.1676	0.1632	0.1195	0.1345	0.2724	0.2421

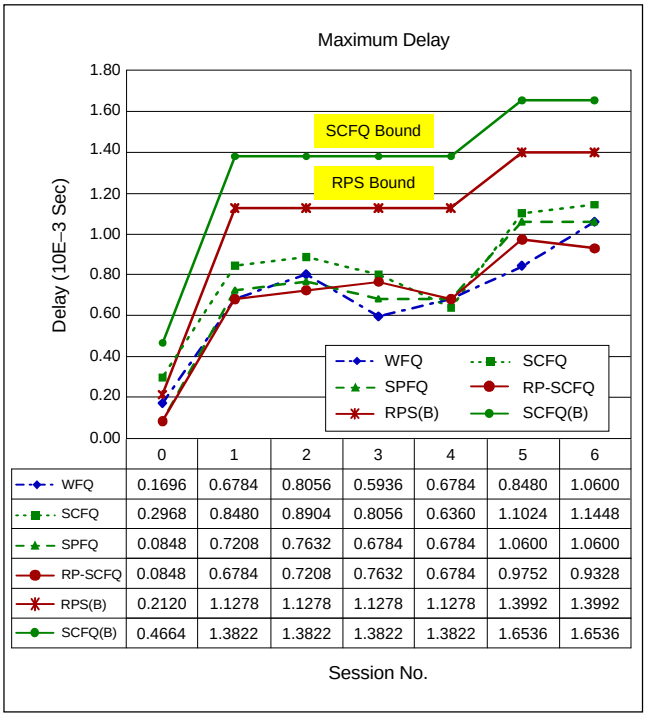
Fig. 3. Average delay.

We use an on/off traffic source to generate the bursty traffic and the generated traffic is shaped through the leaky bucket with the bucket size $\sigma_i = 2$ for all i. On and off duration follow geometric distributions.

2. Simulation Results and Discussion

A. Delay performance

In Fig. 3, the average delay of well-behaving connections is compared with other algorithms. RP-SCFQ provides similar average delay with SPFQ, while it has lower average delay than that of WFQ and SCFQ. Note that RP-SCFQ can provide



	0	1	2	3	4	5	6
WFQ	0.1696	0.6784	0.8056	0.5936	0.6784	0.8480	1.0600
SCFQ	0.2968	0.8480	0.8904	0.8056	0.6360	1.1024	1.1448
SPFQ	0.0848	0.7208	0.7632	0.6784	0.6784	1.0600	1.0600
RP-SCFQ	0.0848	0.6784	0.7208	0.7632	0.6784	0.9752	0.9328
RPS(B)	0.2120	1.1278	1.1278	1.1278	1.1278	1.3992	1.3992
SCFQ(B)	0.4664	1.3822	1.3822	1.3822	1.3822	1.6536	1.6536

Fig. 4. Maximum delay.

the same level of delay performance as that of SPFQ with simplified enqueue operation.

Figure 4 shows the comparison of the maximum delay among the algorithms. From **Theorem 2**, the analytical delay bound of RPS can be obtained by setting $n=1$ in (21). From the well-known latency of SCFQ [8], $\Theta_i \le (V-1)\frac{L_{\max}}{r}+\frac{L_i}{\rho_i}$, the delay bound of SCFQ can also be calculated by using (20) as $D_i \le \frac{\sigma_i}{\rho_i}+(V-1)\frac{L_{\max}}{r}$, where V is the number of connections.

In Fig. 4, the analytical delay bound is compared with the maximum delay gathered from other algorithms. It reveals that in case of SCFQ, the maximum delay of connection 1 exceeds the RPS bound.

B. Short-term fairness

In general the short-term fairness can be evaluated by comparing the differences of the potentials distributed to each session. To see the short-term fairness, we have to choose the period during which all connections are simultaneously backlogged. For convenience, we consider the case where there are three Poisson connections with the allocated rate of 0.5, 0.3, and 0.2, respectively. We also assume that each connection keeps its allocated rate.

Figure 5 shows the evolution of the session and system potential of RP-SCFQ for a server busy period. It can be seen

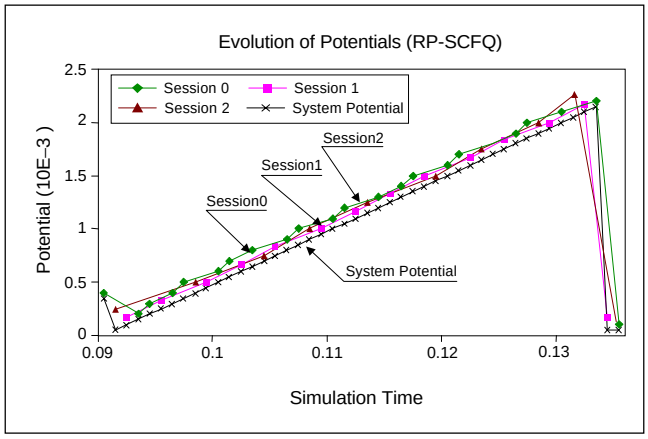


Fig. 5. Evolution of session potentials of RP-SCFQ.

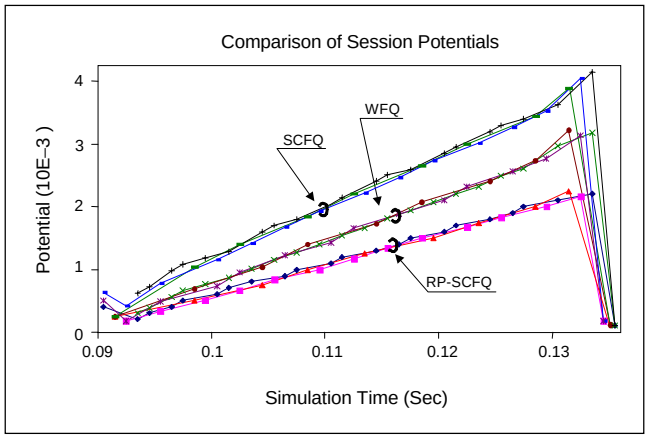


Fig. 6. Comparison of short-term fairness.

that the system potential is always maintained lower than the smallest session potential during a server busy period and this fact guarantees the RPS property of RP-SCFQ. In Fig. 5, the differences of the session potentials between any two sessions during a server busy period can be considered as the short-term fairness [3].

Figure 6 compares the evolution of the session potentials under three different algorithms. Intuitively, the short-term fairness of RP-SCFQ in the server busy period is not worse than that of other algorithms because the potential difference in RP-SCFQ doesn't exceed the others at any time.

Table 2 compares the average and maximum fair index of each algorithm. At every packet departure, the difference between maximum and minimum session potentials is computed per algorithm. We see that the short-term fairness of RP-SCFQ is excellent when it is compared with other algorithms.

V. CONCLUSION

We have proposed the new scheduling algorithm, RP-SCFQ,

Table 2. Difference of normalized service (fair index).

	Average	Maximum
WFQ	0.159	0.620
SCFQ	0.153	0.678
RP-SCFQ	0.124	0.438

whose system potential is proportional to the allocated rate so that the delay performance of SCFQ can be improved. We proved that RP-SCFQ satisfies the properties of RPS and showed that RP-SCFQ provides better delay performance than SCFQ and the maximum delay is maintained under the RPS bound via the simulation study.

We compared the short-term fairness among the algorithms for a server busy period. From the fact that the fairness is generally defined as the maximum difference between normalized service received by backlogged sessions, we showed that RP-SCFQ has better fairness performance. The mathematical analysis of the fairness is still open for further research.

RP-SCFQ is appropriate for high-speed packet-switched networks since its implementation complexity is low while it guarantees the bounded delay even in the worst-case input traffic conditions.

REFERENCES

[1] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *Proceedings of IEEE INFOCOM '92*, 1992, pp. 915–924.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Proceedings of SIGCOMM '89*, Austin, Texas, Sep. 1989, pp. 1–12.

[3] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proceedings of IEEE INFOCOM '94*, Apr. 1994, pp. 636–646.

[4] D.Stiliadis and A.Varma, "Rate-Proportional Servers: A design methodology for fair queueing algorithms," *Technical report*, UCSC-CRL-95-58, Dec. 1995.

[5] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching," *ACM Transactions on Computer Systems*, Vol. 9, No. 2, May 1991, pp. 101–124.

[6] D.Stiliadis and A.Varma, "Efficient fair-queueing algorithms for ATM and packet networks," *Technical Reports*, UCSC-CRL-95-59, Dec. 1995.

[7] H.Zhang, "Service disciplines for guaranteed performance service in packet switching networks," *Proceedings of the IEEE*, Vol. 83, No. 10, Oct. 1995, pp. 1374–1396.

[8] D.Stiliadis and A.Varma, "Latency-Rate Servers; A general model for analysis of traffic scheduling algorithms," *Technical Reports*, UCSC-CRL-95-38, July 1995.

[9] S.Keshav, "An engineering approach to computer networking," Addison Wesley, 1997.

[10] ALTA Group of Cadence Design System Inc., "BONeS designer user's guide for Release 3.6," Mar. 1996.

[11] ALTA Group of Cadence Design System Inc., "BONeS designer core library reference for Release 3.6," Mar. 1996.

[12] ALTA Group of Cadence Design System Inc., "BONeS designer application library reference for Release 3.6," Mar. 1996.

Byung-Hwan Choi received the B.S. degree from Kwang-Woon University, Seoul, Korea in 1989 and the M.S. degree from ICU, Taejon, Korea in 2000 all in Communications Engineering. From 1990 to 1997, he worked for Dacom Corp. and in 1997 he joined Hanaro Telecom Corp. where he is engaged in the planning project for deploying Internet services in Korea. His research interests are mostly related to development of new Internet services, and next generation Internet architecture.

Hong-Shik Park received the B.S. degree from Seoul National University, Seoul, Korea in 1977, and the M.S. and Ph.D. degrees from KAIST, Taejon, Korea all in Electrical Engineering in 1986 and 1995, respectively. In 1977 he joined Electronics and Telecommunications Research Institute and had been engaged in development of TDX digital switching system family including TDX-1, TDX-1A, TDX-1B, TDX-10, and ATM switching systems. In 1998 he moved to ICU, Taejon, Korea as a faculty member. Currently he is an associate professor. His research interests are network architecture, network protocols, and performance analysis of telecommunication systems. He is a member of the IEEK, KICS, Korea, and IEICE, Japan.