

# Low-latency guaranteed-rate scheduling using Elastic Round Robin

Salil S. Kanhere\*, Harish Sethu

Department of Electrical and Computer Engineering, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104, USA

Received 29 January 2001; revised 5 December 2001; accepted 5 December 2001

## Abstract

Packet scheduling algorithms in switches and routers will likely play a critical role in providing the Quality-of-Service (QoS) guarantees required by many real-time multimedia applications. Elastic Round Robin (ERR), a recently proposed fair scheduling discipline designed for best-effort traffic, is very efficient with an  $O(1)$  dequeuing complexity and, in addition, has better fairness characteristics than other algorithms of equivalent complexity. In this paper, we analyze ERR for guaranteed-rate services, and obtain an upper bound on its latency. We further show that the bound obtained in this paper is tight. Our analysis shows that ERR, in comparison to other scheduling disciplines of equivalent complexity, also has significantly better latency properties. The combination of fairness, efficiency and low-latency makes ERR an attractive scheduling discipline for both best-effort and guaranteed-rate services. © 2002 Elsevier Science B.V. All rights reserved.

*Keywords:* Elastic Round Robin; Fair queuing; Guaranteed-rate scheduling; Latency

## 1. Introduction

Future high-speed packet-switched networks are expected to support a variety of services beyond the best-effort service available in the Internet today. A number of new applications such as distance learning and multimedia teleconferencing rely on the ability of the network to guarantee such services. For example, such applications would expect the network to ensure that each flow of traffic receives its fair share of the bandwidth and is able to provide performance guarantees such as an upper bound on the end-to-end delay. This requires a Quality-of-Service (QoS) mechanism to efficiently apportion, allocate and manage limited resources among competing users. An important component of such a mechanism is the traffic scheduling algorithm typically used at the output links of switches and routers.

The function of a packet scheduler at an output link is to select the next packet for transmission from among the packets awaiting transmission through the output link. Some of the most important and desirable properties of a scheduling discipline are fairness, efficiency and low-latency, as described below.

- *Fairness.* The available link bandwidth must be distributed among the flows sharing the link in a fair manner.

This ensures that the performance achieved by a flow is not affected when a possibly misbehaving flow tries to transmit packets at a rate faster than its fair share. In this paper, we use the classic notion of fairness given by the max–min fair share policy [1].

- *Latency.* For guaranteed-rate services, the latency should be measured as the length of time it takes a new flow to begin receiving service at the guaranteed rate. This latency is directly related to the amount of playback buffering required at the receiver.
- *Efficiency.* In high-speed networks with large numbers of active flows, the time available for a scheduler to make its scheduling decision is very small. Hence, it is desirable that the time to enqueue a received packet or to dequeue a packet for transmission is as independent as possible of the number of flows sharing the output link. A per-packet work complexity of  $O(1)$  is most desirable.

Scheduling algorithms can be broadly classified into two categories—sorted-priority schedulers and frame-based schedulers. Sorted-priority schedulers maintain a global variable known as the virtual time or the system potential function. A sorted-priority scheduler then uses this variable to compute the timestamp for each packet indicating the relative priority of the packet for transmission over the output link. The packets are then scheduled in increasing order of their timestamps. Examples of sorted-priority schedulers are Weighted Fair Queuing (WFQ) [2,3], Self-Clocked Fair Queuing (SCFQ) [4], Start-Time Fair Queuing

\* Corresponding author.

E-mail address: salil@ece.drexel.edu (S.S. Kanhere).

(SFQ) [5], Frame-Based Fair Queuing<sup>1</sup> (FFQ) [6] and Worst-Case Fair Weighted Fair Queuing (WF<sup>2</sup>Q) [7]. The sorted-priority schedulers differ in the manner in which they calculate the global virtual time function. There are two major costs associated with the implementation of sorted-priority schedulers:

1. The complexity of computing the system virtual time: For WFQ, the worst-case complexity is  $O(n)$  where  $n$  is the number of flows sharing the same output link. However, in a number of schedulers such as SCFQ, SFQ and FFQ proposed in recent years, the complexity of computing the virtual time is  $O(1)$ .
2. The complexity of maintaining a sorted list of packets based on their timestamps, and the complexity of computing the maximum or the minimum in this list prior to each packet transmission. For  $n$  flows the work complexity of the scheduler prior to each packet transmission is  $O(\log n)$ .

In frame-based schedulers such as Deficit Round Robin (DRR) [8] and Elastic Round Robin (ERR) [9], the scheduler visits all the non-empty queues in a round robin order. During each service opportunity of a flow, the intent of such a scheduler is to provide to the flow an amount of service proportional to its fair share of the bandwidth. The frame-based schedulers do not maintain a global virtual time function and also do not require any sorting among the packets available for transmission. This reduces the implementation complexity of frame-based scheduling disciplines to  $O(1)$ , making them attractive for implementation in routers, and especially so, in hardware switches.

Elastic Round Robin (ERR) [9] is a recently proposed frame-based scheduling discipline for best-effort traffic, that achieves very good efficiency with a low per-packet work complexity of  $O(1)$  with respect to the number of flows. In addition, it has better fairness properties than other schedulers of equivalent work complexity such as DRR. In this paper, we show that ERR can also be easily adapted for scheduling guaranteed-rate connections, and that it belongs to the class of Latency-Rate ( $\mathcal{LR}$ ) Servers [10], with a latency bound significantly lower than those of other scheduling disciplines of comparable work complexity. These properties of ERR make it an attractive scheduling discipline for both best-effort and guaranteed-rate services.

Section 2 of this paper briefly describes the concept of *Latency-Rate ( $\mathcal{LR}$ ) Servers*, a general class of schedulers proposed by Stiliadis and Verma [10]. In Section 3, we present a weighted version of ERR for serving guaranteed-rate flows. In Section 4, we evaluate the latency

<sup>1</sup> Note that frame-based fair queuing, in spite of its name, is actually a sorted-priority scheduling discipline. The algorithm uses a framing approach similar to that used in frame-based schedulers to update the state of the system. However, as in sorted-priority schedulers, packets are transmitted based on their timestamps.

bound of ERR and prove that it belongs to the class of  $\mathcal{LR}$  servers. Section 5 compares the fairness, latency and work complexity of ERR with other guaranteed-rate scheduling disciplines. A tabulated summary of the properties is also provided. Finally, Section 6 concludes the paper.

## 2. $\mathcal{LR}$ servers and related background

In deriving an upper bound on the latency of ERR, we use the concept of  $\mathcal{LR}$  servers first proposed in Ref. [10]. We define a flow as *active* during an interval of time, if at all instants of time during this interval, it has at least one packet awaiting service or being served. We now define the notion of a *busy period*, an essential component of the concept of  $\mathcal{LR}$  servers.

**Definition 1.** A *busy period* of a flow is defined as the maximal time interval during which the flow is active if it is served at exactly its reserved rate.

Let  $Arrived_i(t_1, t_2)$  denote the total number of bits of flow  $i$  that arrive at the scheduler during the time interval  $(t_1, t_2)$ . Consider an interval of time  $(\tau_1, \tau_2)$  which represents a busy period for flow  $i$ . Then for any time interval  $(\tau_1, t)$  such that  $t \in (\tau_1, \tau_2)$ , the number of bits that arrive during this interval is greater than or equal to the number of bits that would exit the scheduler if the flow received service at its reserved rate,  $\rho_i$ . In other words, for all  $t \in (\tau_1, \tau_2)$ ,

$$Arrived_i(\tau_1, t) \geq \rho_i(t - \tau_1)$$

The definition of the busy period supposes that flow  $i$  is served at the constant reserved rate, and therefore, depends only on the reserved rate of the flow and the packet arrival pattern of the flow. An active period of a flow, however, reflects the actual behavior of the scheduler where the instantaneous service offered to flow  $i$  varies according to the number of active flows. If during a busy period of flow  $i$ , the instantaneous service rate offered to flow  $i$  is greater than the allocated rate, then the flow may cease to be active. Thus, a busy period of a flow may include multiple active periods for that flow. Note that the start of a busy period of a flow is always caused by the arrival of a packet belonging to the flow.

The following definitions lead to a formal notion of latency in the case of guaranteed-rate servers. The reader is referred to Ref. [10] for a more detailed discussion.

**Definition 2.** Define  $Sent_i(t_1, t_2)$  as the amount of service received by flow  $i$  during the interval  $(t_1, t_2)$ .

**Definition 3.** Let time instant  $\alpha_i$  represent the start of a certain busy period for flow  $i$ . Let  $t > \alpha_i$  be such that the flow is continuously busy during the time interval  $(\alpha_i, t)$ .

Define  $S_i(\alpha_i, t)$  as the number of bits belonging to packets in flow  $i$  that arrive after time  $\alpha_i$  and are scheduled during the time interval  $(\alpha_i, t)$ .

Note that  $Sent_i(\alpha_i, t)$  is not necessarily equal to  $S_i(\alpha_i, t)$ . This is because, during this interval of time, the scheduler may still be serving packets that arrived during a previous busy period.  $S_i(\alpha_i, t)$ , therefore, is not necessarily the same as the total number of bits scheduled from flow  $i$  in this interval. We are now prepared to present the definition of latency in  $\mathcal{LR}$  servers.

**Definition 4.** The latency of a flow is defined as the minimum non-negative constant  $\Theta_i$  that satisfies the following for all possible busy periods of the flow,

$$S_i(\alpha_i, t) \geq \max\{0, \rho_i(t - \alpha_i - \Theta_i)\} \quad (1)$$

As defined in Ref. [10], a scheduler which satisfies Eq. (1) for some non-negative constant value of  $\Theta_i$  is said to belong to the class of  $\mathcal{LR}$  servers.

The above definition captures the fact that the latency of a guaranteed-rate scheduler should not merely be the time it takes for the first packet of a flow to get scheduled, but should be a measure of the cumulative time that a flow has to wait until it begins receiving service at its guaranteed rate.

### 3. Weighted elastic round robin

ERR was originally proposed as a fair and efficient scheduling discipline for use in wormhole networks, popular in interconnection networks of parallel systems. The reader is referred to Ref. [9] for a detailed discussion of ERR for best-effort traffic. In this paper, we present a weighted version of ERR for guaranteed-rate services.

Consider an output link of transmission rate  $r$ , access to which is controlled by the ERR scheduler. Let  $n$  be the total number of flows and let  $\rho_i$  be the reserved rate for flow  $i$ . Let  $\rho_{\min}$  be the smallest of the reserved rates. Note that since all the flows share the same output link, a necessary constraint is that the sum of the reserved rates be no more than the transmission rate of the output link. In order that each flow receives service proportional to its guaranteed rate, the ERR scheduler assigns a weight to each flow. The weight assigned to flow  $i$ ,  $w_i$ , is given by,

$$w_i = \frac{\rho_i}{\rho_{\min}} \quad (2)$$

Note that for any flow  $i$ ,  $w_i \geq 1$ .

The ERR scheduler maintains a linked list of the active flows, called the *ActiveList*. When a packet arrives, and if it belongs to a new flow not in the *ActiveList*, the flow is added

to the tail of the *ActiveList*. The ERR scheduler always serves a packet from the flow at the head of the *ActiveList*.

An important and distinct feature of ERR is its definition of a round. We define a round as one round robin iteration consisting *only* of the service visits to all the flows included in the *ActiveList* at the start of the round. For example, a new flow that becomes active while a round is in progress is immediately added to the *ActiveList*, but is served only in the next round. Each flow receives no more than one service opportunity in each round. The scheduler selects the flow at the head of the *ActiveList* for service and calculates its *Allowance*, defined as the number of bits that the flow can transmit in the current service opportunity. In an ERR scheduler, this allowance is *elastic*, i.e. a flow may exceed its allowance during a round. Let  $A_i(s)$  represent the allowance for flow  $i$  during round  $s$ , and let  $Sent_i(s)$  be the number of bits actually scheduled from flow  $i$  during this round. The ERR scheduler will not begin transmission of the next packet in the queue for flow  $i$ , unless  $Sent_i(s)$  is less than  $A_i(s)$ . When  $Sent_i(s)$  becomes greater than  $A_i(s)$  during the transmission of a packet, the scheduler does not pre-empt the transmission but instead, waits for the end of the transmission and then begins the service opportunity for the next flow in the *ActiveList*. At this point, if flow  $i$  is still active, it is added back at the tail end of the list.

The number of excess bits of a flow sent in addition to its allowance during a service opportunity is recorded in the *surplus count*. Let  $SC_i(s)$  represent the surplus count of flow  $i$  in round  $s$ . Following the service of flow  $i$  during the  $s$ th round, its surplus count is calculated as:

$$SC_i(s) = Sent_i(s) - A_i(s) \quad (3)$$

$MaxSC(s)$  is defined as the largest surplus count among all the flows served in round  $s$ . The allowance for each flow is calculated using the  $MaxSC$  value in the previous round, as follows:

$$A_i(s) = w_i(1 + MaxSC(s - 1)) - SC_i(s - 1) \quad (4)$$

Note that the allowance of each flow during the next round depends on the amount by which other flows exceed their allowances in the current round. Each flow seeks to catch up with other flows by correspondingly increasing its allowance for the next round. The allowances, however, are bounded as follows. Let  $m$  be the size in bits of the largest packet that is *actually* served during the execution of a scheduling algorithm. It is readily observed that since no new packet of a flow is scheduled for transmission once the flow has exceeded its allowance, the surplus count is always less than the size of the last packet served. More generally, for any flow  $i$  and round  $s$ ,

$$0 \leq SC_i(s) \leq m - 1 \quad (5)$$

$$0 \leq MaxSC(s) \leq m - 1 \quad (6)$$

Note that, as opposed to DRR [8], ERR does not use a pre-determined *quantum* as the service each flow is

expected to receive during a service opportunity. This gives ERR a couple of important advantages over DRR as well as certain other schedulers such as Surplus Round Robin (SRR) [11–13] which are based on the principle of assigning a quantum to each flow. Firstly, the fairness and the latency bounds of ERR depend only on the sizes of the packets that actually arrive in the flow rather than on the size of the largest packet that may potentially arrive at the scheduler. This significantly improves the latency and fairness bounds since, in most networks including the Internet, the average packet size is much smaller than the maximum possible packet size [14,15]. Secondly, unlike DRR and SRR, ERR does not require knowledge of the upper bound on the transmission time of a packet to achieve a work complexity of  $O(1)$ , rendering it easier to adapt to networks in which the scheduler cannot presume knowledge of the size of a packet prior to its scheduling action. This is important in wormhole networks where the transmission time of a packet depends not just on the length of the packet but also on the congestion in the network. This is also important in other contexts such as in an ATM network transmitting IP packets over AAL5, where the end of the packet is not known until the last ATM cell of the packet arrives.

#### 4. Latency bound of ERR

The analysis of the latency of ERR is facilitated by the following result, stated below as Lemma 1 and proved in Ref. [10]. This result allows one to obtain a bound on the latency achieved by a flow as given by Definition 3 by considering only the active periods of a flow.

**Lemma 1.** *Let  $\tau_i$  be an instant of time when flow  $i$  becomes active. Let  $t > \tau_i$  be some instant of time such that the flow is continuously active during the time interval  $(\tau_i, t)$ . Let  $\Theta'_i$  be the smallest non-negative number such that the following is satisfied for all  $t$ .*

$$Sent_i(\tau_i, t) \geq \max\{0, \rho_i(t - \tau_i - \Theta'_i)\} \quad (7)$$

*Even though  $(\tau_i, t)$  may not be a continuously busy period for flow  $i$ , the latency as defined by Eq. (1) is bounded by  $\Theta'_i$ .*

We now proceed to derive a bound on the latency of the ERR scheduler, using the lemma above. Note that the instant of time when a flow  $i$  becomes active,  $\tau_i$ , may or may not coincide with the start of the round robin service opportunity of some other flow. In the following, we prove that a tight upper bound on the latency of the ERR scheduler can be obtained by considering  $\tau_i$  belonging to only a subset of all possible time instants. This subset is the set of all time instants that coincide with the beginning or the end of the service opportunity of flows.

**Definition 5.** Define  $\mathbf{T}$  as the set of all time instants, during an execution of the ERR algorithm, at which the scheduler ends serving one flow and begins serving another. Define  $\mathbf{T}_i$  as the set of all time instants at which the scheduler begins serving flow  $i$ . Note that the set  $\mathbf{T}$  is the union of  $\mathbf{T}_i$  for all flows  $i$ , that are served during the execution of the scheduler.

**Lemma 2.** *The latency experienced by flow  $i$  in an ERR scheduler will reach its upper bound,  $\Theta'_i$ , only if the time instant,  $\tau_i$ , at which flow  $i$  becomes active, belongs to the set  $\mathbf{T}$ .*

**Proof.** Assume that flow  $i$  becomes active at time instant  $\tau_i$ . Let  $(t_1, t_2)$ ,  $t_1 \leq \tau_i < t_2$  be the time interval during which some flow  $j \neq i$  receives its round robin service opportunity. Consider the case when  $\tau_i$  does not coincide with the start of the service opportunity of flow  $i$ , i.e.  $\tau_i > t_1$ . Now, the time interval  $(t_1, \tau_i)$ , which is a part of the round robin service opportunity of flow  $j$ , will not contribute to the latency experienced by flow  $i$ . On the other hand, consider the case when  $\tau_i$  coincides with  $t_1$ , the start of the service opportunity of flow  $j$ . Now, the time for which flow  $i$  has to wait before receiving any service will include the entire time interval  $(t_1, t_2)$  during which flow  $j$  receives its round robin service opportunity. Clearly, the latency experienced by flow  $i$  is always greater when  $\tau_i$  coincides with the start of the service opportunity of some other flow. The statement of the lemma follows from this observation.

□

From Lemma 2, in deriving an upper bound on the latency experienced by flow  $i$ , therefore, one needs to only consider  $\tau_i$  such that  $\tau_i \in \mathbf{T}$ . The following lemma further limits the cases we need to consider in deriving the upper bound. Note that, in proving the upper bound, we need to only consider the intervals of time over which Eq. (7) is an equality. In fact, the upper bound on the latency is achieved at time instants  $t$  when  $Sent_i(\tau_i, t) = \rho_i(t - \tau_i - \Theta'_i)$ . The following lemma provides a simple condition on  $t$  in order to achieve the equality in Eq. (7).

**Lemma 3.** *If flow  $i$  becomes active at time instant  $\tau_i$ , then there exists some  $t \in \mathbf{T}_i$  such that the flow remains active during the interval  $(\tau_i, t)$ , and*

$$Sent_i(\tau_i, t) = \rho_i(t - \tau_i - \Theta'_i)$$

**Proof.** Note that, if  $Sent_i(\tau_i, t) = \rho_i(t - \tau_i - \Theta'_i)$ , then the flow experiences the worst-case latency at time instant  $t$ . Consider any two consecutive time instants  $t_1$  and  $t_2$  which both belong to  $\mathbf{T}_i$ . Consider an instant of time  $t$ ,  $t_1 < t < t_2$ .

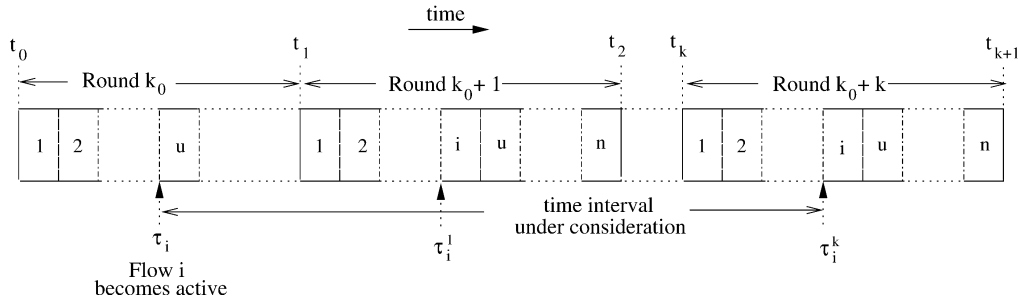


Fig. 1. An illustration of the time interval under consideration.

*Case 1:* Let  $t$  be such that flow  $i$  is receiving service at time instant  $t$ .

During the interval  $(t_1, t)$ , the amount of service received by the flow is  $r(t - t_1)$ , where  $r$  is the rate of the link. Clearly, during this time, the flow is receiving service at the guaranteed rate or higher. Therefore, the worst-case latency experienced by the flow until any time in the interval  $(t_1, t)$  is no worse than that experienced by it until time  $t_1 \in \mathbf{T}_i$ .

*Case 2:* Let  $t$  be such that some flow other than  $i$  is receiving service at time instant  $t$ .

During the interval  $(t, t_2)$ , flow  $i$  receives no service at all. Therefore, the latency experienced by the flow increases after time  $t$  but only until time  $t_2$ . Thus, the worst-case latency experienced by the flow until any time in the interval  $(t, t_2)$  is no worse than that experienced by it until time  $t_2 \in \mathbf{T}_i$ .

The above two cases illustrate that the worst-case latency experienced by a flow during an interval  $(t_1, t_2)$  is equal to the latency experienced by the flow until either time  $t_1$  or time  $t_2$ . Extrapolating to all intervals between consecutive time instants that belong to  $\mathbf{T}_i$ , the statement of the lemma is proved.

□

**Theorem 1.** *The ERR scheduler belongs to the class of  $\mathcal{LR}$  servers, with an upper bound on the latency  $\Theta_i$  for flow  $i$  given by,*

$$\Theta_i \leq \frac{(W - w_i)m + (n - 1)(m - 1)}{r} \quad (8)$$

where  $n$  is the total number of active flows and  $W$  is the sum of the weights of all the flows.

**Proof.** From Lemma 1, we know that the latency of the ERR scheduler as defined by Eq. (1) is bounded by  $\Theta'_i$ . Hence we will prove the theorem by showing that  $\Theta'_i = ((W - w_i)m + (n - 1)(m - 1))/r$ . By the statement of Lemma 2, this upper bound on the latency is reached only if the time instant at which flow  $i$  becomes active,  $\tau_i$ , belongs to  $\mathbf{T}$ . Therefore, in seeking the upper bound on

the latency, we may assume that flow  $i$  becomes active at exactly the instant that some other flow begins receiving service.

Let  $\tau_i^k$  be the time instant marking the start of the  $k$ th service opportunity of flow  $i$  after it becomes active at time instant  $\tau_i$ . Note that  $\tau_i^k$  belongs to  $\mathbf{T}_i$ . Therefore, from Lemma 3, in order to determine the latency bound as defined by Eq. (1), one needs to only consider intervals of time  $(\tau_i, \tau_i^k)$  for all  $k$ . Fig. 1 shows the time interval under consideration for a given  $k$ .

To prove the statement of the theorem, we first obtain the lower bound on the total service received by flow  $i$  during the time interval under consideration. Then we express the lower bound in the form of Eq. (7) to derive the latency bound.

Note that the time instant  $\tau_i$  may or may not coincide with the end of a round and the start of the subsequent round. Let  $k_0$  be the round which is in progress at time instant  $\tau_i$  or which ends exactly at time instant  $\tau_i$ . Let the time instant  $t_h$  mark the end of round  $(k_0 + h - 1)$  and the start of the subsequent round. For any flow  $j$  during a round  $s$  in the interval under consideration, using Eqs. (3) and (4), we have,

$$Sent_j(s) = w_j(1 + MaxSC(s - 1)) + SC_j(s) - SC_j(s - 1) \quad (9)$$

As shown in Fig. 1, assume that the time instant when flow  $i$  becomes active coincides with the time instant when some flow  $j$  is about to start its service opportunity during the  $k_0$ th round. Let  $G_a$  denote the set of flows which receive service during the time interval  $(\tau_i, t_1)$ , i.e. after flow  $i$  becomes active and during round  $k_0$ . Similarly, let  $G_b$  denote the set of flows which are served by the ERR scheduler during the time interval  $(t_0, \tau_i)$ , i.e. before flow  $i$  becomes active and during round  $k_0$ . Note that flow  $i$  is not included in either of these two sets since flow  $i$  will receive its first service opportunity only in the  $(k_0 + 1)$ th round. If the time instant  $\tau_i$  coincides with the end of a round, then the set  $G_a$  will be empty and all the  $n - 1$  flows will belong to the set  $G_b$ .

Consider the time interval  $(\tau_i, \tau_i^k)$  during which flow  $i$  receives  $k$  round robin service opportunities. This time interval can be split into three sub-intervals:

- (1)  $(\tau_i, t_1)$ : This sub-interval includes the part of the  $k_0$ th

round during which all the flows belonging to the set  $G_a$  will be served by the ERR scheduler. Summing Eq. (9) over all these flows,

$$t_1 - \tau_i \leq \frac{1}{r} \sum_{j \in G_a} \{w_j(1 + \text{MaxSC}(k_0 - 1)) + \text{SC}_j(k_0) - \text{SC}_j(k_0 - 1)\} \quad (10)$$

(2)  $(t_1, t_k)$ : This sub-interval includes  $k - 1$  rounds of execution of the ERR scheduler starting at round  $(k_0 + 1)$ . Consider the time interval  $(t_h, t_{h+1})$  when round  $(k_0 + h)$  is in progress. Summing Eq. (9) over all  $n$  flows,

$$t_{h+1} - t_h \leq \frac{W}{r}(1 + \text{MaxSC}(k_0 + h - 1)) + \frac{1}{r} \sum_{j=1}^n \text{SC}_j(k_0 + h) - \frac{1}{r} \sum_{j=1}^n \text{SC}_j(k_0 + h - 1)$$

Summing the above over  $k - 1$  rounds beginning with round  $k_0 + 1$ ,

$$t_k - t_1 \leq \frac{W}{r}(k - 1) + \frac{1}{r} \sum_{j=1}^n (\text{SC}_j(k_0 + k - 1) - \text{SC}_j(k_0)) + \frac{W}{r} \sum_{h=1}^{k-1} \text{MaxSC}(k_0 + h - 1) \quad (11)$$

(3)  $(t_k, \tau_i^k)$ : This sub-interval includes the part of the  $(k_0 + k)$ th round during which all the flows belonging to the set  $G_b$  will be served by the ERR scheduler. Summing Eq. (9) over all these flows,

$$\tau_i^k - t_k \leq \frac{1}{r} \sum_{j \in G_b} w_j(1 + \text{MaxSC}(k_0 + k - 1)) + \frac{1}{r} \sum_{j \in G_b} \text{SC}_j(k_0 + k) - \frac{1}{r} \sum_{j \in G_b} \text{SC}_j(k_0 + k - 1) \quad (12)$$

Combining Eqs. (10)–(12) and using Eq. (5) we have,

$$\begin{aligned} \tau_i^k - \tau_i &\leq \frac{W}{r}(k - 1) + \frac{W - w_i}{r} \\ &+ \frac{1}{r} \sum_{j \in G_a} w_j \text{MaxSC}(k_0 - 1) \\ &+ \frac{W}{r} \sum_{h=1}^{k-1} \text{MaxSC}(k_0 + h - 1) \\ &+ \frac{1}{r} \sum_{j \in G_b} w_j \text{MaxSC}(k_0 + k - 1) \\ &+ \frac{(n - 1)(m - 1)}{r} + \frac{1}{r} \text{SC}_i(k_0 + k - 1) \end{aligned}$$

Solving for  $(k - 1)$ ,

$$\begin{aligned} (k - 1) &\geq (\tau_i^k - \tau_i) \frac{r}{W} - \frac{W - w_i}{W} \\ &- \frac{1}{W} \sum_{j \in G_a} w_j \text{MaxSC}(k_0 - 1) \\ &- \sum_{h=1}^{k-1} \text{MaxSC}(k_0 + h - 1) \\ &- \frac{1}{W} \sum_{j \in G_b} w_j (\text{MaxSC}(k_0 + k - 1)) \\ &- \frac{(n - 1)}{W} (m - 1) - \frac{1}{W} \text{SC}_i(k_0 + k - 1) \end{aligned} \quad (13)$$

Using Eq. (9) over these  $(k - 1)$  rounds of service for flow  $i$ , and since the surplus count of a newly active flow is 0, we get,

$$\begin{aligned} \text{Sent}_i(\tau_i, \tau_i^k) &= \text{SC}_i(k_0 + k - 1) + w_i(k - 1) \\ &+ w_i \sum_{h=1}^k \text{MaxSC}(k_0 + h - 1) \end{aligned} \quad (14)$$

Now, since the reserved rates are proportional to the weights assigned to the flows as given by Eq. (2), and since the sum of the reserved rates is no more than the link rate  $r$ , we have,

$$\rho_i \leq \frac{w_i}{W} r \quad (15)$$

Using the above and using Eq. (13) to substitute for  $(k - 1)$  in Eq. (14), we get,

$$\begin{aligned} \text{Sent}_i(\tau_i, \tau_i^k) &\geq \rho_i(\tau_i^k - \tau_i) - \frac{w_i}{W}(W - w_i) \\ &- \frac{w_i}{W} \sum_{j \in G_a} w_j \text{MaxSC}(k_0 - 1) \\ &- \frac{w_i}{W} \sum_{j \in G_b} w_j \text{MaxSC}(k_0 + k - 1) \\ &- \frac{w_i}{W} (n - 1)(m - 1) - \text{SC}_i(k_0 + k - 1) \left( \frac{w_i}{W} - 1 \right) \end{aligned}$$

Using Eq. (6) and since  $W$  is the sum of the weights of all the  $n$  flows, we have,

$$\begin{aligned} \text{Sent}_i(\tau_i, \tau_i^k) &\geq \rho_i(\tau_i^k - \tau_i) - \frac{w_i}{W}(W - w_i) \\ &- \frac{w_i}{W}(W - w_i)(m - 1) - \frac{w_i}{W}(n - 1)(m - 1) \\ &- \text{SC}_i(k_0 + k - 1) \left( \frac{w_i}{W} - 1 \right) \end{aligned}$$

Table 1  
A comparison between scheduling disciplines

Scheduler	Complexity	Fairness	Latency bound for flow $i$
GPS [3]		0	0
Weighted Fair Queuing [2]	$O(n)$	$O(n)$	$\frac{m}{r} + \frac{m}{\rho_i}$
Self-Clocked Fair Queuing [4]	$O(n)$	$2m$	$\frac{(n-1)m}{r} + \frac{m}{\rho_i}$
Virtual Clock [16]	$O(n)$	$\infty$	$\frac{m}{r} + \frac{m}{\rho_i}$
Frame-based Fair Queuing [6]	$O(n)$	$2M + m$	$\frac{m}{r} + \frac{m}{\rho_i}$
Deficit Round Robin [8]	$O(1)$	$M + 2m$	$\frac{(W - w_i)M + (n-1)(m-1)}{r} + \left(\frac{1}{\rho_i} - \frac{1}{r}\right)(m-1)$
Elastic Round Robin [9]	$O(1)$	$3m$	$\frac{(W - w_i)m + (n-1)(m-1)}{r}$

Using Eq. (15), we have,

$$\begin{aligned} Sent_i(\tau_i, \tau_i^k) &\geq \rho_i(l_i^k - \tau_i) \\ &- \frac{\rho_i}{r}(W - w_i)(m) - \frac{\rho_i}{r}(n-1)(m-1) \\ &- SC_i(k_0 + k - 1)\left(\frac{w_i}{W} - 1\right) \end{aligned}$$

Simplifying further, and noting that the latency bound reaches the upper bound when  $SC_i(k_0 + k - 1)$  equals 0, we get,

$$\begin{aligned} Sent_i(\tau_i, \tau_i^k) &\geq \max\left\{0, \rho_i\left(\tau_i^k - \tau_i - \frac{(W - w_i)m + (n-1)(m-1)}{r}\right)\right\} \end{aligned} \quad (16)$$

As discussed earlier, based on Lemmas 2 and 3, flow  $i$  will experience its worst latency during an interval  $(\tau_i, \tau_i^k)$  for some  $k$ . Therefore, from Eq. (16) and Lemma 1, the statement of the theorem is proved.

□

We now proceed to show that the latency bound given by Theorem 1 is tight by illustrating a case where the bound is actually met. Assume that a flow  $i$  becomes active at time instant  $\tau_i$ , which also coincides with the end of a certain round  $k_0$  and the start of the round  $(k_0 + 1)$ . Since other flows in the *ActiveList* will be served first, flow  $i$  becomes backlogged instantly. Assume that for any time instant  $t$ ,  $t \geq \tau_i$ , a total of  $n$  flows, including flow  $i$ , are active. Also, assume that the summation of the reserved rates of all the  $n$  flows equals the output link transmission rate,  $r$ . Hence,  $\rho_i = (w_i/W)r$ . Since flow  $i$  became active at time  $\tau_i$ , its surplus count at the start of round  $(k_0 + 1)$  is 0. Let the surplus count of all the other flows at the start of round  $(k_0 + 1)$  be equal to 0. Assume that, a flow  $p$  which is not active after time  $\alpha_i$  and hence is not included in the  $n$  flows, was active during the  $k_0$ th round. Also assume that flow  $p$

exceeded its allowance by  $(m - 1)$  during its service opportunity in round  $k_0$ , leading to a value of  $MaxSC(k_0)$  equal to  $(m - 1)$ . From Eqs. (5), (6) and (9), any given flow  $j$  can transmit a maximum of  $w_j m + (m - 1)$  bits during a round robin service opportunity. In the worst case, before flow  $i$  is served by the ERR scheduler, each of the other  $(n - 1)$  flows will receive this maximum service. Hence, the cumulative delay until flow  $i$  receives service is given by,

$$\begin{aligned} D &= \frac{\left(\sum_{j \neq i} w_j\right)m + (n-1)(m-1)}{r} \\ &= \frac{(W - w_i)m + (n-1)(m-1)}{r} \end{aligned}$$

Noting that  $S_i(\alpha_i, \alpha_i + D)$  equals zero, it is readily verified that the bound is exactly met at time  $t = \alpha_i + D$ .

## 5. Comparison to other schedulers

Table 1 summarizes the work complexity, fairness and latency bounds of several guaranteed-rate scheduling disciplines that belong to the class of  $\mathcal{LR}$  servers. We measure fairness using a well-known and widely used metric, known as the Relative Fairness Bound (RFB) [4]. The RFB is defined as the maximum difference in the normalized service received by any two active flows over all possible intervals of time. In this table,  $M$  is the size of the largest packet that may potentially arrive during the execution of a scheduling algorithm. Recall that  $m$  is the size of the largest packet that *actually* arrives during the execution of the scheduler. Typically,  $M \gg m$ , since in most networks including the Internet, the vast majority of the packets are of much smaller size than the maximum possible size of a packet [14,15]. The properties of all the scheduling disciplines in Table 1 except ERR and DRR are derived in Ref. [17]. Note that the latency bound of DRR stated here is tighter than the

one derived in Ref. [17]. The derivation of this bound is similar to the proof of Theorem 1.

The GPS scheduler visits each active flow in a round-robin fashion, and serves an infinitesimally small amount of data proportional to the reserved rate of the flow [3]. Using this fluid model, the GPS scheduler is able to ensure that over any interval of time however small, the normalized difference between the service received by any two active flows is exactly zero. The RFB of GPS, therefore, is zero. The latency of GPS is also zero since a newly active flow begins receiving service instantaneously at the guaranteed rate. Recall that GPS is an ideal but not an implementable scheduler.

WFQ [2], FFQ [6] and Virtual Clock [16] have a low value of the latency bound. However, the work complexity of these schedulers is greater than those of ERR and DRR.

Virtual Clock has an RFB of infinity and therefore, cannot be considered to be a fair scheduler. The RFB of the WFQ scheduler is  $O(n)$  and thus, has better fairness (the exact expression for the RFB may be found in Ref. [17]). FFQ, on the other hand, has a still lower value of the RFB and thus is significantly more fair. However, FFQ requires periodic re-calibration of the virtual time, and also has a work complexity of  $O(n)$  rendering it less efficient than ERR or DRR. In fact, only ERR and DRR are efficient with an  $O(1)$  work complexity. Table 1 shows that ERR has better fairness properties and a lower latency bound than DRR does, especially considering that  $M$  is typically much greater than  $m$ .

## 6. Conclusion

Elastic Round Robin (ERR), a recently proposed fair, efficient and easily implementable scheduling discipline was designed to satisfy the unique needs of wormhole switching, popular in interconnection networks of parallel systems. It may also be readily adapted for fair scheduling of best-effort traffic in the Internet. In this paper, we present a weighted version of ERR for serving guaranteed-rate flows. We prove that ERR belongs to the class of  $\mathcal{LR}$  servers and evaluate the upper bound on the latency experienced by a flow served by an ERR scheduler. We also show that the bound obtained in this paper is tight.

Our analysis reveals that ERR has better fairness characteristics and a significantly better latency bound in comparison to other scheduling disciplines of equivalent complexity such as DRR. While fairness is an intuitively desirable goal, its practical relevance is in the bound on the latency that fair schedulers are able to provide. This latency, as defined for  $\mathcal{LR}$  servers in Ref. [10], has a direct bearing on the size of the playback buffers needed at the receivers for real-time communications. This paper

shows that, with its low-latency bound, ERR is an attractive scheduling discipline for both best-effort and guaranteed-rate traffic.

## Acknowledgements

This work was supported in part by NSF CAREER grant CCR-9984161 and US Air Force Contract F30602-00-2-0501.

## References

- [1] S. Keshav, An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network, Addison-Wesley, Reading, MA, 1997 Ch 9: Scheduling, pp. 209–261.
- [2] A. Demers, S. Keshav, S. Shenker, Design and analysis of a fair queuing algorithm, Proceedings of ACM SIGCOMM, Austin, 1989, pp. 1–12.
- [3] A.K. Parekh, R.G. Gallager, A generalized processor sharing approach to flow control the single node case, Proceedings of IEEE INFOCOM, Florence, Italy, 1992, pp. 915–924.
- [4] S.J. Golestani, A self-clocked fair queuing scheme for broadband applications, Proceedings of IEEE INFOCOM, Toronto, Canada, 1994, pp. 636–646.
- [5] V.P. Goyal, H.M. Vin, H. Cheng, Start-time fair queuing: a scheduling algorithm for integrated services packet switched networks, IEEE Transactions of Networking 5 (5) (1997) 690–704.
- [6] D. Stiliadis, A. Verma, Efficient fair queuing algorithms for packet-switched networks, IEEE Transactions on Networking 6 (2) (1998) 175–185.
- [7] J.C.R. Bennett, H. Zhang, WF2Q: Worst-case fair weighted fair queuing, Proceedings of IEEE INFOCOM, San Francisco, CA, 1996, pp. 120–128.
- [8] M. Shreedhar, G. Varghese, Efficient fair queuing using deficit round-robin, IEEE Transactions on Networking 4 (3) (1996) 375–385.
- [9] S.S. Kanhere, A. Parekh, H. Sethu, Fair and efficient packet scheduling in wormhole net works, Proceedings of the International Parallel and Distributed Processing Symposium, Cancun, Mexico, 2000, pp. 623–631.
- [10] D. Stiliadis, A. Verma, Latency-rate servers: a general model for analysis of traffic scheduling algorithms, IEEE Transactions on Networking 6 (5) (1998) 611–624.
- [11] S. Floyd, Notes on class-based-queuing and guaranteed service, <http://www.aciri.org/floyd/cbq.html>.
- [12] S. Floyd, V. Jacobson, Link-sharing and resource management models for packet networks, IEEE Transactions on Networking 3 (4) (1995) 365–386.
- [13] G.P.H. Adishesu, G. Varghese, A reliable and scalable striping protocol, Proceedings of ACM SIGCOMM, Palo Alto, CA, 1996, pp. 131–141.
- [14] K. Thompson, G. Miller, R. Wilder, Widearea internet traffic patterns and characteristic, IEEE Network (1997) 10–23.
- [15] I. Widjaja, A.I. Elwalid, Performance issues in vc-merge capable switches for multiprotocol label switching, IEEE Journal on Selected Areas in Communications 17 (6) (1999) 1178–1178.
- [16] L. Zhang, Virtual clock: a new traffic control algorithm for packet switching networks, Proceedings of ACM SIGCOMM, Philadelphia, PA, 1990, pp. 19–29.
- [17] D. Stiliadis, Traffic scheduling in packet-switched networks: analysis, design, and implementation, PhD thesis, University of California, Santa Cruz, June 1996.