A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms

Dimitrios Stiliadis
Bell Laboratories
Lucent Technologies
Holmdel, NJ 07733

Anujan Varma
Computer Engineering Department
University of California
Santa Cruz, CA 95064

Abstract

We introduce a general methodology for designing integrated shaping and scheduling algorithms for packet networks that provide fairness, low end-to-end delay, and low burstiness. The methodology is based on integrating a shaping mechanism with a scheduler from the class of *Rate-Proportional Servers* (RPS) defined in [10]. The resulting algorithms provide an end-to-end delay bound identical to that of Weighted Fair Queueing. Their worst-case fairness, in terms of minimizing the worst-case delay to empty the session backlog, is much superior to that of Weighted Fair Queueing, and equal to the best known for any scheduling algorithm. In addition, the algorithms achieve a level of fairness in the distribution of free bandwidth among competing sessions better than that of Weighted Fair Queueing. We show that, under this framework, even an unfair scheduling algorithm belonging to the RPS class, such as VirtualClock, can yield worst-case fairness identical to that obtained with Weighted Fair Queueing. We also develop an integrated shaper-scheduler that provides optimal output burstiness and is attractive for use in both network adapters and in switches that support traffic re-shaping. We describe an efficient implementation of this integrated shaping and scheduling algorithm with $\log_2(V)$ complexity, where V is the number of sessions sharing the outgoing link.

I. Introduction

Traffic scheduling algorithms are a necessary part of future integrated-services networks that will provide a broad range of Quality-of-Service (QoS) guarantees. Scheduling algorithms are essential for the support of delay-sensitive applications in these networks. In addition, traffic scheduling algorithms are also a vital part of many feedback-based congestion control schemes [7]. Several scheduling algorithms with a wide range of performance characteristics have been proposed and analyzed in the literature [1], [2], [3], [5], [9], [10], [14].

A traffic scheduling algorithm must satisfy three essential attributes for use in a packet switch: low end-to-end delay, good fairness properties, and simplicity of implementation [13]. Among the three, the delay and implementation complexity are usually the most important criteria for the selection of an algorithm for use in a real system. A larger delay bound implies increased burstiness of the session at the output of the scheduler, thus increasing the amount of buffering needed in the switches to avoid packet losses. The fairness properties of the algorithm, on the other hand, affect only the short-term distribution of service offered to the connections sharing the link. The short-term fairness, however, is critical in some applications, for example, when the spacing between packets of a session arriving at the destination is used to estimate its bottleneck service rate in the network [7]. Similarly, fair distribution of excess bandwidth among the sessions is often desirable. Therefore, fairness may become the key property for choosing between two scheduling algorithms providing the same delay guarantee.

Several measures have been used in the literature for analysis of the fairness of scheduling algorithms. Golestani [5] defined the fairness parameter of a scheduling algorithm as the maximum difference between the normalized service received by two backlogged connections over an interval in which both are continuously backlogged. We will refer to this fairness measure as the *service fairness index* (SFI). Thus, if $W_i(t_1,t_2)$ and $W_j(t_1,t_2)$ denote the service received by sessions i and j, respectively, during an interval $(t_1,t_2]$ in which both are backlogged, and if ρ_i and ρ_j are the allocated rates of the respective sessions, then the service fairness index is given by

$$\text{SFI} = \left| \frac{W_i(t_1,t_2)}{\rho_i} - \frac{W_j(t_1,t_2)}{\rho_j} \right|. \qquad (1.1)$$

With this criterion, ideal fairness is obtained when the scheduler serves the connections with rates proportional to their reservations at each instant, thus distributing the free bandwidth left behind by idle sessions proportionally among the active ones. Note that, according to this definition, some amount of short-term unfairness between sessions is inevitable in any packet-level scheduler, since each packet must be serviced exclusively. Golestani showed that the service fairness index of a packet-level scheduler is lower-bounded by $(L_i/\rho_i + L_j/\rho_j)/2$, where L_i and L_j are the maximum packet sizes of sessions i and j, respectively [5], [6]. The unfairness can be maintained small by good design. For example, the Self-Clocked Fair Queueing (SCFQ) algorithm achieves a service fairness index of $(L_i/\rho_i + L_j/\rho_j)$, close to the theoretical bound [5].

A second measure of fairness, defined by Parkeh [8], is based on the worst-case delay for clearing the backlog of a session's queue. According to this criterion, a scheduler is called *worst-case packet fair*, if, for every session i that is continuously backlogged during the interval $(t_1,t_2]$,

$$W_i(t_1,t_2) \geq (t_2 - t_1)\rho_i - C_i, \qquad (1.2)$$

This research is supported by the NSF Young Investigator Award No. MIP-9257103.

0-8186-7780-5/97 $10.00 © 1997 IEEE

where $W_i(t_1, t_2)$ represents the service offered to session i during the interval $(t_1, t_2]$, and C_i is a constant. We will call the smallest value of C_i satisfying this inequality, over all such intervals $(t_1, t_2]$, as the *worst-case fairness index* (WFI). Bennett and Zhang [1] defined a fairness parameter based on the smallest value of C_i satisfying the inequality, and normalizing it to the allocated rate of the session.

The worst-case fairness index measures the deviation between the service received by a session in the packet-level scheduler and the service received by the same session in a corresponding fluid server, where the session is serviced at a rate of ρ_i at each instant. Minimizing this parameter improves the traffic mix at the output of the scheduler [1]. However, it should be noted that this parameter does not specify how the free bandwidth left over by idle sessions is distributed across the active ones. For example, we will show that even an algorithm that distributes the excess bandwidth unfairly among the sessions, such as VirtualClock, can achieve ideal fairness according to this criterion, when combined with our shaping mechanism. On the other hand, the worst-case fairness index of Self-Clocked Fair Queueing is worse than that of VirtualClock, although the bandwidth distribution of the former is substantially superior.

Based on this criterion of worst-case fairness, Bennett and Zhang introduced a scheduling algorithm, called Worst-case Fair Fair-Queueing (WF^2Q), that provides fairness superior to that of Weighted Fair Queueing (WFQ). WF^2Q improves the traffic mix at the output of the scheduler by reducing the discrepancy between the packet server and its fluid equivalent. However, the algorithm is still based on Weighted Fair Queueing, which is not easily implementable, owing to the need for simulation of the fluid model.

In this paper, we extend the results on Rate-Proportional Servers [10] by combining them with a shaping mechanism. The resulting algorithms retain the same delay bound of Weighted Fair Queueing, but their fairness behavior can be superior to that of WFQ, and comparable to that of any known scheduling algorithm. This design methodology, although simple, leads to a number of important contributions:

1. The methodology results in a class of fair queueing algorithms with delay bounds as good as that of WFQ, optimal worst-case fairness properties, and simple implementation. These algorithms have worst-case fairness identical to that of WF^2Q. For example, VirtualClock, an algorithm known to be unfair in its short-term bandwidth distribution, can provide the same worst-case fairness as WFQ when combined with the shaping mechanism.
2. Different algorithms in this new class may have different fairness according to the service fairness index. When the scheduler used is Starting Potential-based Fair Queueing (SPFQ) [12], however, we show that the resulting algorithm has service fairness index superior to that of WFQ, and comparable to that of Self-Clocked Fair Queueing (within a constant factor of three). The fairness of this algorithm is no longer dependent on the minimum bandwidth allocation, but depends only on the bandwidth allocations of the two sessions under observation.
3. The methodology introduced here is also useful in the design of integrated shaper-schedulers for use in network interface adapters, and in switches performing traffic re-shaping. We describe an implementation of such an integrated shaper-scheduler with a time-complexity of $O(\log_2 V)$, where V is the number of sessions sharing the link. This implementation combines the VirtualClock algorithm with our shaping mechanism. To our knowledge this is the first implementation of a shaper-scheduler with $O(\log_2 V)$ complexity and optimal output burstiness. Our design can also be used to implement a rate-controlled server in a switch, as described in [4], and be used with other schedulers such as Delay-EDD [3].

II. Preliminaries

A. Definitions and Notations

We assume a packet switch where a set of V connections share a common outgoing link. The terms *connection*, *flow*, and *session* will be used synonymously. We denote with ρ_i the rate allocated to connection i and by r the transmission rate of the outgoing link.

We assume that the servers are non-cut-through devices. Let $A_i(\tau, t)$ denote the arrivals from session i during the interval $(\tau, t]$ and $W_i(\tau, t)$ the amount of service received by session i during the same interval. In a packet-by-packet model, we assume that $A_i(\tau, t)$ increases only when the last bit of a packet is received by the server; likewise, $W_i(\tau, t)$ is increased only when the last bit of the packet in service leaves the server.

Definition 1: A **system busy period** is a maximal interval of time during which the server is never idle.
During a system busy period the server is always transmitting packets.

Definition 2: A **backlogged period for session i** is any period of time during which packets belonging to that session are continuously queued in the system.
Let $Q_i(t)$ represent the amount of session i traffic queued in the server at time t, that is,

$$Q_i(t) = A_i(0, t) - W_i(0, t).$$

A connection is backlogged at time t if $Q_i(t) > 0$.

Definition 3: A **session i busy period** is a maximal interval of time $(\tau_1, \tau_2]$ such that at any time $t \in (\tau_1, \tau_2]$, the accumulated arrivals of session i since the beginning of the interval do not fall below the total service received during the interval at a rate of exactly ρ_i. That is,

$$A_i(\tau_1, t) \geq \rho_i(t - \tau_1).$$

A session busy period is the maximal interval of time during which if the session were serviced at exactly the guaranteed rate, it would remain continuously backlogged.

In [13], we introduced a general model for traffic scheduling algorithms, called *Latency-Rate* ($\mathcal{LR}$) servers. Any server in this class is characterized by two parameters: *latency* Θ_i and *minimum allocated rate* ρ_i. The following worst-case bound on the behavior of a LR-server was shown

in [13] when the arrivals or session i are shaped by a leaky bucket with parameters (σ_i, ρ_i).

Theorem 1: The maximum delay D_i^K of session i after the Kth node in an arbitrary network of $\mathcal{LR}$-servers is bounded as

$$D_i^K \leq \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{K} \Theta_i^{(S_j)};$$

where $\Theta_i^{(S_j)}$ is the latency of the jth server on the path of the session.

For a detailed explanation of the above result, the interested reader is referred to [13].

B. Potential Functions

The GPS scheduler provides ideal fairness by offering the same normalized service to all backlogged connections at every instant of time. Thus, if we represent the total amount of service received by each session by a function, then these functions can be seen to grow at the same rate for each backlogged session. In [10], we introduced such a function to represent the state of each connection in a scheduler and called it *connection potential.* The potential of a connection is a non-decreasing function of time during a system-busy period. When connection i is backlogged, its potential increases exactly by the normalized service it received. That is, if $P_i(t)$ denotes the potential of connection i at time t, then, during any interval $(\tau, t]$ within a backlogged period for session i,

$$P_i(t) - P_i(\tau) = \frac{W_i(\tau, t)}{\rho_i}.$$

Note that the potentials of all connections can be initialized to zero at the beginning of a system-busy period, since all state information can be reset when the system becomes idle.

The above definition of connection potential did not specify how the potential of a connection is updated when it is idle, except that the potential is non-decreasing. This is accomplished by defining a *system potential* function that keeps track of the progress of the total work done by the scheduler. The system potential $P(t)$ is a non-decreasing function of time. When an idle session i becomes backlogged at time t, its potential $P_i(t)$ can be set to $P(t)$ to account for the service it missed. Schedulers use different functions to maintain the system potential, giving rise to widely different delay- and fairness-behaviors. In general, the system potential at time t can be defined as a non-decreasing function of the potentials of the individual connections before time t, and the real time t.

$$P(t) = \mathcal{F}(P_1(t-), P_2(t-), \ldots, P_V(t-), t). \tag{2.1}$$

C. Rate-Proportional Servers

The basic objective of a rate-proportional server is to *equalize* the potential of all backlogged connections at each instant. This is achieved in a fluid server as follows: At any instant t, the scheduler services only the subset of connections with the minimum potential, and each connection in this subset receives service in proportion to its reserved rate ρ_i. Thus, the scheduler can be seen to increase the potentials of the connections in this subset at the same rate. A packet server approximates this behavior by sorting the packets in the order of their *finishing potentials* under the fluid server, and transmitting them in increasing order of the finishing potentials. The finishing potential of the kth packet of a session i, arriving at the scheduler at time t, is computed as

$$F_i^k = \min(F_i^{k-1}, P(t)) + \frac{L_i^k}{\rho_i}, \tag{2.2}$$

where L_i^k is the length of the packet and ρ_i the allocated rate of session i. That is, the finishing potential is computed by adding the service time of the packet at the allocated rate to its *starting potential.* The latter, in turn, is the minimum of the finishing potential of the previous packet received from session i, and the current system potential.

At the time that a connection becomes backlogged, its potential is updated based on the system potential function that keeps track of the progress of the total work done by the scheduler. When an idle session i becomes backlogged at time t, its potential $P_i(t)$ is set as

$$P_i(t) = \max(P_i(t-), P(t)),$$

to account for the service it missed. Rate-Proportional Servers may use a wide range of functions to maintain the system potential, but the function chosen must satisfy two fundamental properties: First, during any interval $(t_1, t_2]$ within a system-busy period, the system potential function must be increased with a rate of at least one, that is,

$$P(t_2) - P(t_1) \geq (t_2 - t_1). \tag{2.3}$$

Second, the system potential function must never exceed the potential of any backlogged connection. These two conditions are sufficient to achieve a delay bound equal to that of Weighted Fair Queueing. In addition, if the difference between the system potential and the potential of every backlogged connection is bounded, then the server is fair and its fairness can be estimated in terms of this difference [11]. *Frame-based Fair Queueing* (FFQ) and *Starting Potential-based Fair Queueing* (SPFQ) are two example algorithms based on the class of Rate Proportional Servers [10], [12]. Both algorithms maintain the system potential function only as an approximation of the actual global state in the fluid model, but re-calibrate the system potential periodically to correct any discrepancies.

III. Shaped Rate-Proportional Servers (SRPS)

Figure 1 illustrates the logical structure of the new class of schedulers. The arriving packets first enter a shaper, and are admitted into the scheduler when they become *eligible.* Packets that have not become eligible for service remain in the shaper queue, while all the eligible packets wait for service in the scheduler queue. The eligibility criterion for admission of a packet into the scheduler queue is that the current value of system potential in the Rate-Proportional Server is equal to or greater than the finishing potential of

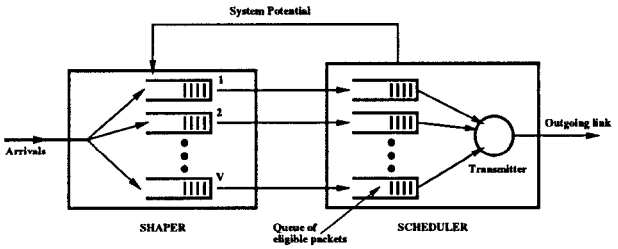

Fig. 1. Logical structure of a scheduler in the SRPS class.

the previous packet admitted from that connection. This is formalized in the following definition:

Definition 4: A packet k of session i is *eligible* for service at time t if and only if

$$P(t) \geq F_i^{k-1}, \tag{3.1}$$

where F_i^{k-1} denotes the finishing potential of the $(k-1)$th packet arrived from session i.

Note that the finishing potential can be computed using Eq. (2.2). We will call this general class of schedulers consisting of the shaper and a Rate-Proportional Server as *Shaped Rate-Proportional Servers* (SRPS).

Before we proceed to analyze this class of schedulers, we can make several observations. First, a scheduler in the SRPS class may or may not be work-conserving, depending on the system-potential function used by the corresponding Rate-Proportional Server. However, we will show that this does not affect either the worst-case service offered by the server, or its worst-case fairness as defined by Eq. (1.2). The work-conserving property, however, will affect its service fairness as defined by Eq. (1.1), since only a work-conserving scheduler is able to distribute free bandwidth among competing connections in a fair manner. Later, in Section V, we will describe the design of a special member of the SRPS class that is also work-conserving and allows a fair distribution of the free bandwidth.

A second observation we can make is that the shaping mechanism must have access to the current value of the system potential function from the scheduler to determine eligibility of packets. In addition, the scheduler in an SRPS may need access to the packets queued in the shaper for maintaining its system potential function. The amount of sharing of information necessary between the shaping and scheduling subsystems depends on the specific Rate-Proportional Server used to construct the SRPS. For example, if the VirtualClock algorithm is used as the RPS, the only information that needs to be shared is the system clock. Note also that, although the shaper and scheduler are shown as logically distinct functions in Figure 1, they may be implemented in an integrated manner, as will be shown in Section VII.

The addition of the shaping mechanism affects the arrival pattern at the scheduler. The result is improved fairness in the multiplexing of session traffic. However, we will now show that the worst-case service offered by the scheduler is not affected by the addition of the shaping function. In [10], we showed that every fluid-model RPS is an $\mathcal{LR}$ server with

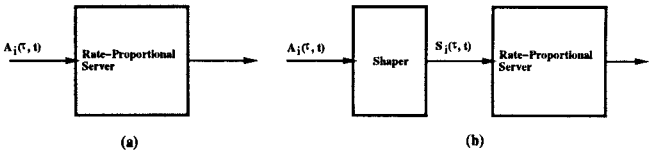

Fig. 2. An RPS scheduler with and without shaping. The busy periods seen by both schedulers are identical.

zero latency. We will now show that the busy periods of a session, under a given arrival pattern, remain identical when the shaping function is added. This will prove that every fluid-model SRPS is also an $\mathcal{LR}$ server with zero latency.

To prove that the shaping mechanism does not affect the busy periods, consider the two systems of Figure 2. The system in Figure 2(a) is a simple RPS, while in Figure 2(b) a shaper has been added to obtain an SRPS. Let $A_i(\tau,t)$ represent the arrival function for session i at the input of both systems during the interval $(\tau,t]$. Let $S_i(\tau,t)$ denote the service offered by the shaper to packets of session i during the interval $(\tau,t]$. Then, $S_i(\tau,t)$ is the arrival function for the scheduler part of the SRPS in Figure 2(b). We will show that the arrival function $S_i(\tau,t)$ results in the same busy periods for session i as those under $A_i(\tau,t)$.

Lemma 1: Let τ_j denote the starting time of the j-th busy period of session i in an RPS. Then, at any time t during this busy period, the service offered to the same traffic by the shaper in the corresponding SRPS satisfies the inequality

$$S_i(\tau_j,t) \geq \rho_i(t-\tau_j).$$

Proof: We will prove this lemma by double induction, first on the number of packets within a busy period, and subsequently on the number of busy periods.

Base Step: Consider the first busy period of session i in the RPS of Figure 2(a). This busy period started at time τ_1. Let L_i^m be the size of the m-th packet that arrived from session i during this busy period. Let t_m be the time when the mth packet would leave the server if the packet was serviced at exactly the reserved rate ρ_i. Then,

$$t_m = \tau_1 + \sum_{k=1}^{m} \frac{L_i^k}{\rho_i}. \tag{3.2}$$

We will prove the lemma during the first busy period by induction on the number of packets. When the first packet of the first busy period arrived at time τ_1, the system potential is at least equal to the finishing potential of the previous packet (assumed zero if no packet was transmitted before). Therefore, this packet will be immediately forwarded by the shaper to the scheduler, and the connection potential $P_i(t)$ will become equal to the system potential in the scheduler. Since the interval (τ_1,t_m) falls within a busy period of session i in the RPS of Figure 2(a), we must have from the definition of the busy period,

$$A_i(\tau_1,t) \geq \rho_i(t-\tau_1). \tag{3.3}$$

The service offered during the interval $(\tau_1,t]$, with $t \leq t_1$, to connection i from the shaper of Figure 2(b) will be

$$S_i(\tau_1,t) = L_i^1 = \rho_i(t_1-\tau_1), \quad \text{from Eq. (3.2)}$$

$$\geq \rho_i(t - \tau_1).$$

We can now extend this result for the entire busy period by induction on the number of packets within the busy period. Assume that, until time t_m, the service offered by the shaper to packets of session i is at least

$$S_i(\tau_1, t_m) \geq \rho_i(t_m - \tau_1).$$

We will show that this result also holds at any time during the interval $(t_m, t_{m+1}]$. Let t^* be the time at which the the mth packet was released by the shaper. Since this mth packet belongs to the same busy period, by induction hypothesis, we must have

$$t^* \leq t_{m-1} = t_m - \frac{L_i^m}{\rho_i}. \tag{3.4}$$

At time t^*, the potential of connection i became equal to or greater than the system potential, and the mth packet was released. The $(m+1)$th packet of session i can be released when the system potential reaches the finishing potential of the mth packet. Let t' be the time at which this occurs. Since the system potential in the Rate-Proportional Server is increasing with a rate at least equal to that of real time, we must have $t' \leq t^* + \frac{L_i^m}{\rho_i} \leq t_m$. Thus, the $(m+1)$th packet will become eligible before time t_m. The service offered by the shaper to session i from the start of the busy period to any time t, $t_m \leq t \leq t_{m+1}$, will be

$$\begin{aligned} S_i(\tau_1, t) &= S_i(\tau_1, t_m) + S_i(t_m, t) \\ &\geq \rho_i(t_m - \tau_1) + L_i^{m+1} \\ &\geq \rho_i(t_m - \tau_1) + \rho_i(t_{m+1} - t_m) \\ &\geq \rho_i(t_{m+1} - \tau_1) \\ &\geq \rho_i(t - \tau_1). \end{aligned}$$

Furthermore, it is easy to verify that, when the busy period of session i completes, there will be no more packets backlogged in the shaper from session i, and that the system potential will be at least equal to the potential of connection i.

Inductive Step: We will assume that the lemma holds until the nth busy period, that is, $S_i(\tau_j, t) \geq \rho_i(t - \tau_j)$ at any time t during the jth busy period, for $1 \leq j \leq n$. We need to prove the lemma for the $(n+1)$th busy period. Since there are no packets from the nth busy period backlogged in the shaper at time τ_{n+1}, and the system potential is at least equal to the potential of session i at time τ_{n+1}, we can repeat the proof for the first busy period to show that $S_i(\tau_{n+1}, t) \geq \rho_i(t - \tau_{n+1})$, at any time t during the $(n+1)$th busy period. □

The above lemma establishes that the worst-case service offered by the systems in Figure 2(a) and (b) are identical. That is, if τ_j is the beginning of the jth busy period of session i, in both systems we must have $W_{i,j}(\tau_j, t) \geq \rho_i(t - \tau_j)$, at any time t within the busy period. Therefore, the shaper-scheduler combination can be thought of as a single $\mathcal{LR}$-server with zero latency. We can thus arrive at the main result of this section:

Corollary 1: A Shaped Rate-Proportional Server is an $\mathcal{LR}$-server with zero latency.

Thus, from the theory of Latency-Rate Servers [13], we can conclude that the packet version of SRPS will have a delay bound identical to that of a simple RPS, which, in turn, is equal to that of Weighted Fair Queueing.

IV. Worst-case Fairness of SRPS

In this section we evaluate the worst-case fairness of algorithms belonging to the SRPS class, and show that every member of this class provides a worst-case fairness index (WFI) identical to that of WF^{2}Q. The latter, in turn, is optimal when the packet size is fixed [1].

The worst-case fairness can be seen as a measure of how closely the packet-by-packet scheduler approximates a fluid-model server where each session is served *at each instant* with a rate equal to its allocated rate ρ_i. Thus, a smaller WFI generally indicates a finer level of multiplexing of packets in the output stream. For a packet-based system, we can define the worst-case fairness index as follows: Let $Q_i(t)$ be the backlog of session i in the system at time t and $d_i(t)$ the time taken to clear this backlog. Then the worst-case fairness index is the minimum constant C_i such that

$$d_i(t) \leq \frac{Q_i(t)}{\rho_i} + C_i. \tag{4.1}$$

Note that, in the case of SRPS, the backlog $Q_i(t)$ includes packets queued at both the shaper and the scheduler.

To evaluate the worst-case fairness index of SRPS, we will consider a fluid version of the Rate-Proportional Server. In this fluid version, the shaper still delivers complete packets to the scheduler, but the scheduler can simultaneously service multiple sessions at different rates. This fluid system will be used as a reference to evaluate the fairness of the packet system. We will use $Q_i^P(t)$ and $Q_i^F(t)$ to denote the backlog of session i at time t in the packet system and the fluid system, respectively. Note that the backlog may not be identical in the two systems. We will first show that, in the fluid system, the backlog is always cleared at a rate at least equal to the allocated rate, if the potential of connection i is equal to the system potential at time t.

Lemma 2: Let $W_i^F(\tau, t)$ be the service received by session i in the fluid SRPS during an interval $(\tau, t]$ in which it is continuously backlogged. If $P_i(\tau) = P(\tau)$, then

$$W_i^F(\tau, t) \geq \rho_i(t - \tau).$$

Proof: Since session i is backlogged during the interval $(\tau, t]$, the connection potential is increasing by the normalized service offered to that connection. Thus,

$$P_i(t) - P_i(\tau) = \frac{W_i^F(\tau, t)}{\rho_i}. \tag{4.2}$$

Therefore,

$$W_i^F(\tau, t) = \rho_i(P_i(t) - P_i(\tau)). \tag{4.3}$$

Since $P_i(t) \geq P(t)$ and $P_i(\tau) = P(\tau)$, we can re-write this as

$$W_i^F(\tau, t) \geq \rho_i(P(t) - P(\tau)). \tag{4.4}$$

During the interval $(\tau, t]$, the system potential is increasing with a rate at least equal to that of real time. That is,

$$P(t) - P(\tau) \geq t - \tau. \tag{4.5}$$

From equations (4.4) and (4.5), we have $W_i^F(\tau, t) \geq \rho_i(t-\tau)$. This concludes the proof of Lemma 2.

We can now evaluate the worst-case fairness index of the packet-based SRPS.

Theorem 2: The queueing delay of a session-i packet arriving at time τ in a packet SRPS system is upper-bounded by

$$d_i^P(\tau) \leq \frac{Q_i^P(\tau)}{\rho_i} + \frac{L_{max}}{r} + L_i(\frac{1}{\rho_i} - \frac{1}{r}), \tag{4.6}$$

where $Q_i^P(\tau)$ is the backlog of session i at time τ, L_i the maximum packet size of session i, and L_{max} the maximum packet size across all sessions.

Proof: Let us denote with t_k the time at which the k-th packet of session i is released from the shaper to the scheduler in the SRPS. Note that this time is identical in both the fluid and packet versions considered. We will prove the theorem for each instant t in the interval $(t_k, t_{k+1}]$. $Q_i^F(t)$ is the backlog in the fluid server at time t and $Q_i^P(t)$ that in the packet-by-packet server. Notice that, at time t_k, the potential of connection i in the fluid server becomes equal to the system potential. Therefore, by Lemma 2, the service offered to session i after time t_k is with a rate at least equal to ρ_i, and without any latency.

Let us assume that the backlog of connection i is cleared at some time t^* in the fluid server. Then we can write

$$\begin{aligned} Q_i^F(t) &= W_i^F(t, t^*) \\ &= W_i^F(t, t_{k+1}) + W_i^F(t_{k+1}, t^*) \\ &\geq W_i^F(t, t_{k+1}) + \max(\rho_i(t^* - t_{k+1}), 0) \end{aligned} \tag{4.7}$$

We assume that if $t^* < t_{k+1}$, then $W_i^F(t_{k+1}, t^*) = 0$. We can re-write Eq. (4.7) as

$$\begin{aligned} t^* &\leq t_{k+1} + \frac{Q_i^F(t) - W_i(t, t_{k+1})}{\rho_i} \\ &\leq (t_{k+1} - t) + t + \frac{Q_i^F(t) - W_i(t, t_{k+1})}{\rho_i}. \end{aligned} \tag{4.8}$$

The above equation holds for the fluid sever. The time to clear the backlog at time t in the fluid server is thus given by $d_i^F(t) = t^* - t$. The packet sever may lag the fluid server at most by the transmission time of one packet [10]. Therefore, the time $d_i^P(t)$ needed to clear the backlog of connection i in the packet-by-packet server after time t is bounded by

$$\begin{aligned} d_i^P(t) &\leq d_i^F(t) + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^F(t) - W_i^F(t, t_{k+1})}{\rho_i} + (t_{k+1} - t) + \frac{L_{max}}{r} \end{aligned} \tag{4.9}$$

To evaluate the above expression, we will consider two separate cases.

Case 1: $Q_i^P(t) \geq Q_i^F(t)$. Let L_i^k denote the size of the kth packet of session i. Then,

$$\begin{aligned} W_i^F(t, t_{k+1}) &= \frac{L_i^k}{\rho_i} - W_i^F(t_k, t) \\ &\geq \frac{L_i^k}{\rho_i} - \min(L_i^k, r(t - t_k)). \end{aligned} \tag{4.10}$$

From Eq. (4.9) and (4.10), we can write

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^F(t)}{\rho_i} - \frac{L_i^k}{\rho_i} + \min(L_i^k, r(t - t_k)) + \frac{L_{max}}{r}. \tag{4.11}$$

By using the hypothesis that $Q_i^P(t) \geq Q_i^F(t)$, this becomes

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t)}{\rho_i} - \frac{L_i^k}{\rho_i} + \frac{\min(L_i^k, r(t - t_k))}{\rho_i} + \frac{L_{max}}{r}. \tag{4.12}$$

The right-hand side of the above equation is maximized when $L_i^k = r(t - t_k)$. Thus,

$$\begin{aligned} d_i^P(t) &\leq (t_{k+1} - t_k - \frac{L_i^k}{r}) + \frac{Q_i^P(t)}{\rho_i} - \frac{L_i^k}{\rho_i} + \frac{L_i^k}{\rho_i} + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^P(t)}{\rho_i} + L_i^k(\frac{1}{\rho_i} - \frac{1}{r}) + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^P(t)}{\rho_i} + L_i(\frac{1}{\rho_i} - \frac{1}{r}) + \frac{L_{max}}{r}. \end{aligned} \tag{4.13}$$

Case 2: $Q_i^F(t) > Q_i^P(t)$. In this case, let $\Delta Q = Q_i^F(t) - Q_i^P(t)$. We can re-write Eq.(4.9) as

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t) + \Delta Q}{\rho_i} - \frac{W_i^F(t, t_{k+1})}{\rho_i} + \frac{L_{max}}{r}. \tag{4.14}$$

At time t, the packet-by-packet server has offered more service to connection i than the fluid server. However, packet $k+1$ has not yet been released by the shaper. Thus, the additional service that fluid server will offer until time t_{k+1} is equal to the additional service that the packet-by-packet server has offered until time t, plus the service it will offer until time t_{k+1}. That is,

$$W_i^F(t, t_{k+1}) = \Delta Q + W_i^P(t, t_{k+1}). \tag{4.15}$$

From Eq. (4.14) and (4.15),

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t)}{\rho_i} - \frac{W_i^P(t, t_{k+1})}{\rho_i} + \frac{L_{max}}{r}. \tag{4.16}$$

Notice that at time t_k, the packet-by-packet server can only be behind the fluid server. Let us assume without loss of generality, that

$$W_i^F(0, t_k) = W_i^P(0, t_k) + \Delta W. \tag{4.17}$$

For the service offered to connection i by the packet-by-packet server after time t, we can write

$$\begin{aligned} W_i^P(t, t_{k+1}) &= W_i^P(t_k, t_{k+1}) - W_i^P(t_k, t) \\ &\geq L_i^k + \Delta W - W_i^P(t_k, t) \\ &\geq L_i^k + \Delta W - \min(L_i + \Delta W, r(t - t_k)). \end{aligned} \tag{4.18}$$

Thus, Eq. (4.16) becomes

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t)}{\rho_i} - (L_i + \Delta W) + \min(L_i^k + \Delta W, r(t - t_k)) + \frac{L_{max}}{r}. \quad (4.19)$$

The right-hand side of the above equation is maximized when $L_i + \Delta W = r(t - t_k)$. Therefore,

$$\begin{aligned} d_i^P(t) &\leq t_{k+1} - t_k - \frac{L_i^k + \Delta W}{r} + \frac{Q_i^P(t)}{\rho_i} + \frac{L_{max}}{r} \\ &\leq \frac{L_i^k}{\rho_i} - \frac{L_i^k + \Delta W}{r} + \frac{Q_i^P(t)}{\rho_i} + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^P(t)}{\rho_i} + \frac{L_{max}}{r} + L_i^k(\frac{1}{\rho_i} - \frac{1}{r}) \\ &\leq \frac{Q_i^P(t)}{\rho_i} + \frac{L_{max}}{r} + L_i(\frac{1}{\rho_i} - \frac{1}{r}). \end{aligned} \quad (4.20)$$

This concludes the proof of Theorem 2. Note that the worst-case fairness index of $(L_{max}/r) + L_i/\rho_i - L_i/r$ given by the theorem is identical to that of WF^{2}Q [1]. Thus, the worst-case fairness of WF^{2}Q can be obtained with a simple Rate-Proportional Server such as VirtualClock. For the case of an ATM network, the WFI becomes L/ρ_i, the best achievable in a packet-by-packet server.

V. Fairness of Shaped Rate Proportional Servers

The Service Fairness of any SRPS depends on the method of calculation of the system potential function. The degree of short-term unfairness of the algorithm depends on the difference between the system potential and the potentials of backlogged connections at any time [10]. In an idealized fluid server it is possible to update the system potential at any instant of time. However, in a packet-by-packet server it is desirable to update the system potential only when a packet departs from the system. Furthermore, it is advantageous to update the system potential based on only information extracted from the packet-by-packet implementation of the algorithm.

The Starting Potential-based Fair Queueing algorithm proposed in [12], and independently in [15] under the name WF^{2}Q+, is a special case of Rate Proportional Servers. The system-potential function in SPFQ is defined as follows: When the system is not busy the system potential function is equal to zero. During a system-busy period, the function $P(t)$ is a piecewise linear function of time t. Let τ_0 be the beginning of the current system-busy period. Then,

1. At times $\tau_1, \tau_2, \ldots, \tau_k$, with $\tau_0 < \tau_1 < \ldots < \tau_k$, a re-calibration is performed by updating P(t) to the value of a reference function, called the *base potential*, at that instant, if the system potential is lower than the base potential. That is,

$$P(\tau_j) = \max(P(\tau_j -), S^P(\tau_j)), \quad (5.1)$$

where $\tau_j -$ denotes the instant of time just before the update, and $S^P(t)$ is the base potential function used for calibration.

2. At any time time t between updates, the system potential increases linearly with time. That is,

$$P(t) = P(\tau_j) + (t - \tau_j), \quad \tau_j \leq t < \tau_{j+1}. \quad (5.2)$$

Note that, if the interval between successive re-calibrations is bounded, the system potential function satisfies the necessary conditions for the scheduling algorithm to belong to the RPS class.

In order to define the SPFQ algorithm completely, we must define the update instants τ_j and the base potential function $S_P(t)$ in Eq. (5.1). The calibration instants are chosen as the times at which a packet completes its service in the server. The base potential function $S_P(t)$ is chosen as follows: We define the *starting potential* of a packet of connection i as the potential of connection i when the first bit of the packet starts service in the fluid server. Let $R_i(t)$ denote the starting potential of the first packet in the queue of a backlogged connection i in the packet server at time t. That is, $R_i(t)$ is a step function that is increased every time a new packet is placed at the head of the queue of connection i. Then, we define the base potential function $S^P(t)$ as

$$S^P(t) = \min_{i \in B(t)} R_i(t), \quad (5.3)$$

where $B(t)$ denotes the set of backlogged connections in the scheduler at time t. That is, the base potential at any time t is defined as the *minimum* of the starting potentials of the backlogged connections. This value can be obtained by maintaining the starting-potential values of backlogged connections in a separate priority list. Although the starting potentials in Eq. (5.3) are defined with reference to the fluid server, simulation of the fluid server can be avoided by taking advantage of a result from [12]: If the starting potential of every backlogged session in the packet server is greater than or equal to $S^P(t)$ at time t, then the potential of each backlogged session in the corresponding fluid server at time t is also greater than or equal to $S^P(t)$. This allows starting potential values obtained from the packet server to be used in Eq. (5.3).

A. Shaped SPFQ Algorithm

The algorithm above needs to be modified slightly to accommodate shaping. Since the scheduler in the shaped version sees only the eligible packets, the definition of the base potential function must include packets queued in both the scheduler and the shaper. Therefore, we modify Eq. (5.3) as

$$S^P(t) = \min_{i \in B^P(t) \cup B^S(t)} R_i(t), \quad (5.4)$$

where $B^P(t)$ denotes the set of backlogged connections in the scheduler and $B^S(t)$ those in the shaper, at time t. This ensures that the value of the base potential function never exceeds the minimum starting potential value in the system. Therefore, when the system potential is updated using the base potential, the former can never exceed the starting potential of any backlogged connection. Thus, the scheduler part of Shaped SPFQ remains a Rate-Proportional Server.

The above modification results in a work-conserving algorithm belonging to the SRPS class. When the scheduler queue becomes empty, and the shaper still has queued packets, the system potential will automatically increase to the minimum of the starting potentials of the waiting packets, causing one or more packets to be eligible for service. The following lemma proves the work-conserving property.

Lemma 3: Shaped SPFQ is a work-conserving server.

Proof: We will prove the lemma by contradiction. Let us assume that the server is not work-conserving. Consider a time t when there are packets in the system and the server is idle. Then, at this time the packets have not yet been released from the shaper. Thus, the system potential at time t is less than the finishing potentials of the last packet transmitted by all the backlogged connections in the system. From the definition of the packet-by-packet SPFQ server, however, it is easy to verify that the system potential at that point will become at least equal to the minimum of the starting potentials of any packet that is backlogged in the system. Thus, at that time the system potential is at least equal to the finishing potential of at least one connection backlogged in the shaper. Hence, there is always at least one packet that is eligible for transmission. □

B. Fairness of Shaped SPFQ

In this section we derive the service fairness index of SPFQ and show that it is comparable to that of Self-Clocked Fair Queueing, and superior to that of Weighted Fair Queueing. We denote by $\hat{W}_i(t_1,t_2)$ the service received by session i during the interval $(t_1,t_2]$, including the partial service received by a packet currently in transmission. Then, the service fairness index of the algorithm with respect to two sessions i and j that are continuously backlogged during the interval $(t_1,t_2]$ is given by

$$\left|\frac{\hat{W}_i(t_1,t_2)}{\rho_i} - \frac{\hat{W}_j(t_1,t_2)}{\rho_j}\right|.$$

The following lemma provides an upper bound on the service fairness index of Shaped SPFQ algorithm.

Lemma 4: For any two connections i,j that are continuously backlogged in the interval $(t_1,t_2]$ in the Shaped SPFQ server,

$$\left|\frac{\hat{W}_i(t_1,t_2)}{\rho_i} - \frac{\hat{W}_j(t_1,t_2)}{\rho_j}\right| \leq 2\max(\frac{L_i}{\rho_i},\frac{L_j}{\rho_j}) + \max(\frac{L_{max}}{\rho_i},\frac{L_{max}}{\rho_j}),$$

where L_i and L_j are the maximum packet sizes for connections i and j, respectively, and L_{max} the maximum packet size across all sessions.

The above bound applies to a packet server. Interested readers are referred to [16] for a detailed proof.

When the maximum packet size is identical for all sessions, the right-hand side of this equation reduces to $3\max(1/\rho_i, 1/\rho_j)L_{max}$. On comparing with the corresponding fairness parameter of Self-Clocked Fair Queueing [5], we note that the service fairness of Shaped SPFQ is within a constant factor of 3. Furthermore, the service fairness of Weighted Fair Queueing has been shown to be [13]

$$\max(C_j + \frac{L_{max}}{\rho_i} + \frac{L_j}{\rho_j}, C_i + \frac{L_{max}}{\rho_j} + \frac{L_i}{\rho_i}),$$

where

$$C_i = \min((V-1)\frac{L_{max}}{\rho_i}, \max_{1\leq n\leq V}(\frac{L_n}{\rho_n})).$$

Thus, it is easy to note that the service fairness index of WFQ is determined by the *minimum* among the bandwidth allocations of the sessions sharing the link, while in both Self-Clocked Fair Queueing and Shaped SPFQ, the service fairness depends only on the allocations of the two sessions under observation.

VI. Shaped VirtualClock

Perhaps the simplest implementation of SRPS is obtained when VirtualClock [14] is used as the underlying Rate-Proportional Server. Since the system-potential function in VirtualClock is the real time itself, the shaping and scheduling functions in the Shaped VirtualClock server can operate independently, except for sharing a common real-time clock. Such a shaper-scheduler combination is attractive when both bandwidth guarantees and bounds on the burstiness of session traffic are required. For example, such a server can be used for shaping and multiplexing constant bit-rate (CBR) flows at an ATM network interface, and at reshaping points within the network. In this section, we prove some simple properties of the Shaped VirtualClock algorithm. An efficient implementation of the shaper-scheduler combination is described later in Section VII.

First, we show that the shaper in the Shaped VirtualClock server releases packets exactly at the rate of ρ_i.

Lemma 5: Let L_i^k be the length of the kth packet arrived from session i during one of its busy periods that started at time τ. Then, the mth packet is released by the shaper in the Shaped VirtualClock server at time

$$t_m = \tau + \sum_{k=1}^{m}\frac{L_i^k}{\rho_i}.$$

Proof: The proof follows directly from the fact that the system potential in the server increases at the rate of real time. □

Since a scheduler can distort session traffic even when the arriving traffic of each session is shaped perfectly, an important performance metric for a shaper-scheduler is the burstiness of session traffic at its output. The following lemma characterizes the burstiness at the output of the Shaped VirtualClock server.

Lemma 6: The worst-case burstiness of session-i traffic at the output of the Shaped VirtualClock server is given by

$$L_i + \rho_i(\frac{L_{max}}{r}),$$

where L_i is the maximum packet size of session i.

Proof: By Lemma 5, the input traffic to the VirtualClock scheduler can be seen as leaky-bucket shaped with a bucket

size of $\sigma_i = 0$. Therefore, from the general result for output burstiness of $\mathcal{LR}$ servers [13], we have

$$\sigma_{out} = \sigma_i + \rho_i(\frac{L_i}{\rho_i} + \frac{L_{max}}{r}) = L_i + \rho_i(\frac{L_{max}}{r}). \quad (6.1)$$

In the case of an ATM network, the expression for burstiness reduces to $L(1 + \rho_i/r)$. Since $\rho_i/r < 1$, we can omit the second term in a packet-level system. Note that the best achievable burstiness in a packet-level scheduler is L. Therefore, we can conclude that the burstiness of VirtualClock is optimal when the packets are of fixed size.

VII. Implementation

In this section, we describe an implementation of a general Shaped Rate-Proportional Server (SRPS) that integrates the shaping and scheduling functions. This implementation is intended for use in network adapters, where a set of flows originating at an end system needs to be individually shaped and multiplexed into an outgoing link. The integrated shaper-scheduler requires only $O(\log_2 V)$ operations to be performed within a packet transmission time, making it suitable for implementation in high-speed networks, especially in hardware. This time-complexity is the best known for an integrated shaper-scheduler, and is an improvement over existing implementations that have a complexity of $O(V)$. For convenience, we will describe an implementation with respect to ATM networks supporting a fixed cell-size. It can easily be modified for packet networks with variable packet sizes.

The implementation in this section can be applied to any member of the SRPS family. Thus, using VirtualClock as the scheduler, it can be used to implement a traffic shaper-scheduler for an ATM network adapter for scheduling constant bit-rate (CBR), variable bit-rate (VBR), and available bit-rate (ABR) traffic, all of which require shaping. The idle slots left can then be used to transmit unspecified bit-rate (UBR) traffic. Similarly, when Starting Potential-based Fair Queueing (SPFQ) is used as the scheduler, the resulting scheduler (Shaped SPFQ) can be used as the scheduling algorithm in an ATM switch.

The basic difficulty in the implementation of the shaper-scheduler combination arises from the processing of packets to mark their conformance. When the transmission of a packet has been completed, the system must transfer all packets that became eligible for service during the transmission time of the last packet to the scheduler queue. This processing will result in $O(V)$ complexity. However, we can avoid this complexity by noting that only a small number of packets transferred will be candidates for transmission by the scheduler in the next cycle. Thus, by transferring only eligible packets that are candidates for selection in the next cycle, and retaining other eligible packets in the shaper queue, the complexity can be reduced significantly. We will show that it is sufficient to transfer at most *two* packets from the shaper to the scheduler during the transmission time of each packet.

The fixed size of the ATM cell allows potentials to be represented as integers, instead of the floating-point numbers

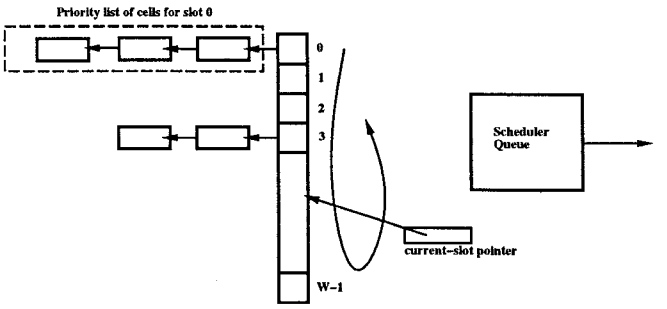

Fig. 3. Block diagram of an integrated shaper-scheduler for ATM networks.

required in the general implementation. Since the ATM cell has a fixed size, the unit of time is chosen as the transmission time of an ATM cell through the outgoing link. Bandwidth reservations of sessions are assumed to be in terms of number of ATM cells per second. Integer representations of starting and finishing potentials is achieved by imposing the restriction that the values of $1/\rho_i$ for each session i is an integer. Note that a suitable scaling constant can be added, if required, to increase the granularity required in the bandwidth reservations of the individual sessions. That is, a constant $K \geq 1$ can be chosen such that the value of K/ρ_i is an integer for each session i.

The system potential function in this implementation is maintained as an integer, advancing it by one after the transmission of each ATM cell. A re-calibration step may then be used to update the system potential close to the potentials of connections with ATM cells queued in the system, if the underlying Rate-Proportional Server requires such re-calibration.

Figure 3 illustrates the implementation of the shaper-scheduler combination for an ATM network. The shaper queue is implemented as a calendar queue based on the starting potentials of the queued cells. The system maintains W flip-flops, each representing the state of a slot in the calendar queue. The flip-flops are organized as a circular queue. A pointer, referred to as *current-slot pointer*, represents the current value of system potential (modulo W) in the scheduler. The pointer is moved cyclically through the slots as the system potential is updated at the end of transmission of each cell. Each slot in the calendar queue represents a distinct value of the starting potential. Associated with each slot j is a list of cells whose starting potentials have a value of j (modulo W). The cells in each list is maintained in sorted order of their finishing potentials, so that the cell with the minimum finishing potential appears at the head of the list.

A separate priority queue of cells is maintained by the scheduler, prioritized by their finishing potential values. These represent conforming cells whose starting potentials are lower or equal to the current system-potential. Only the first cell in each session's queue is maintained in these data structures, so that the maximum number of nodes in the shaper and scheduler queues taken together, is V.

The processing performed by the algorithm can be divided into two parts: (i) a part that is performed when a new cell arrives, and (ii) a part that is executed when the transmission of a cell has been completed. Since the transmission time of an ATM cell is very short, the cells arriving at the scheduler need to be processed only at the boundaries of cell transmissions, so that calculation of the system potential need not take into account the partial service received by the cell currently being transmitted, as was done in the algorithm of Section V. When a new cell arrives from a session that has no cells currently in the system, the cell must be added to either one of the shaper lists or the scheduler queue. The starting and finishing potentials of the arriving cell are first determined. If this starting potential value is not below the system potential, the cell is directly added to the scheduler queue, prioritized by its finishing potential. Otherwise, it is added to the shaper list corresponding to its starting potential value (modulo W), again prioritized by the finishing potential.

The processing performed on the departure of a cell can be summarized as follows: First, the system potential value is advanced by one. This may be followed by a re-calibration step, depending on the specific scheduler used. If the updated value of system potential is j (modulo W), all the cells in the shaper list associated with slot j have now become eligible. However, transferring all the cells in this list to the scheduler queue will incur a time-complexity of $O(V)$. Instead, only the following cells are transferred to the scheduler queue.

1. The cell at the head of the shaper list associated with slot j.
2. If the cell transmitted by the scheduler in the previous slot had a starting potential value of k (modulo W), the cell at the head of the shaper list associated with slot k, if any, is also moved to the scheduler queue.

It is easy to observe that none of the other eligible cells that remain in the shaper lists are candidates for transmission in the next cycle, since their finishing potentials are greater than or equal to that of the cell at the head of the scheduler queue.

Note that, on completion of transmission of a cell from session i, the next cell from the session must be added to the shaper-scheduler, if it has already arrived. This operation is identical to the case of arrival into an empty queue, discussed earlier. Since insertions into the priority lists can be performed in $O(\log_2 V)$ time, the asymptotic time-complexity of the shaper-scheduler combination is $O(\log_2 V)$.

VIII. Conclusion

In this paper, we introduced a general methodology for the design of fair scheduling algorithms by combining a shaping mechanism with a scheduling algorithm from the class of Rate-Proportional Servers [10], [11]. These algorithms provide a delay bound equal to that of Weighted Fair Queueing and superior fairness properties at a much lower complexity of implementation. We described two specific algorithms from this class of Shaped Rate-Proportional Servers, namely Shaped VirtualClock and Shaped SPFQ. The former is attractive for shaping and scheduling constant bit-rate flows at the interface of a network, while the latter is attractive as a switch scheduling algorithm. It is hoped that this design methodology will lead to the development of other scheduling algorithms in the future.

An open question that remains is whether a single fairness measure can be found that captures the key fairness characteristics of scheduling algorithms. It is clear that most of the metrics proposed in the literature to measure the fairness of scheduling algorithms do not adequately capture their fairness properties. For example, the worst-case fairness we used does not take into account how the free bandwidth is distributed among competing sessions. The service fairness index, on the other hand, is valid only when the sessions are greedy. Thus, it is possible for a scheduling algorithm to have ideal fairness in terms of these metrics, yet cause significant discrepancies in the the amount of service given to competing connections as compared to the ideal fluid-model GPS.

References

[1] J. C. R. Bennett and H. Zhang, "WF^{2}Q: Worst-case Fair Weighted Fair Queueing," in *Proceedings of IEEE INFOCOM 96*, pp. 120–128, March 1996.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Internetworking: Research and Experience*, vol. 1, no. 1, pp. 3–26, 1990.

[3] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 368–379, April 1990.

[4] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping," in *Proceedings of IEEE INFOCOM '96*, pp. 102–110, March 1996.

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

[6] S. Golestani, "Network delay analysis of a class of fair queueing algorithms," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1057–70, August 1995.

[7] S. Keshav, "A control-theoretic approach to flow control," in *Proceedings of ACM SIGCOMM'91*, pp. 3–15, September 1991.

[8] A. Parekh, *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. Ph.D. thesis, Massachusetts Institute of Technology, 1992.

[9] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control — the single node case," in *Proceedings of IEEE INFOCOM '92*, vol. 2, pp. 915–924, May 1992.

[10] D. Stiliadis and A. Varma, "Design and analysis of Frame-based Fair Queueing: A new traffic scheduling algorithm for packet-switched networks," in *Proceedings of ACM SIGMETRICS '96*, pp. 104–115, May 1996.

[11] D. Stiliadis and A. Varma, "Rate-Proportional Servers: A design methodology for fair queueing algorithms." submitted to IEEE/ACM Transactions on Networking, January 1996 (http://www.cse.ucsc.edu/research/hsnlab/publications/).

[12] D. Stiliadis and A. Varma, "Efficient fair-queueing algorithms for ATM and packet networks." submitted to IEEE/ACM Transactions on Networking, January 1996 (http://www.cse.ucsc.edu/research/hsnlab/publications/).

[13] D. Stiliadis and A. Varma, "Latency-Rate servers: A general model for analysis of traffic scheduling algorithms," in *Proceedings of IEEE INFOCOM '96*, pp. 111–119, April 1996.

[14] L. Zhang, "VirtualClock: a new traffic control algorithm for packet switching networks," *ACM Transactions on Computer Systems*, vol. 9, pp. 101–124, May 1991.

[15] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proceedings of ACM SIGMETRICS 96*, pp. 143—156, September 1996.

[16] D. Stiliadis and A. Varma, "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms," Tech. Rep., U.C. Santa Cruz, May 1996. Available from http://www.cse.ucsc.edu/research/hsnlab/publications/.