Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms

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

Abstract

In this paper, we develop a general model, called *Latency-Rate servers* ($\mathcal{LR}$-servers), for the analysis of traffic scheduling algorithms in broadband packet networks. The behavior of an $\mathcal{LR}$ scheduler is determined by two parameters — the *latency* and the allocated *rate*. We show that several well-known scheduling algorithms, such as Weighted Fair Queueing, VirtualClock, Self-Clocked Fair Queueing, Weighted Round Robin, and Deficit Round Robin, belong to the class of $\mathcal{LR}$-servers. We derive tight upper bounds on the end-to-end delay, internal burstiness, and buffer requirements of individual sessions in an arbitrary network of $\mathcal{LR}$-servers in terms of the latencies of the individual schedulers in the network, when the session traffic is shaped by a leaky bucket. Thus, the theory of $\mathcal{LR}$-servers enables computation of tight upper-bounds on end-to-end delay and buffer requirements in a network of servers in which the servers on a path may not all use the same scheduling algorithm. We also define a self-contained approach to evaluate the fairness of $\mathcal{LR}$-servers and use it to compare the fairness of many well-known scheduling algorithms.

I. Introduction

Broadband packet networks are currently enabling the integration of traffic with a wide range of characteristics within a single communication network. Different types of traffic have significantly different quality-of-service (QoS) requirements [1]. Providing QoS guarantees in a packet network requires the use of traffic scheduling algorithms in the switches (or routers). The function of a scheduling algorithm is to select, for each outgoing link of the switch, the packet to be transmitted in the next cycle from the available packets belonging to the flows sharing the output link.

Several service disciplines are known in the literature for providing bandwidth guarantees to individual sessions in output-buffered switches [2], [3], [4], [5], [6], [7], [8], [9]. Many of these algorithms are also capable of providing deterministic delay guarantees when the burstiness of the session traffic is bounded (for example, shaped by a leaky bucket).

While there is a significant volume of work on the analysis of various traffic scheduling algorithms, most of these studies apply only to a particular scheduling algorithm. Little work has been reported on analyzing the characteristics of the service offered to individual sessions in a network of servers where the schedulers on the path of the session may use different scheduling algorithms. Since future networks are unlikely to be homogeneous in the type of scheduling algorithms employed by the individual switches (routers), a general model for the analysis of scheduling algorithms will be a valuable tool in the design and analysis of such networks. Our basic objective in this paper is to develop such a general model to study the worst-case behavior of individual sessions in a network of schedulers where the schedulers in the network may employ a broad range of scheduling algorithms. Such an approach will enable us to calculate tight bounds on the end-to-end delay of individual sessions and the buffer sizes needed to support them in an arbitrary network of schedulers.

Our basic approach consists in defining a general class of schedulers, called *Latency-Rate* servers, or simply $\mathcal{LR}$-servers. The theory of $\mathcal{LR}$ servers provides a means to describe the worst-case behavior of a broad range of scheduling algorithms in a simple and elegant manner. For a scheduling algorithm to belong to this class, it is only required that the *average* rate of service offered by the scheduler to a busy session, over every interval starting at time Θ from the beginning of the busy period, is at least equal to its reserved rate. The parameter Θ is called the *latency* of the scheduler. All the work-conserving schedulers known to us, including Weighed Fair Queueing (or PGPS), VirtualClock, SCFQ, Weighted Round Robin, and Deficit Round Robin, exhibit this property and can therefore be modeled as $\mathcal{LR}$-servers.

The behavior of an $\mathcal{LR}$ scheduler is determined by two parameters — the *latency* and the *allocated rate*. The latency of an $\mathcal{LR}$-server is the worst-case delay seen by the first packet of the busy period of a session, that is, a packet arriving when the session's queue is empty. The latency of a particular scheduling algorithm may depend on its internal parameters, its transmission rate on the outgoing link, and the allocated rates of various sessions. However, we show that the maximum end-to-end delay experienced by a packet in a network of schedulers can be calculated from only the latencies of the individual schedulers on the path of the session, and the traffic parameters of the session that generated the packet. Since the maximum delay in a scheduler increases directly in proportion to its latency, the model brings out the significance of using low-latency schedulers to achieve low end-to-end delays. Likewise, upper bounds on the queue size and burstiness of individual sessions at any point within the network can be obtained directly from the latencies of the schedulers. We also show how the latency parameter can be computed for a given scheduling algorithm by deriving the latencies of several well-known schedulers.

Our approach in modeling the worst-case behavior of scheduling algorithms with respect to an end-to-end ses-

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

sion is related to the work of Cruz [10], [11], Zhang [12], and Parekh and Gallager [4], [13]. Cruz [10], [11] analyzed the end-to-end delay, buffer requirements, and internal network burstiness of sessions in an arbitrary topology network where all sources are leaky-bucket controlled. While the objectives of our analysis are similar, there are three major differences between the approaches taken: First, the class of scheduling algorithms we study are capable of providing bandwidth guarantees to individual sessions. Therefore, we can derive deterministic end-to-end delay guarantees that are independent of the behavior of other sessions. Second, we do not study individual schedulers in isolation and accumulate the session delays as in Cruz's work , but instead model the behavior of the chain of schedulers on the path of the connection as a whole. Third, we estimate the latency parameters for the individual schedulers tightly, taking into account their internal structure. Thus, our approach, in general, provides much tighter end-to-end delay bounds for individual sessions.

Another model for delay-analysis based on a class of guaranteed-rate servers was presented in [14]. The main problem of this model, however, is that it is closely coupled with time-stamp based algorithms; the analysis of scheduling algorithms based on a different architecture is not straightforward. The $\mathcal{LR}$-class provides a more natural approach for analyzing the worst-case behavior of traffic-scheduling algorithms, independent of the scheduler architecture. Finally, Golestani recently presented a delay analysis of a class of fair-queueing algorithms including Self-Clocked Fair Queueing [15]. However, this analysis does not apply to unfair algorithms like VirtualClock.

In addition to the delay analysis, we also study the fairness characteristics of $\mathcal{LR}$-schedulers. The fairness analysis was motivated by Golestani's work [5], where a self-contained approach for fairness was defined. This approach is based on comparing the normalized service offered to any two connections that are continuously backlogged over an interval of time. We will analyze many well-known scheduling algorithms belonging to the $\mathcal{LR}$ class using this approach.

II. A Common Framework

Scheduling algorithms for output-buffered switches have been classified based on two criteria: work-conservation and internal architecture. Neither of these classifications provides us with a common framework that will allow evaluation of their relative performance in real networks. In this section we discuss the three important attributes of scheduling algorithms that are most important in their application in real networks. These are (i) delay behavior, (ii) fairness, and (iii) implementation complexity. We will, therefore, compare schedulers along these three dimensions.

A. End-to-End Delay Guarantees

The algorithm must provide end-to-end delay guarantees for individual sessions, without severely under-utilizing the network resources. In order to provide a deterministic delay bound, it is necessary to bound the burstiness of the session at the input of the network. The most common approach for bounding the burstiness of input traffic is by shaping through a leaky bucket. Several previous studies have used this traffic model [4], [10], [11], [13]. We use the same traffic model in our derivations of end-to-end session delays and assume that the traffic of session i is smoothed through a leaky bucket with parameters (σ_i, ρ_i), where σ_i is the maximum burstiness and ρ_i is the average arrival rate. In deriving the end-to-end delay bound for a particular session, however, we do not make any assumptions about the traffic from the rest of the sessions sharing the same links of the network.

In addition to minimizing the end-to-end delay in a network of servers, the delay behavior of an ideal algorithm includes the following attributes:

1. Insensitivity to traffic patterns of other sessions: Ideally, the end-to-end delay guarantees for a session should not depend on the behavior of other sessions. This is a measure of the level of isolation provided by the scheduler to individual sessions.
2. Delay bounds that are independent of the number of sessions sharing the outgoing link:
3. Ability to control the delay bound of a session by controlling only its bandwidth reservation:

Note that the three attributes are related. We will show later that the worst-case delay behavior of a session can differ greatly in two different schedulers with identical bandwidth reservations.

B. Fairness

Significant discrepancies may exist in the service provided to different sessions over the short term among scheduling algorithms. Some schedulers may penalize sessions for service received in excess of their reservations at an earlier time. Thus, a backlogged session may be starved until others receive an equivalent amount of normalized service, leading to short-term unfairness. Therefore, two scheduling algorithms capable of providing the same delay guarantee to a session may exhibit vastly different fairness behaviors.

While there is no common accepted method for estimating the fairness of a scheduling algorithm, it is easy to define fairness in an informal manner. In general, we would like the system to always serve connections proportional to their reservations and distribute the unused bandwidth left behind by idle sessions equally among the active ones. In addition, sessions should not be penalized for excess bandwidth they received while other sessions were idle. Following Golestani's work [5], we define 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.

C. Implementation Complexity

Finally, schedulers differ greatly in their implementation complexity. The scheduling algorithm may need to be implemented in hardware in a high-speed network. In addition, it is desirable to have the time-complexity of the algorithm not depend on the number of active connections in the scheduler.

If V is the maximum number of connections that may share an output link, the implementation of a scheduler

based on the sorted-priority architecture involves three main steps for processing each cell [16]:

1. Calculation of the timestamp: The PGPS scheduler has the highest complexity in this respect with a worst-case processing overhead of $O(V)$ per packet transmission [4].
2. Insertion in a sorted priority list: The first cell of each session's queue must be stored in a sorted priority list. When a cell arrives into an empty queue, its insertion into the priority list requires $O(\log V)$ steps.
3. Selection of the cell with the minimum timestamp for transmission: Since the cells are stored in a sorted-priority structure, the cell with the highest priority may be retrieved in $O(\log V)$ time.

The last two operations are identical for any sorted-priority architecture.

Frame-based algorithms such as Weighted Round Robin and Deficit Round Robin can be implemented in $O(1)$ time, without any timestamp calculations. Unfortunately, these algorithms yield delay bounds that may grow linearly with the number of sessions sharing the outgoing link. Thus, in practice, the scheduling algorithm must trade off the complexity of implementation with the other desirable properties of low delay and bounded short-term unfairness.

III. $\mathcal{LR}$-Servers

In this section we introduce $\mathcal{LR}$-servers and derive some key results on their delay behavior. This model will allow us to derive deterministic bounds on end-to-end delays in an arbitrary topology network. In addition, it will help us define the necessary properties of a scheduling algorithm to utilize the network resources efficiently.

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

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 system based on the fluid model, $W_i(\tau, t)$ is a continuous function of t. However, in the 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. Thus, the fluid model may be viewed as a special case of the packet-by-packet model with infinitesimally small packets.

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 $Q_i(t) > 0$.

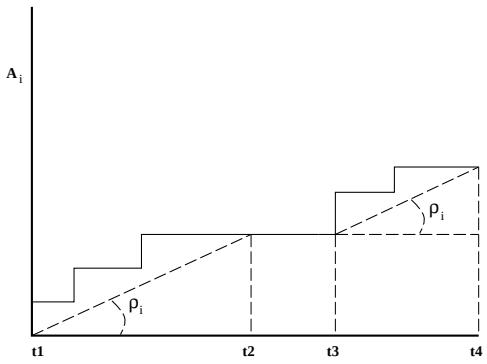

Fig. 1. Intervals $(t_1, t_2]$ and $(t_3, t_4]$ are two different busy periods.

Definition 3: A **session** i **busy period** is a maximal interval of time $(\tau_1, \tau_2]$ such that for any time $t \in (\tau_1, \tau_2]$, packets of session i arrive with rate greater than or equal to ρ_i, or,

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

A session busy period is the maximal interval of time during which if the session were serviced with exactly the guaranteed rate, it would be remain continuously backlogged (Figure 1). Multiple session-i busy periods may appear during a system busy period.

The session busy period is defined only in terms of the arrival function and the allocated rate. It is important to realize the basic distinction between a session backlogged period and a session busy period. The latter is defined with respect to a hypothetical system where a backlogged connection i is serviced at a constant rate ρ_i, while the former is based on the actual system where the instantaneous service rate varies according to the number of active connections and their service rates. Thus, a busy period may contain intervals during which the actual backlog of session i traffic in the system is zero; this occurs when the session receives an instantaneous service rate of more than ρ_i during the busy period.

Note that, when the same traffic distribution is applied to two different schedulers with identical reservations, the resulting backlogged periods can be quite different. This makes it difficult to use the session-backlogged period for analyzing a broad class of schedulers. The session busy period, on the other hand, depends only on the arrival pattern of the session and its allocated rate, and can therefore be used as an invariant in the analysis of different schedulers. This is the fundamental reason why the following definition of an $\mathcal{LR}$-server is based on the service received by a session over a busy period. Since we are interested in a worst-case analysis of the system, the session busy period provides us a convenient means to bound the delay within the system.

We can now define the general class of *Latency-Rate*($\mathcal{LR}$) servers. A server in this class is characterized by two parameters: *latency* Θ_i and *allocated rate* ρ_i. Let us assume that the jth busy period of session i starts at time τ. We denote by $W_{i,j}^{\mathcal{S}}(\tau, t)$ the total service provided to the packets of the session that arrived after time τ and until time t by server $\mathcal{S}$. Notice that the total service offered to session i in this interval $(\tau, t]$ may actually be more than $W_{i,j}^{\mathcal{S}}(\tau, t)$ since some

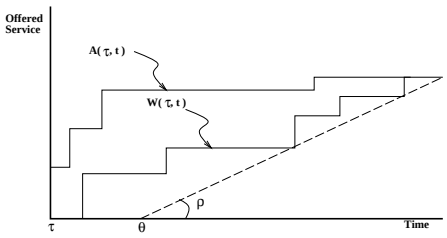

Fig. 2. An example of the behavior of an $\mathcal{LR}$ scheduler.

packets from a previous busy period, that are still queued in the system, may be serviced as well.

Definition 4: A server $\mathcal{S}$ belongs in the class $\mathcal{LR}$ if and only if for all times t after time τ that the jth busy period started and until the packets that arrived during this period are serviced,

$$W_{i,j}^S(\tau, t) \geq \max(0, \rho_i(t - \tau - \Theta_i^S)).$$

Θ_i^S is the minimum non-negative number that satisfies the above inequality.

The definition of $\mathcal{LR}$ servers involves only the two parameters, latency and rate. The right-hand side of the above equation defines an envelope to bound the minimum service offered to session i during a busy period (Figure 2). It is easy to observe that the latency Θ_i^S represents the worst-case delay seen by the first packet of a busy period of session i. Notice that the restriction imposed is that the service provided to a session from the beginning of its busy period is lower-bounded. Thus, for a scheduling algorithm to belong to this class, it is only required that the *average* rate of service offered by the scheduler to a busy session, over every interval starting at time Θ_i^S from the beginning of the busy period, is at least equal to its reserved rate. This is much less restrictive than GPS multiplexing, where the *instantaneous* bandwidth offered to a session is bounded. That is, the lower bound on the service rate of GPS multiplexing holds for any interval $(\tau, t]$ that a session is backlogged, whereas in $\mathcal{LR}$-servers the restriction holds only for intervals starting at the beginning of the busy period. Therefore, GPS multiplexing is only one member of the $\mathcal{LR}$ class.

The latency parameter depends on the scheduling algorithm used as well as the allocated rate and traffic parameters of the session being analyzed. For a particular scheduling algorithm, several parameters such as its transmission rate on the outgoing link, number of sessions sharing the link, and their allocated rates, may influence the latency.

In our definition of $\mathcal{LR}$ servers, we made no assumptions on whether the server is based on a fluid model or a packet-by-packet model. The only requirement that we impose, however, is that a packet is not considered as departing the server until its last bit has departed. Therefore, packet departures must be considered as impulses. This assumption is needed to bound the arrivals into the next switch in a chain of schedulers. We will remove this assumption later from the last server of a chain to provide a slightly tighter bound on the end-to-end session delay in a network of schedulers. In a fluid system, we require that all schedulers operate on a fluid basis and the maximum packet size to be infinitesimally small.

We will now derive delay bounds for $\mathcal{LR}$ schedulers. We will first consider the behavior of a session in a single node, and subsequently extend the analysis to networks of $\mathcal{LR}$ servers. In both cases, we will assume that the input traffic of the session we analyze is leaky-bucket smoothed and the allocated rate is at least equal to the average arrival rate. That is, if i is the session under observation, its arrivals at the input of the network during the interval $(\tau, t]$ satisfy the inequality

$$A_i(\tau, t) \leq \sigma_i + \rho_i(t - \tau), \tag{3.1}$$

where σ_i and ρ_i denote its burstiness and average rate, respectively. However, we make no assumptions about the input traffic of other sessions.

A. Analysis of a Single $\mathcal{LR}$ Server

Assume a set of V sessions sharing the same output link of an $\mathcal{LR}$ server.

The following theorem bounds the queueing delays within the server, as well as the buffer requirements, for session i.

Theorem 1: If $\mathcal{S}$ is an $\mathcal{LR}$-server, then the following bounds must hold:

1. If $Q_i^S(t)$ is the backlog of session i at time t, then
$$Q_i^S(t) \leq \sigma_i + \rho_i \Theta_i^S. \tag{3.2}$$
2. If D_i^S is the delay of any packet of session i in server S,
$$D_i^S \leq \frac{\sigma_i}{\rho_i} + \Theta_i^S. \tag{3.3}$$
3. The output traffic conforms to the leaky bucket model with parameters $(\sigma_i + \Theta_i^S \rho_i, \rho_i)$.

A proof for the above theorem can be found in [17]. Notice that both the output burstiness of the traffic as well as the delay bound depend strongly on the latency of the server. As we show in the next section, this dependency becomes stronger in a network of $\mathcal{LR}$-servers.

B. Analysis of a Network of $\mathcal{LR}$ Servers

The only restrictions that we impose in the network is that all the servers belong to the $\mathcal{LR}$ class and that the traffic of session i under observation is shaped at the source with a leaky bucket (σ_i, ρ_i). We will also assume that the bandwidth reservation of the session at every node in its path is at least ρ_i.

We first prove the following lemma:

Lemma 1: The traffic process after the kth node in a chain of $\mathcal{LR}$ servers is a leaky bucket process with parameters

$$\sigma_i + \rho_i \sum_{j=1}^{k} \Theta_i^{(S_j)}, \text{ and } \rho_i,$$

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

We already proved in Theorem 1, that the output traffic of an $\mathcal{LR}$ server conforms to the leaky bucket model with parameters $(\sigma_i + \rho_i \Theta_i^S, \rho_i)$. But this means that the input traffic in the next node conforms to the same model. The proof of the lemma is straightforward [17].

This is an important result that allows us to bound the burstiness of session-i traffic at each point in the network. Notice that the increase in burstiness that a session may see in the network is proportional to the sum of the latencies of the servers it has traversed. Therefore, even a session with no burstiness at the input of the network may accumulate a significant amount of burstiness in the network because of the latency of the servers.

Using Theorem 1 and Lemma 1 it is easy to prove the following lemma that will bound the backlog of session i in each node of the network [17].

Lemma 2: The maximum backlog $Q_i^{(S_k)}(t)$ in the kth node of a session is bounded as

$$Q_i^{(S_k)}(t) \leq \sigma_i + \sum_{j=1}^{k} \Theta_i^{(S_j)}.$$

To derive tighter bounds for session delay in a network of $\mathcal{LR}$ servers, we first show that the maximum end-to-end delay in a network of two $\mathcal{LR}$-servers in series is the same as that in a single $\mathcal{LR}$ server whose latency is the sum of the latencies of the two servers it replaces. This result allows an arbitrary number of $\mathcal{LR}$ servers in the path of a session to be modeled by combining their individual latencies.

Analyzing two $\mathcal{LR}$ servers in series introduces a difficulty: If the first server has non-zero latency, the busy period of a session in the second server may not coincide with the corresponding busy period in the first server. That is, a packet that started a busy period in the first server may not start a busy period in the second, but instead might arrive while a busy period is in progress at the second server. Also, since the actual service rate seen by session i in the first server can be more than ρ_i, a single busy period in the first server may result in multiple busy periods in the second server. This is illustrated in Figure 3. We will take these effects into account in our analysis of a network of $\mathcal{LR}$ servers.

In the following, we will use the term *network busy period* to mean a busy period of the session being analyzed in the first server on its path. Similarly, when we refer to *service offered by the network* to session i during an interval of time, we mean the amount of session-i traffic transmitted by the last server on the path during the interval under consideration. We will use $W_{i,j}(\tau, t)$ to denote the amount of service offered by the network to session i during an interval $(\tau, t]$ within its jth network busy period. Also, we will use $W_{i,j}^1(\tau_1, t_1)$ to denote the amount of service offered by the first server during an interval (τ_1, t_1) within its local busy period, and $W_{i,j}^2(\tau_2, t_2)$ the same parameter for the second server during an interval (τ_2, t_2) within its busy period.

We first prove the following lemma to bound the service provided by the network to a session during an interval within a network busy period.

Lemma 3: Consider a network of two $\mathcal{LR}$-servers S_1 and S_2 in series, with latencies $\Theta_i^{(S_1)}$ and $\Theta_i^{(S_2)}$, respectively. If ρ_i is the rate allocated to session i in the network, the service offered by the network to the packets of the jth network busy period of that session during an interval $(\tau, t]$ within the busy period is bounded as

$$W_{i,j}(\tau, t) \geq \max\left(0, \rho_i(t - \tau - (\Theta_i^{(S_1)} + \Theta_i^{(S_2)}))\right).$$

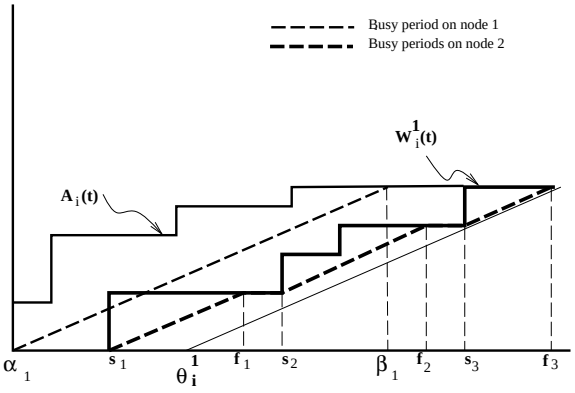

Fig. 3. Illustration of busy periods of a session in two servers in series. The network busy period for session i in this case is split into multiple busy periods in the second server. The busy period in the first server is $(\alpha_1, \beta_1]$. The packets arriving at the second server from this busy period form multiple busy periods $(s_1, f_1], (s_2, f_2], (s_3, f_3]$ in the second server. The line with slope ρ_i that starts at Θ_i^1 bounds all these busy periods.

The proof is omitted due to lack of space. The interested reader is referred to [17] for more details. Lemma 3 asserts that the service offered to a session in a network of two $\mathcal{LR}$ servers in series is no less in the worst case than the service offered to the session by a single $\mathcal{LR}$ server whose latency is equal to the sum of the latencies of the two servers it replaces. Since we make no assumptions on the maximum service offered by the servers to a session, we can merge an arbitrary number of servers connected in series to estimate the service offered after the kth node. We can therefore state the following corollary.

Corollary 1: Let τ be the start of the jth network busy period of session i in a network of $\mathcal{LR}$ servers. If ρ_i is the minimum bandwidth allocated to session i in the network, the service offered to packets of the jth network busy period after the kth node in the network is given by

$$W_{i,j}^{S_k}(\tau, t) \geq \max\left(0, \rho_i(t - \tau - \sum_{j=1}^{k} \Theta_i^{(S_j)})\right),$$

where $\Theta_i^{(S_j)}$ is the latency of the jth server in the network for session i.

Using the above corollary we can bound the end-to-end delays of session i if the input traffic is leaky-bucket shaped and the average arrival rate is less than ρ_i. In [17] we prove the following theorem:

Theorem 2: The maximum delay D_i of a session i in a network of $\mathcal{LR}$ servers, consisting of k servers in series, is bounded as

$$D_i \leq \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{k} \Theta_i^{(S_j)}, \tag{3.4}$$

where $\Theta_i^{(S_j)}$ is the latency of the jth server in the network for session i.

This maximum delay is independent of the topology of the network. The bound is also much tighter than what could be obtained by analyzing each server in isolation. Note that the end-to-end delay bound is a function of only two parameters: the burstiness of the session traffic at the source and the latencies of the individual servers on the path of the session. Since we assumed only that each of the servers in the network belongs to the $\mathcal{LR}$ class, these results are more general than the delay bounds due to Parekh and Gallager [13]. In the next section, we will show that all well-known work-conserving schedulers are in fact $\mathcal{LR}$ servers. Thus, our delay bound applies to almost any network of schedulers.

The delay bound in Eq. (3.4) shows that there are two ways to minimize delays and buffer requirements in a network of $\mathcal{LR}$ servers: i) allocate more bandwidth to a session, thus reducing the term σ_i/ρ_i, or ii) use $\mathcal{LR}$ servers with low latencies. Since the latency is accumulated through multiple nodes, the second approach is preferred in a large network. The first approach reduces the utilization of the network, thus allowing only a smaller number of simultaneous sessions to be supported than would be possible with minimum-latency servers. Minimizing the latency also minimizes the buffer requirements of the session at the individual servers in the network.

Proposition 1: The end-to-end delay and increase in burstiness of a session in a network of $\mathcal{LR}$ servers is proportional to the latency Θ_i^S of the servers. We can minimize both of these parameters by designing servers with minimum latency.

Note that the latency of a server depends, in general, on its internal parameters and the bandwidth allocation of the session under consideration. In addition, the latency may also vary with the number of active sessions and their allocations. Such a dependence of the latency of one session on other sessions indicates the poor isolation properties of the scheduler. Likewise, in some schedulers the latency may depend heavily on its internal parameters, and less on the bandwidth allocation of the session under observation. Such schedulers do not allow us to control the latency of a session by controlling its bandwidth allocation. On the other hand, the latency of a PGPS server depends heavily on the allocated bandwidth of the session under consideration. This flexibility is greatly desirable.

Since the definition of an $\mathcal{LR}$ server is not based on any assumptions on the input traffic, it is easy to derive delay bounds for traffic distributions other than the (σ, ρ) model. For example, when the peak rate of the source is known, a modified upper bound on the delay of an $\mathcal{LR}$ server can be obtained. Let us denote with g_i the service rate allocated to connection i, and let ρ_i and P_i respectively denote the average and peak rate at the source of connection i. The arrivals at the input of the server during the interval $(\tau, t]$ now satisfy the inequality

$$A_i(\tau, t) \leq \min\left(\sigma_i + \rho_i(t-\tau), P_i(t-\tau)\right). \tag{3.5}$$

In [17] we prove the following lemma:

Lemma 4: The maximum delay D_i of a session i in a network of $\mathcal{LR}$ servers, consisting of k servers in series, where the peak rate of the source is known, is bounded as

$$D_i^S \leq \left(\frac{P_i - g_i}{g_i}\right)\left(\frac{\sigma_i}{P_i - \rho_i}\right) + \sum_{j=1}^{k} \Theta_i^{(S_j)}. \tag{3.6}$$

where $\Theta_i^{(S_j)}$ is the latency of the jth server in the network for session i.

IV. Schedulers in the Class $\mathcal{LR}$

In this section we will show that several well-known work-conserving schedulers belong to the class $\mathcal{LR}$ and determine their latencies. Recall that our definition of $\mathcal{LR}$ servers in the previous section is based on session-busy periods. In practice, however, it is easier to analyze scheduling algorithms based on session backlogged periods. The following lemma enables the latency of an $\mathcal{LR}$ server to be estimated based on its behavior in the session backlogged periods. We will use this as a tool in our analysis of several schedulers in this section.

Lemma 5: Let $(s_j, t_j]$ indicate an interval of time in which session i is continuously backlogged in server $\mathcal{S}$. If the service offered to the packets that arrived in the interval $(s_j, t_j]$ can be bounded at every instant t, $s_j < t \leq t_j$ as

$$W_i(s_j, t) \geq \max(0, \rho_i(t - s_j - \Theta_i)),$$

then $\mathcal{S}$ is an $\mathcal{LR}$ server with a latency less than or equal to Θ_i.

A formal proof of the Lemma is presented [17]. This lemma will allow us to estimate the latency of an $\mathcal{LR}$-server. However, it does not necessarily provide us a tight bound for the parameter Θ_i.

A main contribution of the theory of $\mathcal{LR}$-servers is the notion of the busy period. The bound on the service offered by an $\mathcal{LR}$-server is based on the busy period. This is a more general approach than bounding the service offered by the server based on the concept of the *backlogged period*. An approach based on the latter was proposed in [18] for providing QoS guarantees. The model bounds the service offered to a connection during one or more backlogged periods, thus providing a means to design a class of scheduling algorithms that can provide specific end-to-end delay guarantees. Using the concept of busy period instead of backlogged period in this model will likely result in tighter end-to-end delay bounds and a larger class of schedulers that can provide these delay bounds.

In Lemma 5 we proved that, if we use the backlogged period to bound the service offered by a server $\mathcal{S}$, then server $\mathcal{S}$is an $\mathcal{LR}$ server and its latency can not be larger than that found for the backlogged period. However, we must emphasize the fact that the opposite is not true. Consider the example of Figure 4. Le us assume an $\mathcal{LR}$-server with rate ρ and latency Θ. Referring to Figure 4, time-intervals $(0, t_1]$ and $(t_2, t_3]$ form two busy periods. However, the server remains backlogged during the whole interval $(t_1, t3]$. If the backlogged period was used to bound the service offered by

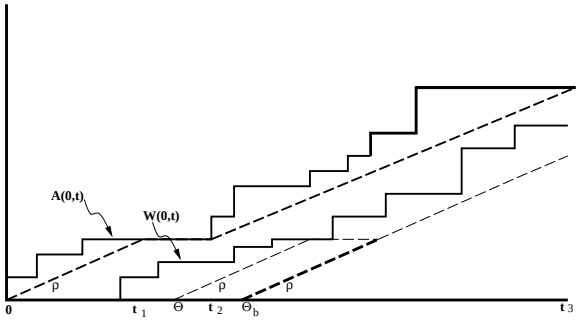

Fig. 4. Difference in bounding the service based on the backlogged or busy periods.

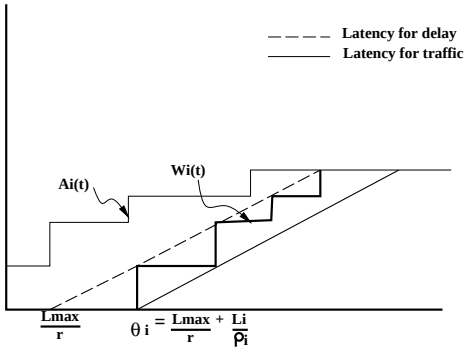

Fig. 5. Illustration of the two envelopes used to bound the service received by session i in a session-busy period. Each step in the arrival function indicates a new packet. The lower envelope is a valid lower-bound for the service at any point in the busy period, while the upper one is valid only at the points when a packet leaves the system.

the server, a latency $\Theta_b > \Theta$ would result. By repeating the above example over multiple busy periods, it is easy to verify that Θ_b can not be bounded. This shows that if backlogged period was used instead of busy period in the definition of the $\mathcal{LR}$ server model, the end-to-end delays of the server would not be bounded.

By using Lemma 5 as our tool, we analyzed several work conserving servers and proved that they belong in the class $\mathcal{LR}$. A summary of our results is presented in Table I. It is easy to see that PGPS and VirtualClock have the lowest latency among all the servers. In addition, their latency does not depend on the number of connections sharing the same outgoing link. As we will show in Section VI, however, VirtualClock is not a fair algorithm.

In self-clocked fair queueing, the latency is a linear function of the maximum number of connections sharing the outgoing link. In Deficit Round Robin [8], the latency depends on the frame size F. By the definition of the algorithm, the frame size, in turn, is determined by the granularity of the bandwidth allocation and the maximum packet size of its session. That is,

$$\sum_{i=1}^{V} L_i \leq F,$$

where L_i is the maximum packet-size of session i. Thus, the latency in Deficit Round Robin is also a function of the number of connections that share the outgoing link. Weighted Round Robin can be thought of as a special case of Deficit Round Robin and its latency is again a function of the maximum number of connections sharing the outgoing link.

V. An Improved Delay Bound

The latencies of $\mathcal{LR}$ servers computed in the last section are based on the assumption that a packet is considered serviced only when its last bit has left the server. Thus, the latency Θ_i^S was computed such that the service performed during a session busy period at time t, $W_{i,j}(\tau, t)$, is always greater than or equal to $\rho_i(t - \tau - \Theta_i^S)$. Since the maximum difference between $W_{i,j}(\tau, t)$ and $\rho_i(t - \tau)$ occurs just before a session-i packet departs the system, the latency Θ_i^S is calculated at such points. This is necessary to be able to bound the arrivals to the next server in a chain of servers; since our servers are not cut-through devices, a packet can be serviced only after its last bit has arrived. Our assumption that the packet leaves as an impulse from the server allows us to model the arrival of the packet in the next server as an impulse as well.

When we compute the end-to-end delay of a session, however, we are only interested in finding the time at which the last bit of a packet leaves the last server. Thus, for the last server in a chain, we can determine the latency Θ_i^S based only on the instants of time just after a packet was serviced from the session. This results in a lower value of latency and, consequently, a tighter bound on the end-to-delay in a network of servers than that given by eq. (3.4).

To apply this idea, the analysis of the network is separated into two steps. If the session passes through k hops, we bound the service offered to the session in the first $k-1$ servers considering arbitrary instants during session-busy periods. On the last node, however, we calculate the latency based only on the points just after a packet completes service.

This idea is best illustrated by an example in the case of the PGPS server. Assume that a busy period starts at time τ, and that a packet leaves the PGPS server at time t_k. Then, on the corresponding GPS server, this packet left at time $t_k - L_{max}/r$ or later. Therefore, if we consider only such points t_k, we can write

$$\begin{aligned} W_{i,j}^P(\tau, t_k) &\geq W_{i,j}^F(\tau, t_k - \frac{L_{max}}{r}) \\ &\geq \rho_i(t_k - \tau - \frac{L_{max}}{r}). \end{aligned}$$

This results in a latency of L_{max}/r as compared to $(L_i/\rho_i + L_{max}/r)$ presented in Table I.

Figure 5 shows the two envelopes based on bounding the service received by the session in the two different ways. The lower envelope applies to arbitrary points in the session-busy period, while the upper envelope is valid only at points when a packet leaves the system. For computing end-to-end delay bounds in a network of servers, we can use the upper envelope in the last server. In all the work-conserving schedulers we have studied, the two envelopes are apart by L_i/ρ_i, where L_i is the maximum packet-size for session i.

Therefore, for these $\mathcal{LR}$ servers, we can obtain an improved bound for the end-to-end delay in a network by subtracting L_i/ρ_i from eq. (3.4). Therefore,

$$D_i \leq \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{k} \Theta_i^{(S_j)} - \frac{L_i}{\rho_i}. \tag{5.1}$$

If we substitute the latency obtained for PGPS from Table I, that is, $\Theta_i^{(S_j)} = L_i/\rho_i + L_{max}/r$, we get

$$D_i \leq \frac{\sigma_i}{\rho_i} + (k-1)\frac{L_i}{\rho_i} + k\frac{L_{max}}{r}, \tag{5.2}$$

which agrees with the bound obtained by Parekh and Gallager [13] for a network of PGPS servers. Since the latencies of PGPS and VirtualClock are identical, the bound of (5.2) applies to VirtualClock as well; this is also in agreement with the results of Lam and Xie [19].

While we have verified that this improvement of L_i/ρ_i in the delay bound is valid for all the $\mathcal{LR}$ servers analyzed in this paper, whether this is true for all $\mathcal{LR}$ servers remains an open question. We have not yet found a formal proof on its validity for arbitrary $\mathcal{LR}$ servers.

VI. Fairness of $\mathcal{LR}$ Servers

In Section III, we showed that the worst-case delay behavior of individual sessions in a network of $\mathcal{LR}$ servers can be analyzed knowing only their latencies. However, the latency of an $\mathcal{LR}$ server, by itself, provides no indication of its fairness. For example, VirtualClock and PGPS are two different $\mathcal{LR}$ servers with the same latency, but with substantially different fairness characteristics. In this section we analyze the fairness characteristics of several well-known LR servers and compare them.

The fairness parameter that we use is based on the definition presented by Golestani [5] for analysis of self-clocked fair queueing. Let us assume that $W_i^{\mathcal{S}}(\tau, t)$ is the service offered to connection i in the interval $(\tau, t]$ by server $\mathcal{S}$. If ρ_i is the bandwidth allocated to connection i, we will call the fraction $W_i^{\mathcal{S}}(\tau, t)/\rho_i$ the *normalized service* offered to connection i in the interval $(\tau, t]$. A scheduler is perfectly fair if the difference in normalized service offered to any two connections that are continuously backlogged in the system in the interval $(\tau, t]$ is zero. That is,

$$\left| \frac{W_i^{\mathcal{S}}(\tau, t)}{\rho_i} - \frac{W_i^{\mathcal{S}}(\tau, t)}{\rho_j} \right| = 0.$$

GPS multiplexing is proven to have this property. However, this condition cannot be met by any packet-by-packet algorithm since packets must be serviced exclusively. Therefore, in a packet by packet server, we can only require that the difference in normalized service received by the connections be bounded by a constant.

Golestani suggested use of the difference in normalized service offered to any two connections as the measure of fairness for the algorithm [5]. More precisely, an algorithm is considered close to fair if, for any two connections i, j that are continuously backlogged in an interval of time $(t_1, t_2]$,

$$\left| \frac{W_i^{\mathcal{S}}(t_1, t_2)}{\rho_i} - \frac{W_j^{\mathcal{S}}(t_1, t_2)}{\rho_j} \right| \leq \mathcal{F}^{\mathcal{S}},$$

where $\mathcal{F}^{\mathcal{S}}$ is a constant. Let us call $\mathcal{F}^{\mathcal{S}}$ as the *fairness* of server $\mathcal{S}$. A difficulty arises, however, in the use of the above definition in comparing the fairness of different schedulers. For the same pattern of session arrivals, the backlogged periods of the session can vary across schedulers; a comparison of fairness of different scheduling algorithms can therefore yield misleading results if the arrival pattern is not chosen so as to produce the same backlogged periods in all the schedulers. Hence, we modify Golestani's definition slightly. We consider a time τ at which the connections i and j being compared have an infinite supply of packets. This forces them to be continuously backlogged in the servers, regardless of the scheduling algorithm used. We use as a measure of fairness the difference in normalized service offered to the two connections for any time interval $(t_1, t_2]$ after time τ. A typical example of unfairness occurs in the VirtualClock algorithm, as presented in [4].

We evaluated the fairness of a number of servers. We use L_i to denote the maximum packet-size for session i and L_{max} the maximum packet-size over all sessions. A summary of our results is presented in Table I. Detailed proofs for these results are presented in [17], [20].

We note that all the algorithms studied, except VirtualClock, can be considered as fair based on our definition of fairness. However, it is interesting to note that self-clocked fair queueing (SCFQ) has the best fairness among all the packet-by-packet schedulers, even better than that of PGPS in some cases. On the other hand, the latency of an SCFQ server can be much higher than that of a PGPS server; this is because SCFQ may delay service to connections when they become backlogged after an idle period, while PGPS penalizes the connections that have already received their allocated bandwidth to serve newly backlogged connections.

VII. Conclusions

In Table I, we have summarized the characteristics of several scheduling algorithms belonging to the $\mathcal{LR}$ class based on the three parameters discussed in Section 2. Based on this summary, it is easy to see that the PGPS scheduler has the best performance both in terms of latency and fairness properties. However, it also has the highest implementation complexity. VirtualClock has latency identical to that of PGPS, but is not a fair algorithm.

All the other algorithms studied have bounded unfairness, but also have much higher latencies than PGPS. From our analysis of networks of $\mathcal{LR}$ servers, it becomes clear how this increased latency leads to high end-to-end delay bounds, large buffer requirements in the switch nodes, and increased traffic burstiness inside the network. Even with constant-bit-rate traffic at the source, sessions may accumulate considerable burstiness after many hops through the network if the the servers have high latencies. Thus, the use of servers with minimum latency is extremely important in a broadband packet network. In both the SCFQ server and the

Server	Latency	Fairness	Complexity
PGPS	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$	$\max(C_j + \frac{L_{max}}{\rho_i} + \frac{L_j}{\rho_j}, C_i + \frac{L_{max}}{\rho_j} + \frac{L_i}{\rho_i})$, where $C_i = \min((V-1)\frac{L_{max}}{\rho_i}, \max_{1 \le n \le V}(\frac{L_n}{\rho_n}))$.	$O(V)$
SCFQ	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}(V-1)$	$\frac{L_i}{\rho_i} + \frac{L_j}{\rho_j}$	$O(\log V)$
VirtualClock	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$	∞	$O(\log V)$
Deficit Round Robin	$\frac{(3F-2\phi_i)}{r}$	$\frac{3F}{r}$	$O(1)$
Weighted Round Robin	$\frac{(F-\phi_i+L_c)}{r}$	$\frac{F}{r}$	$O(1)$
Frame-Based Fair Queueing (FFQ) [20]	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$	$\frac{2F}{r} + \max(\frac{L_i}{\rho_i}, \frac{L_j}{\rho_j})$	$O(\log V)$

TABLE I. Latency, fairness and implementation complexity of several work-conserving servers. L_i is the maximum packet size of session i and L_{max} the maximum packet size among all the sessions. C_i is the maximum normalized service that a session may receive in a PGPS server in excess of that in the GPS server. In weighted round-robin and deficit round-robin, F is the frame size and ϕ_i is the amount of traffic in the frame allocated to session i. L_c is the size of the fixed packet (cell) in weighted round-robin.

round-robin schedulers, the latency and fairness are greatly affected by the number of connections sharing a common outgoing link. This property makes it difficult to control end-to-end session delays in networks where a large number of flows may share the links.

Our comparison of schedulers along the three dimensions leaves open the question whether a scheduling algorithm can be designed that has the same low latency as that of PGPS, bounded unfairness, and an efficient implementation. In [20], we extend this work by presenting such a scheduling discipline that we call Frame-based Fair Queueing (FFQ). FFQ is a sorted-priority algorithm in which the calculation of timestamps can be performed in $O(1)$ time. The latency of frame-based fair queueing is identical to that of a PGPS server, yet the algorithm can be shown to be fair based on the criterion we used in this paper to evaluate fairness.

References

[1] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," in *Proc. ACM SIGCOMM '92*, pp. 14–26, August 1992.

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

[3] 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.

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

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

[6] 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.

[7] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1265–79, October 1991.

[8] M. Shreedhar and G. Varghese, "Efficient Fair Queueing using Deficit Round Robin," in *Proc. SIGCOMM'95*, September 1995.

[9] C. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very high-speed networks," in *IEEE Global Telecommunications Conference*, pp. 300.3.1–300.3.9, December 1990.

[10] R. Cruz, "A calculus for network delay. I. Network elements in isolation.," *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, January 1991.

[11] R. Cruz, "A calculus for network delay. II. Network elements in isolation.," *IEEE Transactions on Information Theory*, vol. 37, pp. 132–141, January 1991.

[12] H. Zhang, *Service Disciplines for Packet-Switching Integrated-Services Networks*. PhD thesis, U.C. Berkeley, 1992.

[13] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," in *Proc. INFOCOM '93*, vol. 2, pp. 521–530, March 1993.

[14] P. Goyal, S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 287–298, April 1995.

[15] 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.

[16] H. Zhang and S. Keshav, "Comparison of rate based service disciplines," in *Proc. ACM SIGCOMM '91*, pp. 113–122, 1991.

[17] D. Stiliadis and A. Varma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," Tech. Rep. UCSC-CRL-95-38, U.C. Santa Cruz, July 1995, http://www.cse.ucsc.edu/research/hsnlab/publications/.

[18] R. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal on Selected Areas In Communications*, vol. 13, pp. 1048–1056, August 1995.

[19] S. Lam and G. Xie, "Burst scheduling: Architecture and algorithm for switching packet video," in *INFOCOM'95*, April 1995.

[20] D. Stiliadis and A. Varma, "Frame-based fair queueing: A new traffic scheduling algorithm for packet-switched networks," Tech. Rep. UCSC-CRL-95-39, U.C. Santa Cruz, July 1995, http://www.cse.ucsc.edu/research/hsnlab/publications/.