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

Dimitrios Stiliadis
Anujan Varma

UCSC-CRL-95-38
July 18, 1995

Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

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.

Keywords: Traffic scheduling, ATM switch scheduling, fair queueing, end-to-end delay bounds, fairness.

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

1 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]. Rigid real-time applications require a guaranteed portion of the link bandwidth, as well as bounded end-to-end delay, low delay jitter, and low packet loss rate. Adaptive applications can adjust their behavior to the current network state, as long as they can receive a specific portion of the link bandwidth over a period of time; that is, they can receive the guaranteed bandwidth over an averaging period longer than that of rigid real-time applications. Finally, most of the data traffic is transmitted in "best-effort" mode, requiring no bandwidth guarantees. The instantaneous bandwidth left over after allocating the bandwidth to real-time flows can be used to transmit packets belonging to best-effort traffic.

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. Implementation of the algorithm may be in hardware or software. Because of the small cell-size in an ATM (Asynchronous Transfer Mode) network, the scheduling algorithm must usually be implemented in hardware in an ATM switch. In a packet network with larger packet-sizes, such as the current Internet, the algorithm can be implemented in software.

Several service disciplines are known in the literature for bandwidth allocation and transmission scheduling in output-buffered switches. FIFO scheduling is perhaps the simplest to implement, but does not provide any isolation between individual sessions that is necessary to achieve deterministic bandwidth guarantees. 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, 10]. 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).

In general, schedulers can be characterized as *work-conserving* or *non-work-conserving*. A scheduler is work-conserving if the server is never idle when a packet is buffered in the system. A non-work-conserving server may remain idle even if there are available packets to transmit. A server may, for example, postpone the transmission of a packet when it expects a higher-priority packet to arrive soon, even though it is currently idle. When the transmission time of a packet is short, as is typically the case in an ATM network, however, such a policy is seldom justified. Non-work-conserving algorithms are also used to control delay jitter by delaying packets that arrive early [11]. Work-conserving servers always have lower average delays than non-work-conserving servers. Examples of work-conserving schedulers include Generalized Processor Sharing (GPS) [4], Weighted Fair Queueing [3], VirtualClock [2], Delay-Earliest-Due-Date (Delay-EDD) [6], Weighted Round Robin [7], and Deficit Round Robin [8]. On the other hand, Hierarchical-Round-Robin (HRR) [9], Stop-and-Go queueing [10], and Jitter-Earliest-Due-Date [11] are non-work-conserving schedulers.

Another classification of schedulers is based on their internal structure [12]. According to this classification there are two main architectures: *sorted-priority* and *frame-based*. In a sorted-priority scheduler, there is a global variable — usually referred to as the *virtual time* — associated with each outgoing link of the switch. Each time a packet arrives or gets serviced, this variable is updated. A timestamp, computed as a function of this variable, is associated with each packet in the system. Packets are sorted based on their timestamps, and are transmitted in that order. VirtualClock, Weighted Fair Queueing, and Delay-EDD follow this architecture.

In a frame-based scheduler, time is split into frames of fixed or variable length. Reservations of sessions are made in terms of the maximum amount of traffic the session is allowed to transmit during a frame period. Hierarchical Round Robin and Stop-and-Go Queueing are frame-based schedulers that use a constant frame size. As a result, the server may remain idle if sessions transmit less traffic than their reservations over the duration of a frame. In contrast, Weighted Round Robin and Deficit Round Robin schedulers allow the frame size to vary within a maximum. Thus, if the traffic from a session is less than its reservation, a new frame can be started early. Therefore, both of these schedulers are work-conserving.

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 session is related to the work of Cruz [13, 14], Zhang [15], and Parekh and Gallager [4, 16]. Cruz [13, 14] 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.

Parekh and Gallager analyzed the worst-case behavior of sessions in a network of GPS schedulers [4, 16] and derived upper bounds on end-to-end delay and internal burstiness of sessions. However, the analysis applies to a homogeneous network consisting of only GPS schedulers. Our analysis accommodates a broad range of scheduling algorithms and the ability to combine the schedulers in arbitrary ways in a network.

Zhang [15] derived end-to-end delay bounds for a class of non-work-conserving scheduling algorithms when traffic is re-shaped at each node of the network. This allows the delays of individual schedulers on the path to be accumulated in a simple manner. Our approach differs from this work in that we consider the broader class of work-conserving schedulers in our analysis, and we do not assume any traffic re-shaping mechanisms within the network.

Another model for delay-analysis based on a class of guaranteed-rate servers was presented in [17]. 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 [18]. 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.

The rest of this paper is organized as follows: In Section 2, we discuss the necessary properties that a scheduling algorithm must satisfy for application in a real network. In Section 3, we define the class of *Latency-Rate* ($\mathcal{LR}$) servers and derive upper bounds on the end-to-end delays, buffer requirements, and traffic burstiness of individual sessions in an arbitrary-topology network of $\mathcal{LR}$ servers when the session traffic is shaped by a leaky bucket. In Section 4 we prove that several well-known scheduling algorithms, such as VirtualClock, Packet-by-Packet Generalized Processor Sharing (PGPS), Self-Clocked Fair Queueing (SCFQ), Weighted-Round-Robin (WRR), and Deficit-Round-Robin (DRR) all belong to the class of $\mathcal{LR}$ servers, and derive their latencies. In Section 5, we derive a slightly improved upper bound on end-to-end delay in a network of $\mathcal{LR}$ servers by bounding the service of the last server on the path more tightly. In Section 6, we analyze the fairness properties of these algorithms. Finally, Section 7 presents some conclusions and directions for future work. Appendix A contains the proofs of many lemmas and theorems presented in the paper.

2 A Common Framework for Scheduling Algorithms

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.

2.1 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 [19]. Several previous studies have used this traffic model [4, 13, 14, 16]. 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. Note that isolation is necessary even when policing mechanisms are used to shape all the flows at the entry point of the network, as the flows may accumulate burstiness within the network.
2. Delay bounds that are independent of the number of sessions sharing the outgoing link: This is necessary if the algorithm is to be used in switches supporting a large number of flows.
3. Ability to control the delay bound of a session by controlling only its bandwidth reservation: This property of the algorithm provides significant flexibility in trading off session delays with their bandwidth allocations.

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.

2.2 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.

Based only on the end-to-end delay bounds and fairness properties, Generalized-Processor-Sharing (GPS) is an ideal scheduling discipline [4]. GPS multiplexing is defined with respect to a fluid-model, where packets are considered to be infinitely divisible. The share of bandwidth reserved by session i is represented by a real number ϕ_i. Let $B(\tau, t)$ be the set of connections that are backlogged in the interval $(\tau, t]$. If r is the rate of the server, the service $W_i(\tau, t)$ offered to a connection i that belongs in $B(\tau, t)$ is proportional to ϕ_i. That is,

$$W_i(\tau, t) \geq \frac{\phi_i}{\sum_{j \in B(\tau,t)} \phi_j} r(t - \tau).$$

The minimum service that a connection can receive in any interval of time is

$$\frac{\phi_i}{\sum_{j=1}^{V} \phi_j} r(t - \tau),$$

where V is the maximum number of connections that can be backlogged in the server at the same time. Thus, GPS serves each backlogged session with a minimum rate equal to its reserved rate at each instant; in addition, the excess bandwidth available from sessions not using their reservations is distributed among all the backlogged connections at each instant in proportion to their individual reservations. This results in perfect isolation, ideal fairness, and low end-to-end session delays.

The VirtualClock algorithm, in contrast, does not bound the difference in service received by two backlogged sessions over an interval that is smaller than the backlogged period. This is the result of the scheduler performing an averaging process on the rate of service provided to individual sessions. In VirtualClock, the averaging interval can be arbitrarily long. The GPS scheduler, on the other hand, occupies the opposite extreme where no memory of past bandwidth usage of sessions is maintained. Note that, according to our definition of fairness, some amount of short-term unfairness between sessions is inevitable in any packet-level scheduler, since each packet must be serviced exclusively. In practice, we can only require that the difference in normalized service received by two sessions be bounded by a constant.

2.3 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 [12]:

1. Calculation of the timestamp: The PGPS scheduler has the highest complexity in this respect, since a GPS scheduler must be simulated in parallel in order to update the virtual time. This simulation may result in a process overhead of $O(V)$ per packet transmission in the worst-case. On the other hand, in both VirtualClock and self-clocked fair queueing the timestamp calculation involves only a constant number of computations, resulting in a worst-case complexity of $O(1)$.
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 [20].

The last two operations are identical for any sorted-priority architecture. A parallel implementation of these operations with $O(1)$ time complexity by using a set of $O(V)$ simple processing elements has been shown [21, 22].

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.

3 $\mathcal{LR}$-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.

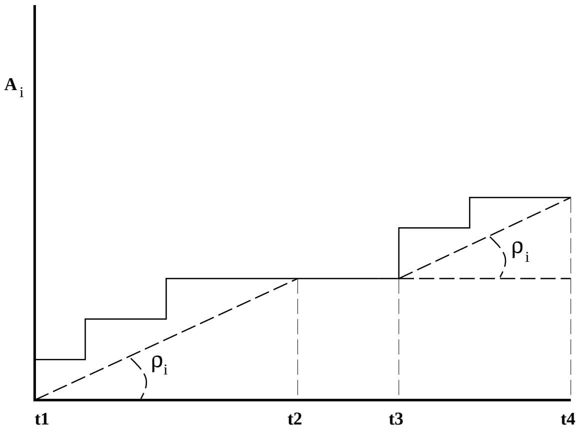

Figure 3.1: Intervals $(t_1, t_2]$ and $(t_3, t_4]$ are two different busy periods.

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, both $A_i(\tau, t)$ and $W_i(\tau, t)$ are continuous functions 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 $Q_i(t) > 0$.

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,*

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

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 3.1). 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

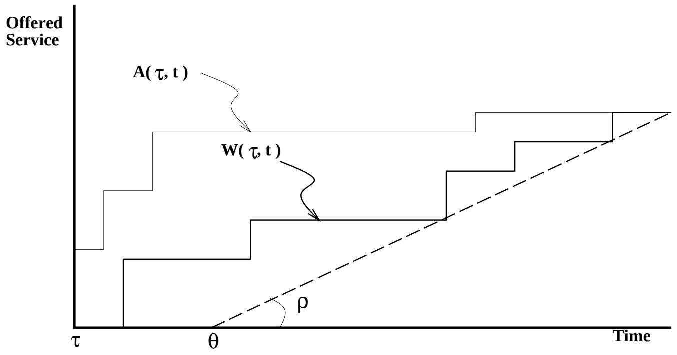

Figure 3.2: An example of the behavior of an $\mathcal{LR}$ scheduler.

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.

For a given session-i backlogged period, the corresponding busy period can be longer. In addition, if $(s_1, f_1]$ is a busy period for session i, multiple backlogged periods may occur in the actual system during the interval $(s_1, f_1]$. The beginning of a busy period of session i always marks the beginning of a backlogged period, but the converse is not always true. In addition, the beginning of a busy period is always caused by the arrival of a packet into the system. A busy period cannot end before the backlogged period that started it.

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 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,*

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

$\Theta_i^{\mathcal{S}}$ 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 3.2). It is easy to observe that the latency $\Theta_i^{\mathcal{S}}$ 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 $\Theta_i^{\mathcal{S}}$ 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. However, we will now 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.

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 present 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.

3.1 Analysis of a Single $\mathcal{LR}$ Server

Assume a set of V sessions sharing the same output link of an $\mathcal{LR}$ server. First, we show that the packets of a busy period in the server complete service no later than $\Theta_i^{\mathcal{S}}$ after the busy period ends.

Lemma 1: *Let $(t_1, t_2]$ be a session-i busy period. All packets that arrived from session i during the busy period will be serviced by time $t_2 + \Theta_i^{\mathcal{S}}$.*

Proof: Let us assume that the last packet of the jth busy period completes service at time t. Then at t,

$$A_i(t_1, t_2) = W_{i,j}^{\mathcal{S}}(t_1, t). \tag{3.2}$$

But we know from the definition of the busy period that

$$A_i(t_1, t_2) = \rho_i(t_2 - t_1). \tag{3.3}$$

From the definition of $\mathcal{LR}$ servers and equations (3.2) and (3.3),

$$\rho_i(t - t_1 - \Theta_i^{\mathcal{S}}) - \rho_i(t_2 - t_1) \leq 0 \tag{3.4}$$

or equivalently,

$$t \leq t_2 + \Theta_i^{\mathcal{S}}. \tag{3.5}$$

□

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 $Q_i^{\mathcal{S}}(t)$ is the backlog of session i at time t, then

$$Q_i^{\mathcal{S}}(t) \leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \tag{3.6}$$

2. If $D_i^{\mathcal{S}}$ is the delay of any packet of session i in server S,

$$D_i^{\mathcal{S}} \leq \frac{\sigma_i}{\rho_i} + \Theta_i^{\mathcal{S}}. \tag{3.7}$$

3. The output traffic conforms to the leaky bucket model with parameters $(\sigma_i + \Theta_i^{\mathcal{S}} \rho_i, \rho_i)$.

Proof: We will prove each of the above statements separately.

1. Upper bound on session-i backlog: Assume that the jth busy period covered the time interval $(s_j, f_j]$. Let us denote with $W_{i,j}^{\mathcal{S}}(\tau, t)$ the service offered to the packets of the jth busy period of session i in the interval $(\tau, t]$, with $\tau \geq s_j$ and $t \leq f_j + \Theta_i^{\mathcal{S}}$. We will denote by $W_i^{\mathcal{S}}(\tau, t)$ the total service offered to session i during the same interval of time. Note that $W_i^{\mathcal{S}}(\tau, t) \geq W_{i,j}^{\mathcal{S}}(\tau, t)$, since, during the interval $(\tau, t]$ the server may transmit packets of a previous busy period as well.

During a system busy period, let us denote with τ_j the time at which the last packet of the jth busy period completes service. At this point of time all backlogged packets belong to busy periods that started after time f_j. Note also that $\tau_j \leq f_j + \Theta_i^{\mathcal{S}}$. We will prove the theorem by induction over the time intervals $(\tau_{j-1}, \tau_j]$.

Base step: Assume that a session i busy period started at time $s_1 = \tau_0$. Since this is the first busy period for session i, there are no other packets backlogged from session i at τ_0. By the definition of $\mathcal{LR}$-servers, we know that for every time t with $\tau_0 \leq t \leq \tau_1$,

$$W_{i,1}^{\mathcal{S}}(\tau_0, t) = W_{i,1}^{\mathcal{S}}(s_1, t) \geq \max(0, \rho_i(t - s_1 - \Theta_i^{\mathcal{S}})).$$

From the definition of leaky bucket,

$$A_i(s_1, t) \leq \sigma_i + \rho_i(t - s_1).$$

The backlog at time t is defined as the total amount of traffic that arrived since the beginning of the system busy period minus those packets that were serviced. Therefore,

$$Q_i(t) = A_i(s_1, t) - W_{i,1}^{\mathcal{S}}(s_1, t). \tag{3.8}$$

We will consider two cases for t:
Case 1: If $t < s_1 + \Theta_i^{\mathcal{S}}$ then,

$$W_{i,1}^{\mathcal{S}}(s_1, t) \geq 0.$$

Therefore,

$$\begin{aligned} Q_i^{\mathcal{S}}(t) &\leq A_i(s_1, t) \\ &\leq \sigma_i + \rho_i(t - s_1) \\ &\leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \end{aligned} \tag{3.9}$$

The first inequality follows from the definition of the leaky bucket and the second from the restriction $t < s_1 + \Theta_i^{\mathcal{S}}$.
Case 2: If $t \geq s_1 + \Theta_i^{\mathcal{S}}$, then from the definition of $\mathcal{LR}$-server,

$$W_{i,1}^{\mathcal{S}}(s_1, t) \geq \rho_i(t - s_1 - \Theta_i^{\mathcal{S}}).$$

From equations (3.1) and (3.8),

$$\begin{aligned} Q_i^{\mathcal{S}}(t) &\leq \sigma_i + \rho_i(t - s_1) - \rho_i(t - s_1 - \Theta_i^{\mathcal{S}}) \\ &\leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \end{aligned} \tag{3.10}$$

Inductive step: We will assume that the theorem holds until the end of the nth busy period. That is, for every time $t \leq \tau_n$, $Q_i^{\mathcal{S}}(t) \leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}$. We will now prove that it holds also during the interval $(\tau_n, \tau_{n+1}]$. After time τ_n, only packets from the $(n+1)$th busy period will be serviced, and by time τ_{n+1} all of the packets that belong in the $(n+1)$th busy period will complete service. In addition, no packet from the $(n+1)$th busy period has been serviced before time τ_n. Therefore, for every time t with $\tau_n < t \leq \tau_{n+1}$, we can write

$$W_{i,n+1}^{\mathcal{S}}(\tau_n, t) = W_{i,n+1}^{\mathcal{S}}(s_{n+1}, t).$$

From the definition of the $\mathcal{LR}$-servers,

$$W_{i,n+1}^{\mathcal{S}}(\tau_n, t) \geq \max(0, \rho_i(t - s_{n+1} - \Theta_i^{\mathcal{S}})).$$

The amount of packets that are backlogged in the server is equal to those packets that arrived after the end of the nth busy period minus those packets that were serviced after time τ_n; but we know that , since s_{n+1} is the beginning of the $(n+1)$th busy period, the first packet of the busy period arrived at time s_{n+1}. Therefore,

$$\begin{aligned} Q_i^{\mathcal{S}}(t) &\leq A_i(s_{n+1}, t) - W_{i,n+1}^{\mathcal{S}}(\tau_n, t) \\ &\leq \sigma_i + \rho_i(t - s_{n+1}) - \max(0, \rho_i(t - s_{n+1} - \Theta_i^{\mathcal{S}})) \\ &\leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \end{aligned}$$

2. Upper bound on delay: Let us assume that the maximum delay $D_i^{\mathcal{S}}$ was obtained for a packet that arrived at time t^* during the jth busy period. This means that the packet was serviced at time $t^* + D_i^{\mathcal{S}}$. Hence, the amount of service offered to the session until time $t^* + D_i^{\mathcal{S}}$ is equal to the amount of traffic that arrived from the session until time t. Since s_j is the beginning of the jth busy period,

$$W_{i,j}^{\mathcal{S}}(s_j, t^* + D_i^{\mathcal{S}}) = A_i(s_j, t^*). \tag{3.11}$$

Since the server may provide no service for time $\Theta_i^{\mathcal{S}}$, $D_i^{\mathcal{S}} \geq \Theta_i^{\mathcal{S}}$. From the definition of $\mathcal{LR}$-server,

$$W_{i,j}^{\mathcal{S}}(s_j, t^* + D_i^{\mathcal{S}}) \geq \rho_i(t^* + D_i^{\mathcal{S}} - s_j - \Theta_i^{\mathcal{S}}). \tag{3.12}$$

From (3.11),(3.12), and the leaky bucket constraint (3.1), we have

$$\rho_i(t^* + D_i^{\mathcal{S}} - s_j - \Theta_i^{\mathcal{S}}) \leq \sigma_i + \rho_i(t^* - s_j), \tag{3.13}$$

or equivalently,

$$D_i^{\mathcal{S}} \leq \frac{\sigma_i}{\rho_i} + \Theta_i^{\mathcal{S}}. \tag{3.14}$$

3. Upper bound on burstiness of output traffic: We will now prove that the output traffic of the scheduler conforms to the leaky-bucket model as well. It is sufficient to show that, during any interval $(\tau, t]$ within a session-i busy period,

$$W_i(\tau, t) \leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}} + \rho_i(t - \tau).$$

Let us denote with σ_i^{out} the burstiness of session i at the output of the switch. We will try to find the maximum value for this burstiness. We will assume that the server is work-conserving and that it can service any amount of traffic from a given session without interruption, if the given session is the only active session in the system. Let us denote with $c_i(\tau)$ the amount of tokens that remain in the leaky bucket at time τ. $Q_i^{\mathcal{S}}(\tau)$ is the amount of backlogged packets in the server at time τ. If we assume that we have infinite-capacity input links, at time $\tau+$ it is possible that an amount of $c_i(\tau)$ packets is added to the server queue and no packet is serviced; but we already proved that the maximum backlog of the session i is bounded by $\sigma_i + \rho_i \Theta_i^{\mathcal{S}}$. Therefore,

$$c_i(\tau) + Q_i^{\mathcal{S}}(\tau) \leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \tag{3.15}$$

The arrivals $A_i(\tau, t)$ cannot exceed the amount of tokens in the leaky bucket at time τ plus the amount of tokens that arrived during the interval $(\tau, t]$, minus the amount of tokens at time t. That is,

$$A_i(\tau, t) \leq c_i(\tau) + \rho_i(t - \tau) - c_i(t). \tag{3.16}$$

We can also calculate the service offered to session i in the interval $(\tau, t]$ as

$$\begin{aligned} W_i^{\mathcal{S}}(\tau, t) &= Q_i(\tau) + A_i(\tau, t) - Q_i(t) \\ &\leq Q_i(\tau) + A_i(\tau, t). \end{aligned} \tag{3.17}$$

Then, from (3.16) and (3.17),

$$\begin{aligned} W_i^{\mathcal{S}}(\tau, t) &\leq Q_i(\tau) + c_i(\tau) + \rho_i(t - \tau) - c_i(t) \\ &\leq (\sigma_i + \rho_i \Theta_i^{\mathcal{S}}) + \rho_i(t - \tau). \end{aligned} \tag{3.18}$$

Therefore,

$$\sigma_i^{out} \leq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \tag{3.19}$$

We can show that this bound is tight. If session i is the only active session in the system, it will be explicitly serviced. Let us assume that the maximum backlog for this session is reached at time t. After t, packets arrive from the session with rate ρ_i. The server will first service the backlogged packets and then continue servicing with rate ρ_i. Therefore, we have

$$\sigma_i^{out} \geq \sigma_i + \rho_i \Theta_i^{\mathcal{S}}. \tag{3.20}$$

From Equations (3.19) and (3.20), we can conclude that $\sigma_i^{out} = \sigma_i + \rho_i \Theta_i^{\mathcal{S}}$ □

3.2 Analysis of a Network of $\mathcal{LR}$ Servers

In the previous section we considered a single $\mathcal{LR}$-server and analyzed the delay behavior of a session when the session traffic is leaky-bucket controlled. We will now proceed to prove bounds on backlog and delay over multiple nodes. The straightforward approach would be to accumulate the maximum delays over each node. This approach was used by Cruz [14]. However, this method ignores the correlation between arrivals at two servers in series, and therefore results in very loose bounds. Tighter bounds can be provided by following the approach used by Parekh and Gallager [16] that tries to capture the behavior of a session over multiple nodes at the same time.

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

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

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

Proof: 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. Therefore, after the $(k-1)th$ node, the output traffic of session i, and correspondingly the input traffic of node k, will conform to a leaky bucket model. That is,

$$A_k(\tau, t) \leq \left(\sigma_i + \rho_i \sum_{j=1}^{k-1} \Theta_i^{(S_j)} \right) + \rho_i (t - \tau). \tag{3.21}$$

□

This is a 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.

We can now state the following lemma that will bound the backlog of session i in each node of the network.

Lemma 3: *The maximum backlog $Q_i^{(S_k)}(t)$ in the kth node of a session is bounded as*

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

Proof: Let σ_i^k denote the maximum burstiness of session i at the output of the kth node. By Theorem 1 and Lemma 2 we have,

$$\begin{aligned} Q_i^{(S_k)} &\leq \sigma_i^{k-1} + \rho_i \Theta_i^{(S_k)} \\ &\leq \sigma_i + \rho_i \sum_{j=1}^{k} \Theta_i^{(S_j)}. \end{aligned} \tag{3.22}$$

□

As a result of the increased burstiness, the maximum backlog and therefore the maximum buffer requirements for each node in the network are also proportional to the latency of the servers. In Lemma 3 we bounded the maximum backlog for session i in any node of the network. This bound may be seen as the maximum number of buffers needed in the corresponding node of the network to guarantee zero packet loss for session i. However, this does not mean that we can bound the maximum number of packets in the network for session i by adding the backlog bounds of each switch on the path. Such a bound does not take into account the correlation among arrival processes at the individual nodes and therefore can be very loose.

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.

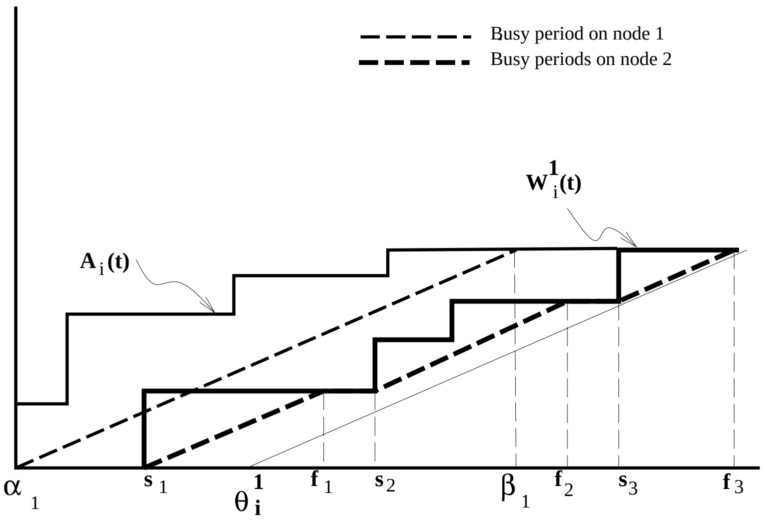

Figure 3.3: Illustration of busy periods of a session in two servers in series. The network busy period for session i in this case is split into multiple busy periods in the second server. The busy period in the first server is $(\alpha_1, \beta_1]$. The packets arriving at the second server from this 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.

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 3.3. 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 4: *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*

$$W_{i,j}(\tau,t) \geq \max\left(0, \rho_i(t-\tau-(\Theta_i^{(\mathcal{S}_1)}+\Theta_i^{(\mathcal{S}_2)}))\right).$$

Proof: We will prove the lemma by induction on the number of network busy periods. Let us denote with $(\alpha_j, \beta_j]$ the starting and ending times of the jth network busy period of session i. $W_{i,j}^1(\alpha_j, t)$ denotes the service offered by the first server S_1 to the packets of the jth network busy period until time t. Note that the jth network busy period is the same as the jth session-busy period on the first server. However, the busy periods seen by the the second server S_2 may be different from these periods. Let $(s_k, f_k]$ denote the kth busy period of session i in S_2. Then $W_{i,k}^2(s_k, t)$ is the service offered by S_2 during its kth busy period until time t.

Let τ_k denote the time at which packets from the kth busy period of session i in S_2 start service in S_2. Similarly, let t_k be the instant at which the last packet from the kth busy period in S_2 is completely serviced by S_2. Then, it is easy to observe that

$$\tau_k \geq s_k, \quad t_k \leq f_k + \Theta_i^{(\mathcal{S}_2)}, \quad \text{and } t_k \leq \tau_{k+1}.$$

Base step: Consider any time t during the first network busy period, that is, $\alpha_1 \leq t \leq \beta_1$. When the first busy period for session i starts there are no other packets from that session in the system. As observed earlier, the first network busy period of session i may result in multiple busy periods in the second server. Let $(s_1, f_1], (s_2, f_2], \ldots, (s_m, f_m]$ be the successive busy periods in the second server in which packets from the the first network busy period were serviced.

Let t^* denote the last instant of time at which a packet from the first network busy period was in service in S_2. We need to show that, for any time t, $\alpha_1 \leq t \leq t^*$,

$$W_{i,1}(\alpha_1, t) \geq \max\left(0, \rho_i(t-\alpha_1-(\Theta_i^{(\mathcal{S}_1)}+\Theta_i^{(\mathcal{S}_2)}))\right).$$

We need to consider three separate cases for t.

Case 1: $t \leq \tau_1$. In this case, no service has occurred in the second server, that is, $W_{i,1}(\alpha_1, t) = 0$. Also,

$$\begin{aligned} s_1 &\leq \alpha_1 + \Theta_i^{(\mathcal{S}_1)}, \quad \text{and} \\ \tau_1 &\leq s_1 + \Theta_i^{(\mathcal{S}_2)}. \end{aligned}$$

Therefore,

$$t - \alpha_1 \leq \Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)}. \tag{3.23}$$

Hence, we can write

$$W_{i,1}(\alpha_1, t) \geq \max\left(0, \rho_i(t-\alpha_1-(\Theta_i^{(\mathcal{S}_1)}+\Theta_i^{(\mathcal{S}_2)}))\right). \tag{3.24}$$

Case 2: $\tau_k \leq t \leq t_k$, $1 \leq k \leq m$ (within the kth busy period in S_2).

$$W_{i,1}(\alpha_1, t) = W_{i,1}(\alpha_1, \tau_k) + W_{i,1}(\tau_k, t) \tag{3.25}$$

But,

$$\begin{aligned} W_{i,1}(\alpha_1, \tau_k) &= W_{i,1}^1(\alpha_1, s_k); \\ \text{and} \quad W_{i,1}(\tau_k, t) &= W_{i,k}^2(\tau_k, t) = W_{i,k}^2(s_k, t). \end{aligned}$$

Substituting in eq. (3.25),

$$\begin{aligned} W_{i,1}(\alpha_1, t) &= W^1_{i,1}(\alpha_1, s_k) + W^2_{i,k}(s_k, t) \\ &\geq \max\left(0, \rho_i(s_k - \alpha_1 - \Theta_i^{(\mathcal{S}_1)})\right) + \max\left(0, \rho_i(t - s_k - \Theta_i^{(\mathcal{S}_2)})\right) \\ &\geq \max\left(0, \rho_i(t - \alpha_1 - (\Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)}))\right). \end{aligned}$$

Case 3: $t_k \leq t \leq \tau_{k+1}$ (between busy periods in S_2): We can write

$$W_{i,1}(\alpha_1, t) = W_{i,1}(\alpha_1, \tau_{k+1}) - W_{i,1}(t, \tau_{k+1}). \tag{3.26}$$

Since no packet arrives at the second server between its busy periods, the second term on the right-hand side is zero. Also, $W_{i,1}(\alpha_1, \tau_{k+1}) = W^1_{i,1}(\alpha_1, s_{k+1})$. Therefore, eq. (3.26) reduces to

$$\begin{aligned} W_{i,1}(\alpha_1, t) &= W^1_{i,1}(\alpha_1, s_{k+1}) \\ &\geq \max\left(0, \rho_i(s_{k+1} - \alpha_1 - \Theta_i^{(\mathcal{S}_1)})\right). \end{aligned} \tag{3.27}$$

But, $t \leq \tau_{k+1} \leq s_{k+1} + \Theta_i^{(\mathcal{S}_2)}$, or

$$s_{k+1} \geq t - \Theta_i^{(\mathcal{S}_2)}.$$

Therefore, we can rewrite eq. (3.27) as

$$W_{i,1}(\alpha_1, t) \geq \max\left(0, \rho_i(t - \alpha_1 - (\Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)}))\right).$$

If f_m marks the last instant of time when a packet from the first network busy period was serviced by S_2, then we have accounted for all the traffic arrived in the network during $(\alpha_1, \beta_1]$. However, it is possible that the transmission of packets arrived during $(\alpha_1, \beta_1]$ extends into the $(m+1)$th busy period in S_2. If so, let t^* be the last instant of time when a packet from the first network busy period was serviced by S_2. Then we need to consider the additional time intervals $(t_m, \tau_{m+1}]$ and $(\tau_{m+1}, t^*]$ in the proof. These can be handled exactly like in case 3 and case 2, respectively. This concludes the base step of the proof.

Inductive step: We assume that for all network busy periods $1, \ldots, n$, the lemma is true, and thus

$$W_{i,n}(\alpha_n, t) \geq \max\left(0, \rho_i(t - \alpha_n - (\Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)}))\right)$$

We will now prove that, for the $(n+1)$th busy period,

$$W_{i,n+1}(\alpha_{n+1}, t) \geq \max\left(0, \rho_i(t - \alpha_{n+1} - (\Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)}))\right).$$

In the simple case, the first session-i packet of the $(n+1)$th network busy period also starts a busy period in the second server. This case can be handled exactly as in the base step. The more difficult case occurs if, when the $(n+1)$th network busy period starts, some packets of session i from earlier busy periods are still backlogged in the second server. That is, the beginning of a network busy period of session i may not always coincide with the beginning of a busy period for S_2, as was the case in the base step. However, we will now show that this does not affect our analysis.

Let us assume that the mth busy period of S_2 is in progress when the first packet from the $(n+1)$th network busy period arrives at S_2. Assume that the mth busy period of S_2 started at time s_m, and the first packet from the $(n+1)$th network busy period was serviced by S_2 at time τ. Then,

$$\begin{aligned} W_{i,n+1}(\alpha_{n+1}, t) &= W_{i,n+1}(\tau, t) \\ &= W^2_{i,m}(s_m, t) - W^2_{i,m}(s_m, \tau) \end{aligned} \tag{3.28}$$

Note that the mth busy period currently in progress in S_2 may contain packets from multiple network busy periods, since multiple busy periods of S_1 may merge into a single busy period in S_2. Assume that the mth busy period of S_2 contained packets from the network busy periods $n-L, n-L+1, \ldots, n+1$. The total number of packets that are served during the mth busy period of the second server until time τ is equal to the total number of packets served from the network busy periods $n-L, n-L+1, \ldots, n$ minus those packets served from the $(n-L)$-th busy period before time s_m. Thus,

$$\sum_{j=n-L}^{n} A_i(\alpha_j, \beta_j) = W^1_{i,n-L}(\alpha_{n-L}, s_m) + W^2_{i,m}(s_m, \tau), \tag{3.29}$$

or equivalently,

$$\begin{aligned} W^2_{i,m}(s_m, \tau) &= \sum_{j=n-L}^{n} A_i(\alpha_j, \beta_j) - W^1_{i,n-L}(\alpha_{n-L}, s_m) \\ &\leq \sum_{j=n-L}^{n} \rho_i(\beta_j - \alpha_j) - \max(0, \rho_i(s_m - \alpha_{n-L} - \Theta_i^{(\mathcal{S}_1)})) \\ &\leq \rho_i(\beta_n - \alpha_{n-L}) - \max(0, \rho_i(s_m - \alpha_{n-L} - \Theta_i^{(\mathcal{S}_1)})) \qquad (3.30) \\ &\leq \rho_i(\beta_n - s_m + \Theta_i^{(\mathcal{S}_1)}) \qquad (3.31) \end{aligned}$$

From equations (3.28) and (3.31), and the definition of $\mathcal{LR}$ servers,

$$\begin{aligned} W_{i,n+1}(\alpha_{n+1}, t) &= W^2_{i,m}(s_m, t) - W^2_{i,m}(s_m, \tau) \\ &\geq \max(0, \rho_i(t - s_m - \Theta_i^{(\mathcal{S}_2)})) - \rho_i(\beta_n - s_m + \Theta_i^{(\mathcal{S}_1)}) \\ &\geq \max\left(0, \rho_i(t - \beta_n - (\Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)}))\right) \\ &\geq \max\left(0, \rho_i(t - \alpha_{n+1} - (\Theta_i^{(\mathcal{S}_1)} + \Theta_i^{(\mathcal{S}_2)})\right). \end{aligned}$$

The last inequality holds since $\beta_n \leq \alpha_{n+1}$.

Now, if the $(n+1)$th network busy period is later split into multiple busy periods in S_2, we can use the same approach used in the base step for subsequent busy periods in S_2 to complete the proof. □

Lemma 4 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.

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*

$$D_i \leq \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{K} \Theta_i^{(S_j)}, \tag{3.32}$$

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

Proof: From Corollary 1, we can treat the whole network as equivalent to a single $\mathcal{LR}$-server with latency equal to the sum of their latencies. By using Theorem 1, we can directly conclude that the maximum delay is

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

□

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 [16]. 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. (3.32) 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 $\Theta_i^{\mathcal{S}}$ 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.

3.3 Delay Bound for Sessions with Known Peak Rate

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

$$A_i(\tau, t) \leq \min\left(\sigma_i + \rho_i(t-\tau), P_i(t-\tau)\right). \tag{3.33}$$

We can prove the following lemma:

Lemma 5: *The maximum delay D_i of a session i in an $\mathcal{LR}$ server, where the peak rate of the source is known, is bounded as*

$$D_i^{\mathcal{S}} \leq \left(\frac{P_i - g_i}{g_i}\right)\left(\frac{\sigma_i}{P_i - \rho_i}\right) + \Theta_i^{\mathcal{S}}. \tag{3.34}$$

Proof: Let us assume that the maximum delay $D_i^{\mathcal{S}}$ was obtained for a packet that arrived at time t^* during the jth busy period. This means that the packet was serviced at time $t^* + D_i^{\mathcal{S}}$. Hence, the amount of service offered to the session until time $t^* + D_i^{\mathcal{S}}$ is equal to the amount of traffic that arrived from the session until time t. Since s_j is the beginning of the jth busy period,

$$W_{i,j}^{\mathcal{S}}(s_j, t^* + D_i^{\mathcal{S}}) = A_i(s_j, t^*). \tag{3.35}$$

From Eq. (3.33), this becomes

$$W_{i,j}^{\mathcal{S}}(s_j, t^* + D_i^{\mathcal{S}}) \leq \min\left(\sigma_i + \rho_i(t^* - s_j), P_i(t^* - s_j)\right). \tag{3.36}$$

Since the server may provide no service for time $\Theta_i^{\mathcal{S}}$, $D_i^{\mathcal{S}} \geq \Theta_i^{\mathcal{S}}$. From the definition of $\mathcal{LR}$-server,

$$W_{i,j}^{\mathcal{S}}(s_j, t^* + D_i^{\mathcal{S}}) \geq g_i(t^* + D_i^{\mathcal{S}} - s_j - \Theta_i^{\mathcal{S}}). \tag{3.37}$$

From (3.36) and (3.37), we have

$$g_i(t^* + D_i^{\mathcal{S}} - s_j - \Theta_i^{\mathcal{S}}) \leq \min\left(\sigma_i + \rho_i(t^* - s_j), P_i(t^* - s_j)\right). \tag{3.38}$$

Case 1: When $P_i(t^* - s_j) \leq \sigma_i + \rho_i(t^* - s_j)$, we have

$$D_i^{\mathcal{S}} \leq \left(\frac{P_i - g_i}{g_i}\right)(t^* - s_j) + \Theta_i^{\mathcal{S}}; \tag{3.39}$$

and

$$\begin{aligned} P_i(t^* - s_j) &\leq \sigma_i + \rho_i(t^* - s_j), \\ \text{or, } t^* - s_j &\leq \frac{\sigma_i}{P_i - \rho_i}. \end{aligned}$$

Substituting for $(t^* - s_j)$ in Eq. (3.39), we get

$$D_i^{\mathcal{S}} \leq \left(\frac{P_i - g_i}{g_i}\right)\left(\frac{\sigma_i}{P_i - \rho_i}\right) + \Theta_i^{\mathcal{S}}. \tag{3.40}$$

Case 2: When $P_i(t^* - s_j) > \sigma_i + \rho_i(t^* - s_j)$, we get

$$(t^* - s_j) > \frac{\sigma_i}{P_i - \rho_i}. \tag{3.41}$$

From Eq. (3.38),

$$(g_i - \rho_i)(t^* - s_j) \leq \sigma_i - g_i D_i^{\mathcal{S}} + g_i \Theta_i^{\mathcal{S}}; \tag{3.42}$$

and by substituting for $t^* - s_j$ from Eq. (3.41),

$$D_i^{\mathcal{S}} \leq \left(\frac{P_i - g_i}{g_i}\right)\left(\frac{\sigma_i}{P_i - \rho_i}\right) + \Theta_i^{\mathcal{S}}. \tag{3.43}$$

□

A network of $\mathcal{LR}$ servers can be modeled as a single $\mathcal{LR}$ server with latency equal to the sum of the latencies. Thus, the following main result can be derived:

Corollary 2: *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*

$$D_i \leq \left(\frac{P_i - g_i}{g_i}\right)\left(\frac{\sigma_i}{P_i - \rho_i}\right) + \sum_{j=1}^{K} \Theta_i^{S_j}. \tag{3.44}$$

where $\Theta_i^{S_j}$ is the latency of the jth node.

4 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 6: *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.

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. An easy way to determine if the bound is tight is to present at least one example in which the offered service is actually equal to the bound. This is the approach we will take in this section to determine the latencies of several $\mathcal{LR}$ servers.

Proof: We will prove the lemma by induction on the number of busy periods. Let us use $(\alpha_k, \beta_k]$ to denote the kth busy period of session i in the server.

Base step: At time α_1, the beginning of the first session-i busy period, the system becomes backlogged for the first time. Let us assume that the first busy period consists of a number of backlogged periods $(s_j, t_j]$. By the definition of session busy period, the following inequality must hold at the beginning of the jth backlogged period:

$$A(\alpha_1, s_j) \geq \rho_i(s_j - \alpha_1).$$

However, the system is empty at time s_j. Therefore,

$$A(\alpha_1, s_j) = W_{i,1}(\alpha_1, s_j). \tag{4.1}$$

Consider any time t in the interval $(s_j, t_j]$. We can write

$$\begin{aligned} W_i(\alpha_1, t) &= W_{i,1}(\alpha_1, t) \\ &= W_{i,1}(\alpha_1, s_j) + W_{i,1}(s_j, t) \\ &\geq \rho_i(s_j - \alpha_1) + \max(0, \rho_i(t - s_j - \Theta_i)) \\ &\geq \max(0, \rho_i(t - \alpha_1 - \Theta_i)) \end{aligned} \tag{4.2}$$

Now consider any time t within the busy period, but between two backlogged periods; that is, $t_j \leq t \leq s_{j+1}$. Since session i receives no service during the interval $(t, s_{j+1}]$, we can write

$$W_i(\alpha_1, t) = W_{i,1}(\alpha_1, t) = W_{i,1}(\alpha_1, s_{j+1}). \tag{4.3}$$

But,

$$\begin{aligned} W_{i,1}(\alpha_1, s_{j+1}) &= A_i(\alpha_1, s_{j+1}) \\ &\geq \rho_i(s_{j+1} - \alpha_1) \\ &\geq \rho_i(t - \alpha_1). \end{aligned} \tag{4.4}$$

Therefore, from (4.3) and (4.4),

$$W_i(\alpha_1, t) \geq \rho_i(t - \alpha_1). \tag{4.5}$$

Combining (4.2) and (4.5), we can conclude that, at any instant t during the first busy period $(\alpha_1, \beta_1]$,

$$W_i(\alpha_1, t) \geq \max(0, \rho_i(t - \alpha_1 - \Theta_i)). \tag{4.6}$$

Inductive step: Assume that the lemma is true for all busy periods $1, 2, \ldots, n$. We will now prove that the lemma is true for the $(n+1)$th busy period as well.

If the system is not backlogged when the $(n+1)$th busy period starts, we can repeat the same proof as in the base step. However, it is possible that, when the $(n+1)$th busy period starts, the system is still backlogged with packets from the nth or earlier busy periods. Let us assume that the backlogged period in progress when the $(n+1)$th busy period starts is the mth backlogged period. This backlogged period started at time s_m; in the general case, we can assume that packets from L earlier busy periods were serviced during the backlogged period $(s_m, t_m]$. Let τ denote the instant at which the last packet from the nth busy period was serviced. Then,

$$\begin{aligned} W_i(s_m, \tau) &= \sum_{j=n-L}^{n} A_i(\alpha_j, \beta_j) - W_{i,n-L}(\alpha_{n-L}, s_m) \\ &\leq \rho_i(\beta_n - \alpha_{n-L}) - \rho_i(s_m - \alpha_{n-L}) \\ &\leq \rho_i(\beta_n - s_m) \end{aligned} \tag{4.7}$$

We need to show that $W_{i,n+1}(\alpha_{n+1}, t) \geq \max(0, \rho_i(t - \alpha_{n+1} - \Theta_i))$. We can proceed as follows:

$$\begin{aligned} W_{i,n+1}(\alpha_{n+1}, t) &= W_i(\tau, t) \\ &= W_i(s_m, t) - W_i(s_m, \tau) \\ &\geq \max(0, \rho_i(t - s_m - \Theta_i)) - \rho_i(\beta_n - s_m) \\ &\geq \max(0, \rho_i(t - \beta_n - \Theta_i)) \\ &\geq \max(0, \rho_i(t - \alpha_{n+1} - \Theta_i). \end{aligned}$$

The last inequality follows from the fact that $\beta_n < \alpha_{n+1}$ and that the service offered to a session can never be negative. If, after time t, the packets of the $(n+1)$th busy period form multiple backlogged periods, the proof for the base step can be repeated to complete the proof of the lemma. □

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 [23] for providing QoS guarantees. This 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 6 we proved that, if we use the backlogged period to bound the service offered by a server $\mathcal{S}$, then $\mathcal{S}$ 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

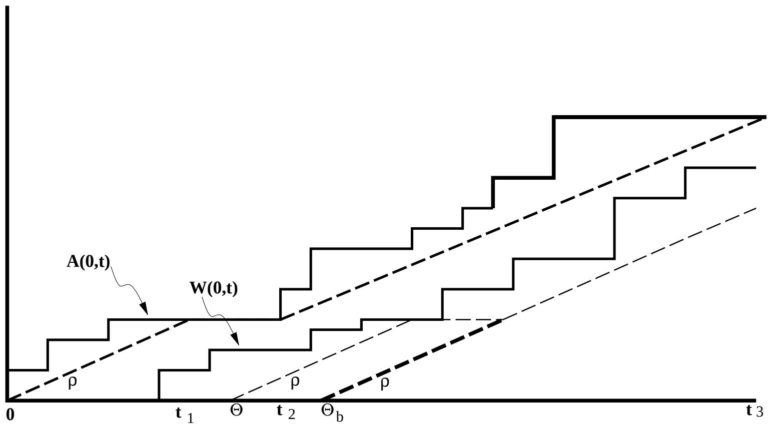

Figure 4.1: Difference in bounding service based on the backlogged and busy periods.

Server	Latency
GPS	0
PGPS	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$
SCFQ	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}(V-1)$
VirtualClock	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$
Deficit Round Robin	$\frac{3F-2\phi_i}{r}$
Weighted Round Robin	$\frac{F-\phi_i+L_c}{r}$

Table 4.1: Some $\mathcal{LR}$ servers and their latencies. L_i is the maximum packet size of session i and L_{max} the maximum packet size among all the sessions. 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.

the example of Figure 4.1. Let us assume an $\mathcal{LR}$-server with rate ρ and latency Θ. Referring to Figure 4.1, 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 the server, a latency $\Theta_b > \Theta$ would result. By extending 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 the above lemma as our tool, in Appendix 7 we analyze several work conserving servers and prove that they belong in the class $\mathcal{LR}$. A summary of our results is presented in Table 4.1.

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 6, however, that 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, 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.

5 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 $\Theta_i^{\mathcal{S}}$ 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 $\rho_i(t - \tau - \Theta_i^{\mathcal{S}})$. 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 $\Theta_i^{\mathcal{S}}$ 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 $\Theta_i^{\mathcal{S}}$ 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. (3.32).

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 $(L_i/\rho_i + L_{max}/r)$ computed in eq. (A.1).

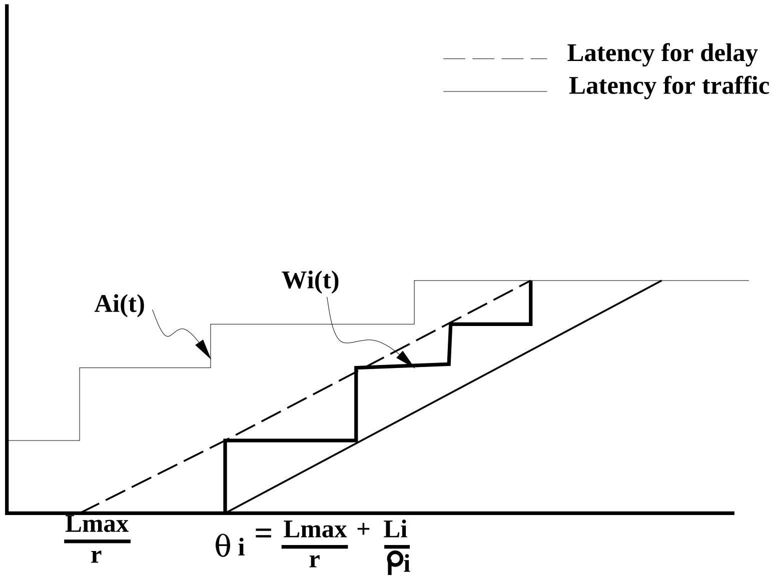

Figure 5.1: Illustration of the two envelopes used to bound the service received by session i in a session-busy period. Each step in the arrival function indicates a new packet. The lower envelope is a valid lower-bound for the service at any point in the busy period, while the upper one is valid only at the points when a packet leaves the system.

Figure 5.1 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. (3.32). 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 eq. (A.1) in this expression, 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 [16] 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 [24].

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.

6 Fairness of $\mathcal{LR}$ Servers

In Section 3, 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 [5] 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 illustrated in Figure 6.1. Assume that two connection share an outgoing link and are allocated equal shares of the link bandwidth. Assume each packet is of unit size and the rate of the server is also unity. Consider an interval of time 0–1000 during which only connection 1 is active, and sends 1000 packets. Connection 2 becomes active at time 1000, and both connections send packets after 1000. Assume that the scheduler is based on the VirtualClock algorithm. Since the server is work-conserving,

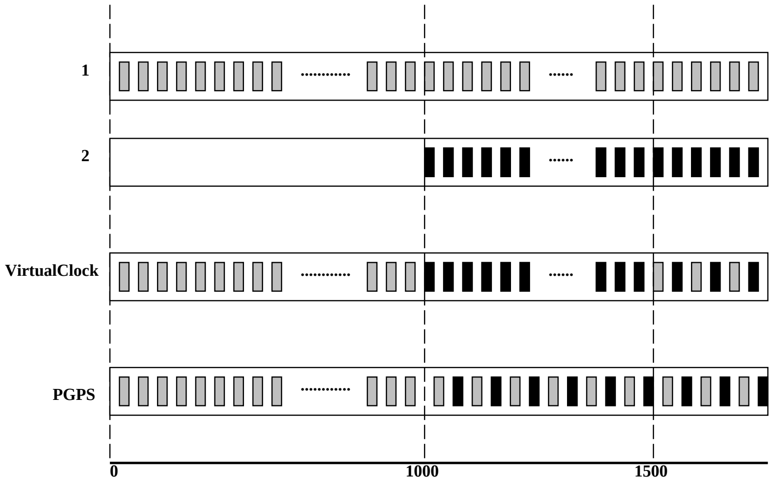

Figure 6.1: Unfair behavior of the VirtualClock scheduling algorithm. The two connections are allocated equal portions of the outgoing link bandwidth.

it will service all 1000 packets from connection 1 by $t = 1000$. However, at time $t = 1000$, the next arriving packet of connection 1 will receive a timestamp of 2000, reflecting the average service received by the connection until 1000. Thus, connection 1 will be starved until time 1500, when the timestamps of the packets of the two connections become equal. If the maximum burstiness of the sources is not bounded, we can see that the interval over which a backlogged connection is denied service can grow to infinity. A PGPS server, in contrast, provides equal service to the two connection after $t = 1000$, regardless of the excess bandwidth received by connection 1 earlier.

We now evaluate the fairness of a number of well-known servers. We use L_i to denote the maximum packet-size for session i and L_{max} the maximum packet-size over all sessions. For VirtualClock, we have already seen that there is no finite value of $\mathcal{F}^{\mathcal{S}}$ satisfying our definition of fairness.

6.1 Fairness of a PGPS Scheduler

Based on the above measure of fairness, the fairness of a PGPS server is given by the following lemma. A detailed proof of the lemma can be found in [25].

Lemma 7: *For a PGPS scheduler,*

$$\mathcal{F}^{\mathcal{S}} = \max(\max(C_j + \frac{L_{max}}{\rho_i} + \frac{L_j}{\rho_j}, C_i + \frac{L_{max}}{\rho_j} + \frac{L_i}{\rho_i})$$

where C_i is the maximum normalized service that a session may receive in a PGPS server in excess of that in the GPS server, given by

$$C_i = \min\left((V-1)\frac{Lmax}{\rho_i}, \max_{1 \le n \le V}(\frac{L_n}{\rho_n})\right).$$

It can be shown that the above bound is tight.

6.2 Fairness of Self-Clocked Fair Queueing

Golestani [5] proved the following bound for SCFQ:

$$\mathcal{F}^{\mathcal{S}} = \frac{L_i}{\rho_i} + \frac{L_j}{\rho_j}.$$

We now prove that this bound is tight by presenting an example where the bound is actually reached.

Let us assume that at time t_1, the kth packet from connection i has just been serviced by the SCFQ server. The next packet will have a virtual finishing time equal to

$$F_i^{k+1} = F_i^k + \frac{L_i}{\rho_i}.$$

Packets from connection j may be serviced after the kth packet of connection i, as long as their virtual finishing times are less than or equal to F_i^{k+1}. Let us assume that connection j has a packet x of size L_j with the same finishing time F_i^k. Assume also that j has a set of packets queued behind x with total size equal to

$$\frac{L_i}{\rho_i}\rho_j.$$

The last of these packets will get a virtual finishing time of

$$F_j^m = F_i^k + \frac{L_i}{\rho_i}.$$

Therefore, a total amount of traffic equal to $L_j + \frac{L_i}{\rho_i}\rho_j$ may be serviced from connection j before the $(k+1)$th packet from connection i is serviced. Let t_2 be time at which the packets of j finish service. Then,

$$\begin{aligned} \frac{W_j(t_1,t_2)}{\rho_j} - \frac{W_i(t_1,t_2)}{\rho_i} &= \frac{1}{\rho_j}(L_j + \frac{L_i}{\rho_i}\rho_j) \\ &= \frac{L_i}{\rho_i} + \frac{L_j}{\rho_j}. \end{aligned}$$

6.3 Fairness of a Round-Robin Scheduler

Deficit Round Robin was proposed by Sreedhar and Varghese [8] as an $O(1)$ algorithm for providing bandwidth guarantees in an output-buffered switch. Deficit Round Robin is a generalization of the Weighted-Round-Robin algorithm that was proposed in the context of ATM networks [7]. The latter assumes that packets from all connections have the same size and connections are serviced in a round-robin order. The time is split into frames of maximum size F and a connection is not allowed to send more than ϕ_i packets during a frame period. Therefore, the bandwidth allocated to a connection is

$$\rho_i \geq \frac{\phi_i}{F} r,$$

Deficit Round Robin estimates the service offered to a connection during a round in terms of bytes and not in terms of packets, allowing it to be used with variable-size packets. Connections are serviced in a round-robin fashion. Once a connection is selected, up to ϕ_i bytes may be sent for that connection. If the transmission of a whole packet forces the connection to send more than ϕ_i packets, the last packet is not sent and a deficit counter D_i^k is updated, indicating the service that was missed from the connection during the kth round. This deficit is then added to ϕ_i in the next round while servicing traffic from connection i.

Another variation of the round-robin service disciplines is the Surplus Round Robin (SRR) [26, 27]. The difference of this policy from Deficit Round Robin is that, if a packet forces the connection to send more than ϕ_i bytes in a frame, the packet is sent and a surplus counter S_i^k is updated to reflect the amount of excess service that was offered to the connection during the kth round. This service will be lost during the next round.

It has been shown that, in the Deficit Round Robin algorithm, the difference in service offered to any two connections that have the same bandwidth reservation is bounded by $3\phi_i$, where ϕ_i is the number of bytes allocated to these connections in each frame [8]. Here we extend the result to the case of two connections with arbitrary bandwidth allocations.

Lemma 8: *For a Deficit-Round-Robin scheduler,*

$$\mathcal{F}^{\mathcal{S}} = \frac{3F}{r}.$$

Proof: Let us assume a time interval $(t_0, t_n]$, where t_0 is the beginning of a round and t_n is the end of the nth round after t_0. For each $k = 1, 2 \ldots, n$ when the connection is continuously backlogged,

$$W_i(t_{k-1}, t_k) = \phi_i + D_i^{k-1} - D_i^k, \tag{6.1}$$

where D_i^k is the value of the deficit counter of connection i at the end of the kth round. Summing over k,

$$W_i(t_0, t_k) = k\phi_i + D_i^0 - D_i^k. \tag{6.2}$$

Since $D_i^k \leq \phi_i$, for every k, we can write

$$W_i(t_0, t_k) \leq k\phi_i + \phi_i. \tag{6.3}$$

Similarly, if connection j is continuously backlogged over the same interval, we can write

$$\begin{aligned} W_j(t_0, t_k) &= k\phi_j + D_j^0 - D_j^k. \\ &\geq k\phi_j - \phi_j. \end{aligned} \tag{6.4}$$

From equations (6.3) and (6.4) we can easily conclude that

$$\begin{aligned} \left| \frac{W_i(t_0, t_k)}{\rho_i} - \frac{W_j(t_0, t_k)}{\rho_j} \right| &\leq \frac{\phi_i}{\rho_i} + \frac{\phi_j}{\rho_j} \\ &\leq \frac{2F}{r}. \end{aligned}$$

This bound applies to time intervals that span complete frames. For an arbitrary interval, the difference in service offered to the two connections may increase by another ϕ_i or ϕ_j bytes. By normalizing this additional difference, we can obtain the general bound for fairness as

$$\mathcal{F}^{\mathcal{S}} = \frac{3F}{r}. \tag{6.5}$$

□

In the case of Weighted Round Robin, where the deficit is always zero across the rounds, an extension of the above proof will lead us to the following corollary:

Corollary 3: *For a Weighted-Round-Robin scheduler,*

$$\mathcal{F}^{\mathcal{S}} = \frac{F}{r}.$$

In concluding this section, 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.

7 Conclusions

In Table 2, 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 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 [25], 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

Server	Latency	Fairness	Complexity
GPS	0	0	-
PGPS	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$	$\max(\max(C_j+\frac{L_{max}}{\rho_i}+\frac{L_j}{\rho_j}, C_i+\frac{L_{max}}{\rho_j}+\frac{L_i}{\rho_i})$, where $C_i = \min((V-1)\frac{L_{max}}{\rho_i}, \max_{1\leq n\leq V}(\frac{L_n}{\rho_n}))$.	$O(V)$
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) [25]	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$	$\max(\frac{2F-\phi_i}{r} + \frac{L_i}{\rho_i}, \frac{2F-\phi_i}{r} + \frac{L_j}{\rho_j}, \frac{L_{max}}{\rho_j} + \frac{L_i}{\rho_i}, \frac{L_{max}}{\rho_i} + \frac{L_j}{\rho_j})$	$O(\log V)$

Table 7.1: 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.

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.

It should be noted that our approach of modeling a network of $\mathcal{LR}$ servers as a single $\mathcal{LR}$ server did not make any assumptions about the input traffic. Although the bounds derived in this paper for the end-to-end delays and buffer requirements hold only when the input traffic is leaky-bucket shaped, the same model can be used for analysis with other input-traffic models. For example the *Exponentially-Bounded-Burstiness* (EBB) model was used in [28, 29] for analyzing the behavior of a GPS multiplexer. Future work will include the use of other traffic models for providing end-to-end delay guarantees in a network of $\mathcal{LR}$ servers.

The fairness analysis presented in Section 6 did not cover the intervals of time where a connection may be idle. A more complete analysis should include such intervals. In a fair algorithm, the service lost by two connections while they were idle should also be proportional to their reservations. In addition, if one connection is idle and the other backlogged, the idle connection must be losing normalized service proportional to the service that the backlogged connection is receiving. Future work will include the definition of a universal method for estimating the fairness of a scheduling algorithm. This measure must depend only on the characteristics of the server, and not on the traffic parameters of individual sessions. However, it must be a sufficient condition to prove that the server is providing service to a connection comparable to that provided by by a GPS multiplexer.

References

[1] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," in *Proceedings of 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 *Proceedings of INFOCOM '92*, vol. 2, pp. 915–924, May 1992.

[5] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of 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 Conferece*, pp. 300.3.1–300.3.9, December 1990.

[10] S. Golestani, "A framing strategy for congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1064–1077, September 1991.

[11] D. Verma, D. Ferrari, and H. Zhang, "Guaranteeing delay jitter bounds in packet switching networks," in *Tricomm 91*, April 1991.

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

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

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

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

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

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

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

[19] J. Turner, "New directions in communications (or which way to the information age?)," *IEEE Communications Magazine*, vol. 24, pp. 8–15, October 1986.

[20] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. New York: McGraw Hill, 1989.

[21] R. Brown, "Calendar queues: a fast 0(1) priority queue implementation for the simulation event set problem," *Communications of the ACM*, vol. 31, pp. 1220–1227, October 1988.

[22] H. Chao and N. Uzun, "A VLSI sequencer chip for ATM traffic shaper and queue manager," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1634–1643, November 1992.

[23] 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.

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

[25] 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, http://www.cse.ucsc.edu/research/hsnlab/publications/, July 1995.

[26] H. Adiseshu, G. Parulkar, and G. Varghese, "Reliable FIFO load balancing over multiple fifo channels," tech. rep., Washington University, St. Louis, May 1995.

[27] S. Floyd, "Notes on guaranteed service in resource management." Unpublished notes, 1993.

[28] O. Yaron and M. Sidi, "Performance and stability of communication networks via robust exponential bounds," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 372–385, June 1993.

[29] Z. Zhang, D. Towsley, and J. Kurose, "Statistical analysis of generalized processor sharing scheduling discipline," in *Proceedings of ACM SIGCOMM '94*, pp. 68–77, September 1994.

Appendix A: Schedulers in the Class $\mathcal{LR}$ and Their Latencies

This appendix contains the proofs for showing that many well-known scheduling algorithms belong to the class of $\mathcal{LR}$ servers and derives their latencies. In the following proofs, L_i denotes the maximum packet-size for session i and L_{max} the maximum packet-size among all sessions.

A.1 Weighted-Fair-Queueing

We first start by showing that a GPS scheduler is an $\mathcal{LR}$ server.

Lemma 9: *A GPS scheduler belongs to the class $\mathcal{LR}$ and its latency is zero.*

Proof: From the definition of the GPS multiplexer, during any interval of time that connection i is continuously backlogged,

$$W_i^{GPS}(\tau, t) \geq \rho_i(t - \tau).$$

From Lemma 6 it easy to conclude that a GPS scheduler is an $\mathcal{LR}$-server with zero latency.

□

Weighted-Fair-Queueing or Packet-by-packet GPS (PGPS) scheduling algorithm is the packet-by-packet equivalent of GPS multiplexing. In [4] it was proven that, if t^F, t^P are the times that a packet finishes under WFQ and GPS, respectively, then

$$t^P \leq t^F + \frac{L_{max}}{r},$$

where r is the transmission rate of the server. For the analysis of a network of $\mathcal{LR}$ servers it is required that the service be bounded for any time after the beginning of a busy period. In addition, we can only consider that a packet left a packet-by-packet server if all of its bits have left the server. These requirements are necessary in order to provide accurate bounds for the traffic burstiness inside the network. Therefore, just before time t^P the whole packet has not yet departed the packet-by-packet server. Let L_i be the size of the packet. The service offered to connection i in the packet-by-packet server will be equal to the service offered to the same connection by the GPS multiplexer until time t^P minus this last packet. Notice also that in a GPS multiplexer, since the latency is zero, the beginning of a busy period finds the system always empty. Therefore,

$$\begin{aligned} W_{i,j}^P(\tau, t) &\geq W_{i,j}^F(\tau, t - \frac{L_{max}}{r}) - L_i \\ &\geq \max(0, \rho_i(t - \tau - \frac{L_{max}}{r}) - L_i) \\ &\geq \max(0, \rho_i(t - \tau - \frac{L_{max}}{r} - \frac{L_i}{\rho_i})). \end{aligned}$$

Hence, we can state the following corollary:

Corollary 4: *Weighted-Fair-Queueing is an $\mathcal{LR}$ server with latency*

$$\frac{L_{max}}{r} + \frac{L_i}{\rho_i}.$$

This latency can be shown to be tight.

A.2 VirtualClock

The VirtualClock service discipline is based on the calculation of a timestamp for each packet that arrives in the system. The timestamp is calculated based on the timestamp of the previous packet of the same connection and the real time that passed after the beginning of the system busy period. In [25], we prove the following theorem that classifies VirtualClock as an $\mathcal{LR}$ server.

Theorem 3: *The VirtualClock service discipline is an $\mathcal{LR}$ server and its latency is*

$$\frac{L_{max}}{r} + \frac{L_i}{\rho_i}.$$

A.3 Self-Clocked Fair Queueing

In this section we will analyze the self-clocked fair queueing (SCFQ) algorithm and prove that it is also an $\mathcal{LR}$-server. SCFQ was proposed as a simplification of the PGPS server and is a sorted-priority algorithm. When a connection becomes backlogged, the virtual time is estimated by the virtual finishing time of the packet that is being serviced. Let TS_{cur} indicate this time. Then, the virtual finishing time of the jth packet of connection i is estimated as

$$F_i^j = \max(F_i^{j-1}, TS_{cur}) + \frac{L_i^j}{\rho_i},$$

where L_i^j is the length of the jth packet of connection i. Packets are always serviced in increasing order of their virtual finishing times. When the system becomes empty, all variables are reset to zero.

Let us assume that a connection becomes backlogged at time τ and let this be the kth packet of the connection. We first prove the following lemma:

Lemma 10: *If the kth packet of session i starts a backlogged period in SCFQ server, its virtual finishing time will be estimated by the system virtual time:*

$$F_i^k = TS_{cur} + \frac{L_i^j}{\rho_k}.$$

Proof: We will prove the lemma by contradiction. Let us assume that

$$F_i^{k-1} > TS_{cur}.$$

Then, the packet F_i^{k-1} has not been serviced yet and thus the kth packet does not find the system empty. □

We can now prove the following theorem:

Theorem 4: *At ant time t during a session i backlogged period in an SCFQ server,*

$$W_i(\tau, t) \geq \rho_i(t - \tau - (V-1)\frac{Lmax}{r} - \frac{L_i}{\rho_i}).$$

Proof: Let τ be the start of a backlogged period for session i in the SCFQ server. Assume that the backlogged period was started by the arrival of the kth packet of session i. By Lemma 10, the virtual finishing time of this packet will be

$$F_i^k = TS_{cur} + \frac{L_i^j}{\rho_i}. \tag{A.1}$$

Consider the nth of the same backlogged period of session i, that is, the $(k+n-1)$th packet of session i. Let us mark this packet. The virtual finishing time of this packet will be

$$F_i^{k+n-1} = TS_{cur} + n\frac{L_i}{\rho_i}. \tag{A.2}$$

Let us calculate how much traffic will be serviced from other sessions ahead of this marked packet in the worst case. Every other session may have a packet of size L_{max} with finishing time $= TS_{cur}$. In addition, every connection $j \neq i$ may transmit a maximum of $(nL_i/\rho_i)\rho_j$ total traffic before the marked packet is transmitted, as their timestamps will be within $TS_{cur} + nL_i/\rho_i$. Thus, the time at which the marked packet will be transmitted is given by

$$\begin{aligned} D_i^{k+n-1} &\leq \tau + (V-1)\frac{L_{max}}{r} + \frac{1}{r}\left(\sum_{j=k}^{k+n-1} L_i + \sum_{\substack{j=1 \\ j\neq i}}^{V} n\frac{L_i}{\rho_i}\rho_j\right) \\ &\leq \tau + (V-1)\frac{L_{max}}{r} + \frac{1}{r}\left(nL_i + n\frac{(r-\rho_i)}{\rho_i}L_i\right) \\ &\leq \tau + (V-1)\frac{L_{max}}{r} + n\frac{L_i}{\rho_i}. \end{aligned} \tag{A.3}$$

Consider any time $t \geq \tau$ during the session backlogged period. Assume the marked packet was the last serviced during the period $(\tau, t]$ from session i. Then, the total service received by session i during $(\tau, t]$ is given by

$$W_i(\tau, t) = nL_i. \tag{A.4}$$

But, t must be less than the time at which the $(k+n)$th packet of the session will leave the server. That is,

$$t \leq \tau + (V-1)\frac{L_{max}}{r} + (n+1)\frac{L_i}{\rho_i}, \tag{A.5}$$

or equivalently

$$nL_i \geq \rho_i\left(t - \tau - (V-1)\frac{L_{max}}{r} - \frac{L_i}{\rho_i}\right). \tag{A.6}$$

Substituting for nL_i from (A.6) in (A.4),

$$W_i(\tau, t) \geq \rho_i\left(t - \tau - (V-1)\frac{L_{max}}{r} - \frac{L_i}{\rho_i}\right). \tag{A.7}$$

Since $W_i(\tau, t)$ cannot be negative, we can write this as

$$W_i(\tau, t) \geq \max\left(0, \rho_i(t - \tau - (V-1)\frac{L_{max}}{r} - \frac{L_i}{\rho_i})\right). \tag{A.8}$$

□

By using Lemma 6 we can state the following corollary:

Corollary 5: *The SCFQ scheduler belongs to the class $\mathcal{LR}$ and its latency is less than or equal to*

$$(V-1)\frac{Lmax}{r}+\frac{L_i}{\rho_i}.$$

To prove that this latency is tight, it is sufficient to show an example where it actually occurs. Let the transmission rate of the server be 1. Assume that connection i is allocated rate ρ_i and the other $V-1$ connections are allocated rate $(1-\rho_i)/(V-1)$. Assume further that all the connections except i became backlogged at time 0 with a packet of size L_{max}. They all get the same virtual finishing time TS_{cur}. Just after the first packet of one of these connections starts being serviced, a packet of size L_i arrives from connection i. Then,

$$F_i^1 = TS\text{cur} + \frac{L_i}{\rho_i}.$$

This packet and all subsequent packets that belong in the same backlogged period of i will be delayed by the packets with virtual finishing time equal to TS_{cur}. Now assume that each connection $j \neq i$ transmits packets with total size equal to

$$S = \frac{L_i}{\rho_i}\rho_j.$$

The virtual finishing times of the last packet in this group will be equal to $TS_{cur} + \frac{L_i}{\rho_i}$, for each connection j. Thus, in the worst case, the packet from connection i may be serviced only after all these packets complete service. Thus, the first packet of connection i will only finish at time

$$\begin{aligned} D_i &= (V-1)\frac{L_{max}}{r}+\frac{1}{r}\sum_{\substack{j=1\\ j\neq i}}^{V}\frac{L_i}{\rho_i}\rho_j+\frac{L_i}{r} \\ &= (V-1)\frac{L_{max}}{r}+\frac{L_i}{\rho_i}\frac{(r-\rho_i)}{r}+\frac{L_i}{r} \\ &= (V-1)\frac{L_{max}}{r}+\frac{L_i}{\rho_i}. \end{aligned}$$

A.4 Deficit Round Robin

In this section we will analyze the Deficit Round Robin algorithm. We prove the following lemma:

Lemma 11: *Let t_0 be the beginning of a backlogged period of session i in a Deficit Round Robin server. The, at any time t during the backlogged period,*

$$W_i(t_0,t) \geq \max(0, \rho_i(t-t_0-\frac{(3F-2\phi_i)}{r})).$$

Proof: Let a connection become backlogged at time t_0. Let t_k indicate the time that a round finishes in the Deficit Round Robin algorithm. It has been shown in [8] that, if connection i is continuously backlogged in the interval $(\tau, t_k]$ then at the end of the kth round,

$$W_i(t_0,t_k) \geq k\phi_i - D_i^k. \qquad \text{(A.9)}$$

For each interval of time $(t_{k-1}, t_k]$, we can write

$$t_k - t_{k-1} \leq \frac{1}{r}\left(F + \sum_{j=1}^{V} D_j^{k-1} - \sum_{j=1}^{V} D_j^k\right). \tag{A.10}$$

By summing over k,

$$\begin{aligned} t_k - t_0 &\leq k\frac{F}{r} + \frac{1}{r}\sum_{j=1}^{V} D_j^0 - \frac{1}{r}\sum_{j=1}^{V} D_j^k \\ &\leq k\frac{F}{r} + \frac{F-\phi_i}{r}, \end{aligned}$$

or equivalently,

$$k \geq \frac{t_k - t_0}{F}r + \frac{\phi_i}{F} - 1. \tag{A.11}$$

Replacing k in Equation (A.9),

$$\begin{aligned} W_i(t_0, t_k) &\geq \phi_i(\frac{t_k - t_0}{F}r + \frac{\phi_i}{F} - 1) - D_i^k \\ &\geq \rho_i(t_k - t_0) - \phi_i(1 - \frac{\phi_i}{F}) - D_i^k. \end{aligned} \tag{A.12}$$

In the worst-case connection i was the last to receive service during the kth round. We will distinguish two cases:

Case 1: Connection i transmitted more than ϕ_i during the kth round. Then, the maximum amount of packets sent is bounded by $2\phi_i - D_i^k$. Therefore, for any time from the beginning of the kth round until these packets start being serviced,

$$\begin{aligned} W_i(t_0, t) &\geq \max(0, \rho_i(t_k - t_0) - \phi_i(1 - \frac{\phi_i}{F}) - D_i^k - 2\phi_i + D_i^k) \\ &\geq \max(0, \rho_i(t_k - t_0) - 3\phi_i + \phi_i\frac{\phi_i}{F}) \\ &\geq \max(0, \rho_i(t - t_0 - \frac{(3F - 2\phi_i)}{r})). \end{aligned}$$

The last inequality holds from the fact that at least ϕ_i bytes were serviced from the connection i in the kth round. Therefore $t \leq t_k - (\frac{\phi_i}{r} + D_i^k)$. When these packets start being serviced they are serviced with rate $r \geq \rho_i$ and therefore the above relation holds for any time t during the kth round.

Case 2: The amount of packets serviced during the kth round for connection i was less than or equal to ϕ_i. Then, again for any time t before these packets are serviced during the kth round,

$$\begin{aligned} W_i(t_0, t) &\geq \max(0, \rho_i(t_k - t_0) - \phi_i(1 - \frac{\phi_i}{F}) - D_i^k - \phi_i + D_i^k) \\ &\geq \max(0, \rho_i(t_k - t_0) - 2\phi_i + \phi_i\frac{\phi_i}{F}) \\ &\geq \max(0, \rho_i(t - t_0 - 2\frac{F}{r} + \frac{\phi_i}{r})) \end{aligned}$$

Since $\phi_i \leq F$,

$$W_i(t_0, t) \geq \max(0, \rho_i(t - t_0 - \frac{(3F - 2\phi_i)}{r})).$$

The case where connection i may have transmitted less than ϕ_i bytes in the last round is covered in the analysis if we assume that the deficit counter is reset to zero only after the end of the round. Thus, the counter will be equal to ϕ_i minus the number of bytes that were transmitted during this round, even if the queue was empty during the round.

Using Lemma 6 we can state the following corollary:

Corollary 6: *Deficit Round Robin is an $\mathcal{LR}$-scheduler and its latency is equal to* $(3F - 2\phi_i)/r$.

In order to be complete we have to show that the latency that we calculated is as tight as possible. It is sufficient to show an example that satisfies the above relation. Let us assume that connection i becomes backlogged at time τ. At this time, all other connections have a deficit counter $D_j = \phi_j - \Delta\phi_j$. Connection i is the last that will be serviced in the round and thus it has to wait for time at least equal to $\sum_{j \neq i}(\phi_j + D_j)$. Let us assume that the first packet of connection i has a size of $\Delta\phi_i$ and the second packet has a size of ϕ_i. Then, only the first packet will be serviced. After that time, ϕ_j packets of each one of the other connections may be serviced before the second packet is serviced. The time that this packet finishes service will be equal to

$$\sum_{j \neq i}(\phi_j + D_j) + \sum_{j \neq i}\phi_j + \phi_i + \Delta\phi_i$$

or equivalently

$$3\sum_{j=1}^{V}\phi_j - 2\phi_i - \sum_{j \neq i}\Delta\phi_j + \Delta\phi_i$$

If $\Delta\phi_j << F$ we can easily see that the bound is very close to $3F - 2\phi_i$.

Weighted round robin scheduling can be considered as a special case of the Deficit Round Robin algorithm. However, there is no requirement for the deficit counter, since ϕ_i is always an integer multiple of packets (cells). The extension of the above proof to weighted-round-robin is straightforward.

Lemma 12: *Weighted-round-robin is an $\mathcal{LR}$-server with latency*

$$\frac{(F - \phi_i + L_c)}{r},$$

where L_c is the size of an ATM cell.

Proof: In the worst case all cells of session i will be transmitted in the end of the round. Let $t_1, \ldots t_k$ denote the ending times of the k-th round after the beginning of a backlogged period at time t_0. Let t be a time during the k-th round after the beginning of a backlogged period for session i that the j-th cell of session i starts transmission. Then,

$$\begin{aligned} W_i(t_0, t) &= W_i(t_0, t_{k-1}) + W_i(t_{k-1}, t) && \text{(A.13)} \\ &= \max(0, (k-1)\phi_i) + (j-1)L_c && \text{(A.14)} \\ & && \text{(A.15)} \end{aligned}$$

But since session i is continuously backlogged,

$$t_{k-1} - t_0 \leq (k-1)\frac{F}{r} \tag{A.16}$$

From Equations (A.15) and (A.16) and substituting for $(k-1)$,

$$\begin{aligned} W_i(t_0, t) &\geq \max(0, (t_{k-1} - t_0)\frac{\phi_i}{F}r) + (j-1)L_c && \text{(A.17)} \\ &\geq \max(0, \rho_i(t_{k-1} - t_0)) + (j-1)L_c && \text{(A.18)} \end{aligned}$$

But we also know that at time t,

$$\begin{aligned} t &\leq t_{k-1} + \frac{\sum_{\substack{n=1 \\ n \neq i}}^{V} \phi_n + jL_c}{r} && \text{(A.19)} \\ &\leq t_{k-1} + \frac{F - \phi_i + jL_c}{r} && \text{(A.20)} \end{aligned}$$

Or,

$$t_{k-1} \geq t - \frac{F - \phi_i + jL_c}{r} \tag{A.21}$$

Substituting in Equation (A.18),

$$W_i(t_0, t) \geq \max(0, \rho_i(t - t_0 - \frac{F - \phi_i + jLc}{r})) + (j-1)L_c \tag{A.22}$$

Since the minimum value of the right-hand side of the above inequality occurs when $j = 1$, we can write

$$W_i(t_0, t) \geq \max(0, \rho_i(t - t_0 - \frac{F - \phi_i + L_c}{r})) \tag{A.23}$$

This is essential the latency of a cell that arrives in the beginning of a round and is serviced at the end of the round. It can be easily verified that this bound for the latency is tight.

□