Leave-in-Time: A New Service Discipline for Real-Time Communications in a Packet-Switching Network

Norival R. Figueira and Joseph Pasquale

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093-0114

{norival, pasquale}@cs.ucsd.edu

Abstract

Leave-in-Time is a new rate-based service discipline for packet-switching nodes in a connection-oriented data network. Leave-in-Time provides sessions with upper bounds on end-to-end delay, delay jitter, buffer space requirements, and an upper bound on the probability distribution of end-to-end delays. A Leave-in-Time session's guarantees are completely determined by the dynamic traffic behavior of that session, without influence from other sessions. This results in the desirable property that these guarantees are expressed as functions derivable simply from a single fixed-rate server (with rate equal to the session's reserved rate) serving only that session. Leave-in-Time has a non-work-conserving mode of operation for sessions desiring low end-to-end delay jitter. Finally, Leave-in-Time supports the notion of *delay shifting*, whereby the delay bounds of some sessions may be decreased at the expense of increasing those of other sessions. We present a set of admission control algorithms which support the ability to do delay shifting in a systematic way.

1 Introduction

Real-time applications generate network traffic that requires stringent performance guarantees in terms of throughput, end-to-end delay, and packet loss rate. These performance guarantees are generally not provided by conventional window-based flow control and first-come-first-served (FCFS) service disciplines. In fact, this is a primary motivation for rate-based flow control and rate-based service disciplines. With rate-based flow control, each session has a guaranteed minimum data rate without being affected by the traffic behavior of the other sessions sharing the same server. This guaranteed data rate generally requires some admission control mechanism to allocate the finite link capacity of the servers.

Several rate-based service disciplines have been proposed: Delay-EDD [5], Jitter-EDD [22], RCSP [26], VirtualClock [29], PGPS [17, 18, 19, 20], Stop-and-Go [9, 10, 11], and Hierarchical Round Robin [13]. All of these service disciplines provide an upper bound on end-to-end delay (this includes VirtualClock for which an upper bound on end-to-end delay was unknown until recently proven in [7]). Jitter-EDD, RCSP, and Stop-and-Go also provide an upper bound on delay jitter.

Providing an upper bound on delay has been a major point of concern for previously proposed service disciplines; however, even this is not enough. The delay *distribution* of packets is likely to be very useful for *tolerant* applications [1]. Tolerant applications permit some brief interruptions in service; the level of tolerance might be defined as a maximum percentage of missing packets over some period of time. For example, tolerant audio applications allow some percentage of missing packets due to excessive delay, i.e. packets that are late and are discarded by the application, in exchange for working with a lower play-back delay. Thus, while a service discipline that provides only an upper bound on delay may not be adequate for tolerant applications, one that provides a delay distribution *even where there is no upper bound on delay* may be appropriate.

We describe a new service discipline called Leave-in-Time which provides a session with upper bounds on delay, delay distribution, and delay jitter, all end-to-end, and an upper bound on server buffer space. The only requirements are that a session declare its minimum required bandwidth which must be reserved for each link along the path carrying the session, and that the maximum length of the session's packets be bounded. *No additional traffic characterization is required for Leave-in-Time.* The main feature of this service discipline is that it provides sessions with performance isolation: all the mentioned performance bounds for a session depend only on the dynamic traffic behavior of that session, and are not affected by the behavior of other sessions being transported over the same links and servers. Thus, a session can "know" what its performance bounds will be based on how it and it alone behaves.

Leave-in-Time builds on ideas found in VirtualClock [29] and Jitter-EDD [22]. The reader will see that Leave-in-Time exploits the good properties of VirtualClock and Jitter-EDD while maintaining efficiency and flexibility, and providing desirable performance bounds. For the special case where Leave-in-Time operates like VirtualClock, we will show that the upper bound on delay for sessions conforming to a token bucket filter (also called leaky bucket constrained sessions) is the same as the upper bound on delay given by PGPS [17, 18, 19, 20]. PGPS is Parekh and Gallager's method for computing delay bounds under Weighted Fair Queueing [4].

An important problem we address is that, in general, an upper bound on delay will grow linearly with the connection length. For example, in VirtualClock and PGPS, the value $(N-1)\,L_{max,s}/r_s$ is part of the upper bound on delay of a session s, where N is the number of server nodes in the session's connection, $L_{max,s}$ is the

This work was supported in part by a scholarship from CAPES and UFRJ (Brazil), and by grants from NASA and NSF.

SIGCOMM '95 Cambridge, MA USA
© 1995 ACM 0-89791-711-1/95/0008...$3.50

maximum packet length of the session, and r_s is its reserved rate in the network. For this reason primarily (but also for others), it is useful to allow some form of adjustment to a session's delay, e.g. possibly trading off the delay of that session with those of others that happen to share some of the same links and nodes. As part of the Leave-in-Time service discipline we present a set of admission control algorithms that support the notion of *delay shifting*. Delay shifting allows the delay bounds of some sessions to be decreased at the expense of increasing those of other sessions. The admission control algorithms are based on an intuitive framework of priority classes within which delay shifting is done in a systematic way.

The remainder of this paper is structured as follows. The Leave-in-Time service discipline is presented in Section 2. In Section 3, the service guarantees of Leave-in-Time are analyzed through simulated experiments. In Section 4, the Leave-in-Time service discipline is compared with other service disciplines. Section 5 is a summary of the paper's contributions. All results are presented without proof to simplify exposition and due to lack of space. See [8] for all proofs.

2 The Leave-in-Time Service Discipline

The base packet scheduling algorithm of Leave-in-Time emulates, for each session, the service provided by a fixed-rate server. A session s reserves a rate r_s in a Leave-in-Time server, and the server provides the session with service no worse (i.e. the emulation error is bounded and small) than the service that would be provided by a fixed-rate server with rate r_s. We call this fixed-rate server the session's *reference server.*

The Reference Server

The Leave-in-Time service discipline requires explicit bandwidth reservation at connection establishment time. Suppose that a session reserves a rate r_s in a Leave-in-Time server, where s is an identifier for the session. Define the *reference server* of this session (see Figure 1) to be a work-conserving FCFS server with rate r_s along with the restriction that the session is served alone (i.e. no other session sharing the reference server).

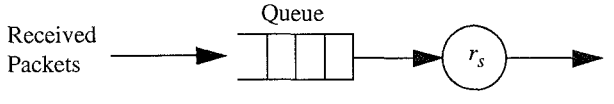

Figure 1: The reference server of session s is a work-conserving FCFS server with rate r_s.

Suppose that session s is being served by its reference server, and that packets of the session are numbered in increasing order as they arrive (the first packet being packet 1). The following definitions apply: $t_{i,s}$ is the arrival time of packet i in the reference server, where a packet has arrived only after its last bit has arrived, $L_{i,s}$ is the length of packet i of the session, $W_{i,s}$ is the finishing transmission time of packet i in the reference server (i.e. the time the last bit of packet i leaves the reference server), and $D^{ref}_{i,s}$ is the delay of packet i in the reference server, i.e. $D^{ref}_{i,s} = W_{i,s} - t_{i,s}$. $W_{i,s}$ is related to $t_{i,s}$ and $L_{i,s}$ by the following equation (proof in [7]):

$$W_{i,s} = \max\{t_{i,s}, W_{i-1,s}\} + \frac{L_{i,s}}{r_s}, i \geq 1, \quad (1)$$

where $W_{0,s} = t_{1,s}$.

The Leave-in-Time Service Discipline

We present the Leave-in-Time service discipline as a construction in three steps: a base server algorithm based on the reference server, and two generalizations that result in the final version.

Base Algorithm: Consider a service discipline that assigns to each received packet a *transmission deadline*, and serves packets from all sessions in increasing order of transmission deadline (ties are ordered arbitrarily), where the transmission deadline $F_{i,s}$ of packet i of session s is assigned a value equal to the finishing transmission time this packet would have if session s were served by its reference server. Thus, the transmission deadline $F_{i,s}$ of packet i of session s can be calculated using equation (1), that is:

$$F_{i,s} = \max\{t_{i,s}, F_{i-1,s}\} + \frac{L_{i,s}}{r_s}, i \geq 1, \quad (2)$$

where $F_{0,s} = t_{1,s}$.

The reader will note that this is identical to VirtualClock's packet-scheduling time calculation. This service discipline is work-conserving, since it is always busy (i.e. transmitting packets) when there are queued packets at the server.

First Generalization: The base algorithm is generalized by allowing it to work in a non-work-conserving mode. Non-work-conserving service disciplines can generally provide lower variance in delay (or delay jitter) than with work-conserving ones [28]. Thus, packets are not necessarily immediately available for transmission upon arrival, and thus arrived packets may be delayed before being queued for transmission. The time a packet joins the server transmission queue is called the *eligibility time* of the packet. Although the server is non-work-conserving, it is never idle when there are eligible packets to serve.

The non-work-conserving service discipline has two components (see Figure 2): a set of delay regulators that hold packets until their eligibility times, and a server transmission queue. The use of delay regulators to shape the traffic pattern to reduce delay jitter is based on Jitter-EDD [22]. A session desiring delay jitter control (i.e. delay jitter reduction) is assigned a delay regulator. A session not desiring delay jitter control has all of its packets sent directly to the server queue upon arrival, i.e. the eligibility time equals the arrival time.

The non-work-conserving service discipline calculates the transmission deadline $F_{i,s}$ of a packet as:

$$F_{i,s} = \max\{E_{i,s}, F_{i-1,s}\} + \frac{L_{i,s}}{r_s}, i \geq 1, \quad (3)$$

where $F_{0,s} = t_{1,s}$, and $E_{i,s}$ is the eligibility time of packet i of session s. $E_{i,s}$ is defined in the final version of the Leave-in-Time service discipline.

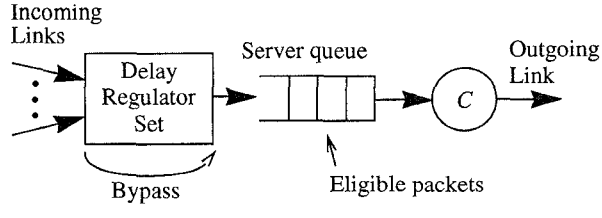

Figure 2: A Leave-in-Time server has an outgoing link with capacity C and a set of delay regulators (one for each session desiring delay jitter control). A delay regulator delays packets of a session in order to shape its traffic pattern. A session will use a delay regulator only if delay jitter control is desired.

Second Generalization: The next generalization is obtained by allowing $L_{i,s}/r_s$ to be replaced by $d_{i,s}$ (a delay for packet i at a server node, which can be customized according to a session's service needs), which splits equation (3) into the following:

$$F_{i,s} = \max\{E_{i,s}, K_{i-1,s}\} + d_{i,s}, i \geq 1, \text{ and} \quad (4)$$

$$K_{i,s} = \max\{E_{i,s}, K_{i-1,s}\} + \frac{L_{i,s}}{r_s}, i \geq 1, \quad (5)$$

where $K_{i,s}$ allows the partitioning of equation (3) into equations (4) and (5), and $K_{0,s} = t_{1,s}$.

As discussed before, the term $(N-1)L_{max,s}/r_s$ is part of the upper bound on delay of VirtualClock and PGPS. In VirtualClock, this term originates from the term $L_{i,s}/r_s$ of equation (2) (the upper bound on delay for VirtualClock is given in [7]). The second generalization replaces $L_{i,s}/r_s$ by $d_{i,s}$ which allows some reduction of the upper bound on delay, i.e. by allowing a smaller value than $L_{i,s}/r_s$ to be assigned to $d_{i,s}$. Thus, this generalization allows control over the upper bounds on the end-to-end delays that sessions experience.

A session's route is composed of Leave-in-Time servers in tandem. Without loss of generality, consider that the path of servers traversed by a session s is numbered from 1 to N (see Figure 3). Since some measures vary with the server node, we need to extend our notation. Thus, $F_{i,s}$, $K_{i,s}$, $t_{i,s}$, $E_{i,s}$, and $d_{i,s}$ are written as $F^n_{i,s}$, $K^n_{i,s}$, $t^n_{i,s}$, $E^n_{i,s}$, and $d^n_{i,s}$, respectively, where n identifies the server node.

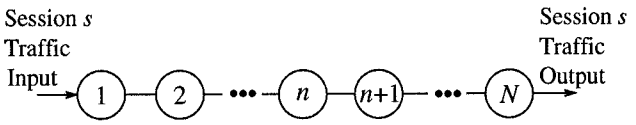

Figure 3: The route of session s traffic is composed of N Leave-in-Time servers in tandem. The path of servers traversed by session s is numbered from 1 to N.

Final Version: We are now ready to present the Leave-in-Time service discipline which works as follows:

(1) Each received packet is assigned an eligibility time and a transmission deadline. The eligibility time of packet i of session s at server node n is defined as:

$$E^n_{i,s} = t^n_{i,s} \text{ (for sessions without delay jitter control), and} \quad (6)$$

$$E^n_{i,s} = t^n_{i,s} + A^n_{i,s} \text{ (for sessions with delay jitter control),} \quad (7)$$

where $A^n_{i,s}$ is the *holding time* of packet i of session s in a delay regulator. $A^n_{i,s}$ is defined as:

$$A^1_{i,s} = 0, \quad (8)$$

$$A^n_{i,s} = F^{n-1}_{i,s} + \frac{L_{MAX}}{C_{n-1}} - \hat{F}^{n-1}_{i,s} + d^{n-1}_{max,s} - d^{n-1}_{i,s}, n > 1, \quad (9)$$

where $\hat{F}^{n-1}_{i,s}$ is the actual finishing transmission time of packet i of the session at server node n - 1, L_{MAX} is the maximum packet length allowed in the network, $d^{n-1}_{max,s}$ is equal to $max\{d^{n-1}_{i,s}: i \geq 1\}$, and C_{n-1} is the capacity of the outgoing link of server node n - 1. As in Jitter-EDD [22], the holding time $A^n_{i,s}$ calculated at server node n - 1 is transmitted in the packet's header to node n. Note that $A^n_{i,s}$ is always positive, and that $\hat{F}^n_{i,s} < F^n_{i,s} + L_{MAX}/C_n$ (both proven in [8]). Thus, the term $F^{n-1}_{i,s} + L_{MAX}/C_{n-1} - \hat{F}^{n-1}_{i,s}$ in (9) tries to eliminate the delay jitter caused by the variation in the finishing transmission times of packets at server node n - 1. We will see later that Leave-in-Time allows $d^{n-1}_{i,s}$ to vary according to the packet's length. Thus, the term $d^{n-1}_{max,s} - d^{n-1}_{i,s}$ in (9) tries to eliminate the delay jitter caused by the variation in the values $d^{n-1}_{i,s}$ of packets.

The transmission deadline of packet i of session s at server node n is calculated as (these are just equations (4) and (5) with the identification of the server node):

$$F^n_{i,s} = \max\{E^n_{i,s}, K^n_{i-1,s}\} + d^n_{i,s}, i \geq 1, \text{ and} \quad (10)$$

$$K^n_{i,s} = \max\{E^n_{i,s}, K^n_{i-1,s}\} + \frac{L_{i,s}}{r_s}, i \geq 1, \quad (11)$$

where $K^n_{0,s} = t^n_{1,s}$.

(2) Eligible packets from all sessions are served in increasing order of transmission deadline (ties are ordered arbitrarily).

Service Commitments Provided by Leave-in-Time

This section summarizes the service commitments, i.e. performance guarantees, provided by the Leave-in-Time service discipline.

Upper Bound on End-to-End Delay

$$D^{1,N}_{max,s} < D^{ref}_{max,s} + \beta^{1,N}_s + \alpha^N_s, N \geq 1, \quad (12)$$

where $D^{1,N}_{max,s}$ is the upper bound on end-to-end delay that packets of session s experience in a route composed of N Leave-in-Time servers, $D^{ref}_{max,s}$ is the upper bound on delay that packets of session s would experience if session s were served by its reference server (i.e. a fixed-rate server), $\beta^{1,N}_s$ is a constant equal to

$$\beta^{1,N}_s = \sum_{n=1}^{N}\left(\frac{L_{MAX}}{C_n} + \Gamma_n\right) + \sum_{n=1}^{N-1} d^n_{max,s}, N \geq 1, \quad (13)$$

where Γ_n is the propagation time for the outgoing link of server n, and α^N_s is a constant equal to

$$\alpha^N_s = \max\{d^N_{i,s} - \frac{L_{i,s}}{r_s} : i \geq 1\}.$$

Note that an upper bound on end-to-end delay exists if the session has an upper bound on delay in its reference server. For a session conforming to a token bucket filter $(r_s, b_{0,s})$,[1]

$$D^{ref}_{max,s} = b_{0,s}/r_s. \quad (14)$$

1. A token bucket filter is characterized by two parameters, a rate r and the maximum number of tokens (b_0) the bucket can store Initially, the bucket has b_0 tokens (a full bucket). New tokens are continuously filling up the bucket at rate r. All tokens exceeding the maximum bucket capacity are discarded. A session's traffic conforms to a token bucket filter $(r_s, b_{0,s})$ (here we adopt a notation similar to the one used in [1], adding an identifier for the session with the subscript s) if, for every generated packet, $L_{i,s}$ tokens are removed from the bucket, where $L_{i,s}$ is the length of the packet, and the bucket size is never negative (i.e. there are always enough tokens to be removed when a packet is generated).

Thus,

$$D^{1,N}_{max,s} < \frac{b_{0,s}}{r_s} + \beta^{1,N}_s + \alpha^N_s, N \geq 1, \quad (15)$$

for a session conforming to a token bucket filter (r_s, $b_{0,s}$). This result is the same as that found using PGPS [17, 18, 19, 20] (see equation (4.36) in [17], or equation (23) in [19]) and for Leave-in-Time with admission control procedure 1 (which we define later) with one class and $d^n_{i,s} = L_{i,s}/r_s$, since in this case α^N_s is zero and $d^n_{max,s} = L_{max,s}/r_s$.

Upper Bound on the End-to-End Delay Distribution

$$P(D^{1,N}_{i,s} > d) \leq P(D^{ref}_{i,s} > d - \beta^{1,N}_s - \alpha^N_s), N \geq 1, \quad (16)$$

where $P(D^{1,N}_{i,s} > d)$ denotes the probability that delay $D^{1,N}_{i,s}$ is larger than d, $D^{1,N}_{i,s}$ is the end-to-end delay packet i of session s experiences in the connection, $P(D^{ref}_{i,s} > d - \beta^{1,N}_s - \alpha^N_s)$ denotes the probability that delay $D^{ref}_{i,s}$ is larger than $d - \beta^{1,N}_s - \alpha^N_s$, and $D^{ref}_{i,s}$ is the delay of packet i in the reference server of the session. This inequality is a function of the probability distribution of delays of packets of the session in its reference server, i.e. a fixed-rate server, which is well-studied [14, 23]. This inequality says that an upper bound on the probability distribution of delays is obtained by shifting the probability distribution of end-to-end delays of the session in a fixed-rate server to the right by the constant $\beta^{1,N}_s + \alpha^N_s$ (see Figure 4).

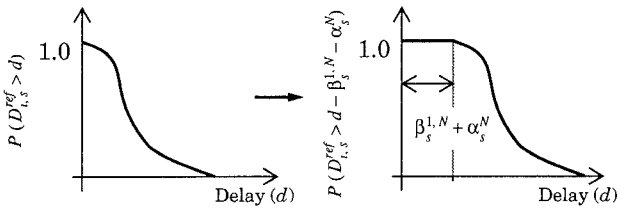

Figure 4: The upper bound on the probability distribution of end-to-end delays is obtained by shifting the probability distribution of delays of the session in a fixed-rate server to the right by a constant.

Upper Bound on End-to-End Delay Jitter

Define the end-to-end delay jitter $J^{1,N}_s$ of session s traversing servers 1 to N as the maximum difference between the delays experienced by any two packets from session s (this is the same definition as the one used in [22]). This definition implies that $D^{1,N}_{max,s}$ must be finite (i.e. the session has an upper bound on end-to-end delay). If $D^{1,N}_{max,s}$ is finite,

$$J^{1,N}_s < D^{ref}_{max,s} + \Delta^{1,N}_{max,s} - d^N_{max,s} + \alpha^N_s, N \geq 1,$$

for sessions without delay jitter control, and

$$J^{1,N}_s < D^{ref}_{max,s} + \delta^N_{max,s} - d^N_{max,s} + \alpha^N_s, N \geq 1, \quad (17)$$

for sessions with delay jitter control, where

$$\Delta^{1,N}_{max,s} = \sum_{n=1}^{N} \delta^n_{max,s},$$

and

$$\delta^n_{max,s} = \frac{L_{MAX}}{C_n} + d^n_{max,s} - \frac{L_{min,s}}{C_n}.$$

Thus, the end-to-end delay jitter of sessions is composed of the upper bound on delay of the session in a fixed-rate server ($D^{ref}_{max,s}$) and the delay jitter contribution of individual server nodes which depends on the kind of connection, i.e. with delay jitter control or not. The end-to-end delay jitter of sessions without delay jitter control grows with the connection's length, which does not happen for sessions with delay jitter control.

Upper Bound on Buffer Space Requirements

If $D^{ref}_{max,s}$ is finite, the upper bound on the buffer space used by a session at server n is

$$Q^n_{max,s} < r_s\left(D^{ref}_{max,s} + \Delta^{1,n-1}_{max,s} + \frac{L_{MAX}}{C_n} + d^n_{max,s}\right), n \geq 1,$$

for sessions without delay jitter control, and

$$Q^n_{max,s} < r_s\left(D^{ref}_{max,s} + \delta^{n-1}_{max,s} + \frac{L_{MAX}}{C_n} + d^n_{max,s}\right), n \geq 1,$$

for sessions with delay jitter control, where $\delta^0_{max,s} = \Delta^{1,0}_{max,s} = 0$.

As with all the other service commitments of Leave-in-Time, the upper bound on buffer space is a function of the delay of the session in a fixed-rate server, the session's reference server. This delay is completely determined by the dynamic traffic behavior of the session. See [6] for an upper bound on the buffer space *probability distribution*.

The Admission Control Procedures

In order for the Leave-in-Time service discipline to provide sessions with service commitments that are independent of the traffic behavior of other sessions, sessions must satisfy some traffic requirements as verified by *admission control tests*. A session's connection is established if the admission control tests are satisfied in all the nodes along the session's route.

Since the following discussions apply to all the nodes in a session's connection, we drop the identification of the server node n. Thus, $d^n_{i,s}$ and C_n will be written as $d_{i,s}$ and C, respectively.

The value assigned to $d_{i,s}$ is not part of the traffic characterization of a session, but is a service parameter. Inequality (12) shows that the values assigned to $d_{i,s}$ along a session's route are responsible in large part for the upper bound on the end-to-end delay of the session. Thus, one would prefer to assign the lowest possible value to $d_{i,s}$. Unfortunately, assigning arbitrary values to $d_{i,s}$ may lead to *scheduler saturation*. This happens when a server is not able to provide an upper bound on the interval of time between the transmission deadline of a packet and its actual end of transmission. Therefore, an *admission control procedure* must regulate the minimum values that can be assigned to $d_{i,s}$ in order to avoid scheduler saturation, i.e. the admission control procedure limits how short the upper bound on end-to-end delay of a session can be. For this reason, we present a set of admission control procedures which allow some flexibility on the assignment of values to $d_{i,s}$. Flexibility is obtained by allowing some sessions to have assigned lower values to $d_{i,s}$ at the expense of other sessions that must be assigned higher values. We define here only the admission mechanism. The decision of *which* sessions should be granted lower delays is a policy decision.

We developed three admission control procedures which differ in degree of flexibility and computational complexity. Admission control procedures 1 and 2 support the assignment of sessions to *delay classes*, where sessions assigned to lower numbered classes get lower $d_{i,s}$ values than sessions in higher numbered classes. The differences between admission control procedures 1 and 2 are discussed later, since they depend on details not yet defined. Admission control procedure 3 allows the assignment of arbitrary values to $d_{i,s}$, which makes it the most flexible of the three procedures. This flexibility has a cost, however. In order to avoid scheduler saturation, a server may not be able to commit all its available bandwidth.

All three admission control procedures share a common admission control test: All sessions must reserve their required lower bound on bandwidth at connection establishment time. The rate reservation is constrained by the following inequality:

$$\sum_{j \in \varphi} r_j \leq C, \tag{18}$$

where C is the capacity of the outgoing link of the Leave-in-Time server, and φ is the set of sessions traversing the server.

Admission Control Procedure 1: Suppose we divide the sessions traversing a Leave-in-Time server into P classes numbered from 1 to P. Class k is defined by a pair of values (R_k, σ_k), where R_k is the maximum bandwidth that may be allocated to sessions in class k, and σ_k is the *base delay* of class k (as defined below), where $R_k \geq R_j$ and $\sigma_k \geq \sigma_j$, $k > j$, and $R_P = C$. For a session s_a to be admitted into class j, the following tests must be satisfied:

$$(1.1)\ R_m \geq \sum_{s \in \bigcup_{l \leq m} \Omega_l} r_s, \text{ for } m = \{j, j+1, \ldots, P\},$$

$$(1.2)\ \sigma_m \geq \sum_{s \in \bigcup_{l \leq m} \Omega_l} \frac{L_{max,s}}{C}, \text{ for } m = \{j, j+1, \ldots, P\text{-}1\}, \text{ and}$$

$$(1.3)\ d_{i,s_a} = \frac{L_{i,s_a} \cdot R_j}{r_{s_a} \cdot C} + \sigma_{j-1} + \varepsilon_{s_a}, \text{ (define } \sigma_0 = 0),$$

where $\Omega_l = \{s$: session s is in class $l\}$, and $\varepsilon_{s_a} \geq 0$ is a constant (defined below) for session s_a. Figure 5 shows how the classes relate to each other.

Admission control procedure 1 also allows rule (1.3) to be written as

$$(1.3a)\ d_{i,s_a} = \frac{L_{max,s_a} \cdot R_j}{r_{s_a} \cdot C} + \sigma_{j-1} + \varepsilon_{s_a},$$

where L_{max,s_a} is the maximum length of a packet of session s_a. With this rule, d_{i,s_a} is independent of the length of individual packets.

ε_{s_a} is assigned by the admission control procedure, and in general, will be set to zero since one gains nothing by arbitrarily increasing delay. However, sometimes it is necessary that ε_{s_a} be greater than zero, e.g. if the admission control procedure uses rule (1.3a) (in which case one can have a fixed delay d_{i,s_a} for all packets of a session) and is limited to selecting discrete values for d_{i,s_a}, having the flexibility of selecting non-zero values is necessary. Since this constant must be the same for all packets in the same session, we could not simply express rules (1.3) and (1.3a) as inequalities.

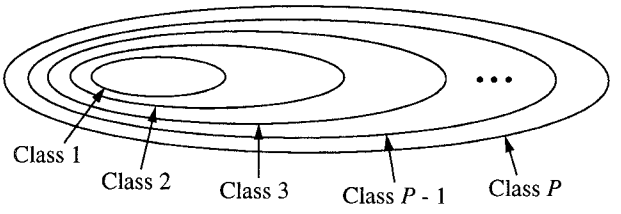

Figure 5: The class hierarchy in admission control procedure 1. A session is in only one class. Class k has a maximum bandwidth of R_k to be assigned to its sessions. The bandwidth of a class k is part of the bandwidth of class $k+1$. This means that class k may lend spare bandwidth to class $j > k$.

Note that sessions allocated to higher numbered classes will experience more delay than sessions in lower numbered classes, and that σ_P (i.e. the base delay of class P) is not used in rules (1.2) and (1.3), which means that the value assigned to σ_P is irrelevant. Also note that, for $P = 1$, i.e. only one class, sessions may have $d_{i,s} = L_{i,s}/r_s$ (i.e. if $\varepsilon_{s_a} = 0$). In this case, equations (10) and (11) can be reduced to

$$F_{i,s} = \max\{E_{i,s}, F_{i-1,s}\} + \frac{L_{i,s}}{r_s}, i \geq 1.$$

With admission control procedure 1, O(P) tests are performed when a session is admitted. Note that $d_{i,s}$ is inversely proportional to r_s. This means that sessions requiring low bandwidth may be imposed with higher delays (i.e. higher values $d_{i,s}$) than sessions requiring high bandwidth. However, with proper selection of values, a session allocated to a low numbered class may still be assigned a reasonable value to $d_{i,s}$. Consider an example in which $R_1 = C/100$. A session allocated to class 1 could have $d_{i,s} = L_{max,s}/(r_s \cdot 100)$ (if we use rule (1.3a)), which would be appropriate even if r_s is small since it is multiplied by a large amount. Note that the ratio R_k/C (in rule (1.3)) defines how much of $L_{max,s}/r_s$ will be considered in the delay $d_{i,s}$. Although this admission control procedure has an undesirable coupling between reserved rate and minimum $d_{i,s}$, it allows the exploitation of the entire bandwidth of the server.

Consider the following more specific example: A Leave-in-Time server with C = 100Mbits/sec and a session with reserved rate equal to 100Kbits/sec and packet length equal to 400bits. Assume admission control procedure 1 with three classes, namely (R_1 = 10Mbits/sec, σ_1 = 0.2ms), (R_2 = 40Mbits/sec, σ_2 = 1.6ms), and ($R_3 = C$ = 100Mbits/sec, σ_3 = 4ms). We could assign this session in class 1, which would lead to $d_{i,s}$ = 0.4ms, or we could assign it in class 2 with $d_{i,s}$ = 1.8ms, or in class 3, with $d_{i,s}$ = 5.6ms. This illustrates how admission control procedure 1 provides flexibility in the assignment of the values $d_{i,s}$ (i.e. delay).

Admission Control Procedure 2: Admission control procedure 2 uses the same scheme as admission control procedure 1, with the exception that it changes rules (1.2), (1.3), and (1.3a) to:

$$(2.2)\ \sigma_m \geq \sum_{s \in \bigcup_{i \leq m} \Omega_i} \frac{L_{max,s}}{C}, \text{ for } m = \{j, j+1, \ldots, P\}, \text{ and}$$

$$(2.3)\ d_{i,s_a} = \frac{L_{i,s_a} \cdot R_{j-1}}{r_{s_a} \cdot C} + \sigma_j + \varepsilon_{s_a}, \text{ (consider } R_0 = 0),$$

$$(2.3a)\ d_{i,s_a} = \frac{L_{max,s_a} \cdot R_{j-1}}{r_{s_a} \cdot C} + \sigma_j + \varepsilon_{s_a}, \text{ (consider } R_0 = 0),$$

respectively, where j is the class to which session s_a is being admitted, and $\varepsilon_{s_a} \geq 0$ is a constant for session s_a, as defined in admission control procedure 1.

Note that σ_P (i.e. the base delay of class P) is now used. As in admission control procedure 1, O(P) tests are performed when a session is admitted. In admission control procedure 2, σ_P (i.e. the base delay of class P) must be allocated a value large enough such that all the bandwidth of the server can be exploited. This condition does not apply to admission control procedure 1, since it does not enforce rule (1.2) on class P.

Consider admission control procedure 2 with the same conditions we assumed for the example in admission control procedure 1, i.e. a Leave-in-Time server with C = 100Mbits/sec, a session with reserved rate equal to 100Kbits/sec and packet length equal to 400bits, and three classes, namely (R_1 = 10Mbits/sec, σ_1 = 0.2ms), (R_2 = 40Mbits/sec, σ_2 = 1.6ms), and ($R_3 = C$ = 100Mbits/sec, σ_3 = 4ms). With admission control procedure 2 we could assign this session in class 1, which would lead to $d_{i,s}$ = 0.2ms, or we could assign it in class 2 with $d_{i,s}$ = 2.0ms, or in class 3, with $d_{i,s}$ = 5.6ms. With admission control procedure 1, these values were 0.4ms, 1.8ms, and 5.6ms, respectively. With the same conditions, consider a session with reserved rate equal to 10Kbits/sec and packet length equal to 400bits. This session would have $d_{i,s}$ = 4ms in class 1 with admission control procedure 1, and it would have $d_{i,s}$ = 0.2ms in class 1 with admission control procedure 2. This example shows a major difference between admission control procedures 1 and 2: d_{i,s_a} does not depend on $L_{i,s_a}/r_{s_a}$ in rule (2.3) (or $L_{max,s_a}/r_{s_a}$ in rule (2.3a)) in class 1 with admission control procedure 2. This feature can be used to reduce the delay of low rate sessions.

Admission Control Procedure 3: $d_{i,s} = d_s$, where d_s is a constant, and

$$C \geq \frac{\sum_{s \in A} L_{max,s} \cdot \sum_{s \in A} r_s}{\sum_{s \in A} r_s \cdot d_s}, \text{ for any non-empty set } A \subseteq \varphi, \quad (19)$$

where φ is the set of sessions traversing the server.

Although feasible, this admission constraint requires a large number of tests, since there are $2^{|\varphi|} - 1$ sets A to be tested. Note that admission control procedure 2 with one class (i.e. P = 1) and ε_s = 0 for all sessions is equivalent to admission control procedure 3 when all sessions are assigned the same value to the constant d_s, since rule (2.2) in admission control procedure 2 implies inequality (19) in this case.

Under admission control procedures 1 and 2, sessions may be assigned $d_{i,s}$ values that are larger or smaller than $L_{i,s}/r_s$, which is a value that may be assigned to $d_{i,s}$ in admission control procedure 1 with one class. However, some sessions must be assigned values larger than $L_{i,s}/r_s$ in order to allow others to receive lower ones. Admission control procedures 1 and 2 allow complete exploitation of the bandwidth of the server, since the values assigned to $d_{i,s}$ are implicitly "pre-allocated," while admission control procedure 3 may lead to incomplete usage of bandwidth, since $d_{i,s}$ may be assigned arbitrary small values.

3 Simulations of Leave-in-Time

In this section, we evaluate the service guarantees of the Leave-in-Time service discipline through simulated experiments. We evaluate here the following service guarantees: upper bound on delay, upper bound on delay jitter (with and without delay jitter control), upper bound on the probability distribution of delays, and upper bound on buffer space.

Traffic Source Models

In this work, we chose three kinds of traffic sources to exemplify the performance guarantees of a Leave-in-Time network: ON-OFF, Poisson, and Deterministic (i.e. fixed packet rate source model) sources. ON-OFF sources have been used extensively in recent studies [25, 29], since they can be used to model standard voice sources. Poisson sources are used in our simulations to examine the firewall property of Leave-in-Time, i.e. that the service guarantees of a session are independent of the behavior of other sessions traversing the network. Poisson sessions are also used to show that the service bounds offered by Leave-in-Time are not loose in the sense that they must predict a session's guarantees in the presence of *any* kind of traffic sharing the network. However, note that we are not implying that the behavior of the combined traffic in a network link is Poisson. Deterministic sources are used in experiments where we want to commit all the bandwidth of a server. All traffic sources in our simulations have packet length of 424bits, the length of an ATM packet.

ON-OFF Traffic Sources: An ON-OFF traffic source is modeled here as a two-state Markov modulated process. In the ON state, packets are generated at fixed intervals of time T. In the OFF state, no packet is generated. The durations of the ON and OFF states are exponentially distributed with means a_{ON} and a_{OFF}, respectively. The number of packets generated in the ON state is approximated by a geometric distribution with mean a_{ON}/T.

We simulate ON-OFF sources with a_{ON} = 352ms and a_{OFF} varying from 6.5ms to 650ms (more specifically, we use 6.5, 18.5, 39.1, 88.0, 150.9, 288.0, and 650ms), which cover traffic sources that resemble from fixed packet rate sources (which have a_{OFF} = 0ms) to standard voice sources (i.e. when a_{OFF} = 650ms). These values are the same as the ones used in [25]. In our simulations, T = 13.25ms, which implies that the generation rate is 32kbits/s in the ON state (since the packet length is 424bits). All ON-OFF sessions reserve a rate of 32kbits/s in the network.

Poisson Traffic Sources: The interarrival time of packets of these traffic sources is exponentially distributed with mean a_P (specific values depend on the experiment). The reserved rate of a Poisson session is at least $424/a_P$.

Deterministic Traffic Sources: The interarrival time of packets of these traffic sources is a *constant* a_D = 13.25ms. These sessions reserve a rate of 32kbits/s.

Network Topology

Our simulations use the network topology of Figure 6. It has five server nodes in tandem and links with a capacity of 1536kbits/s (T1 capacity) and a propagation delay of 1ms (light takes approximately this time to traverse 200km of optical fiber). In Figure 6, numbers represent server nodes and letters identify entrance (i.e. where traffic is generated) or exit (i.e. where traffic is consumed) points in the network. Traffic flows left to right.

To simplify our discussions, we will use pairs of letters to identify a session's route. Thus, the route of a session traversing server nodes 1 to 5 (five hops) will be identified as route a-j.

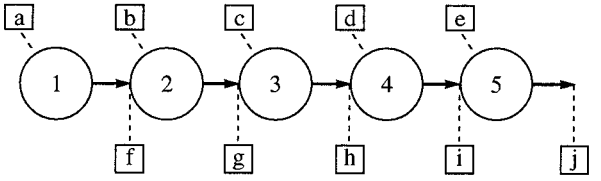

Figure 6: Network Topology.

We simulated two basic *traffic configurations* which we call MIX and CROSS. The MIX traffic configuration has 10 sessions in each one of the routes a-j, b-g, c-h, and d-i, 16 sessions in each one of the routes a-f and e-j, 8 sessions in each one of the routes a-h, c-j, a-g, and d-j, and 6 sessions in each of the routes a-i and b-j, which total 10 five-hop sessions, 8 four-hop sessions, 16 three-hop sessions, 16 two-hop sessions, and 62 one-hop sessions. The CROSS traffic configuration uses routes a-j, a-f, b-g, c-h, d-i, and e-j, which are all one-hop routes with the exception of a-j, which is a five-hop route. The number of sessions in each route of a CROSS configuration will differ in some simulations.

A simulated experiment will be identified by the name of the traffic configuration and the kinds of sessions in each route. When possible, routes will be identified simply by their number of hops. In the CROSS configuration, one-hop sessions will be called the *cross traffic*. All the results we show in this section refer to five-hop sessions, i.e. over route a-j.

Simulation Results

We simulated the Leave-in-Time service discipline to illustrate the various performance guarantees provided. Experiments using admission control procedures 1 and 2 are presented. For more detailed results, see [6].

Admission Control Procedure 1 with One Class

We simulated Leave-in-Time using admission control procedure 1 with one class, i.e. with R_1 = 1536kbits/s.

End-to-End Delay

Figure 7 shows the maximum delay and the delay jitter of an ON-OFF five-hop session (without delay jitter control) in the MIX traffic configuration in a 5 minute run of the network. All sessions in this simulation are ON-OFF sessions with identical parameters: average off period, a_{OFF}, varying from 6.5ms to 650ms. This experiment shows that this kind of traffic load exhibits low delay (when compared to the calculated upper bound) and that the utilization factor, which varies from 35.1% to 98.2% (for an average off period from 650ms to 6.5ms, respectively), does not have much influence on the maximum delay of the sessions.

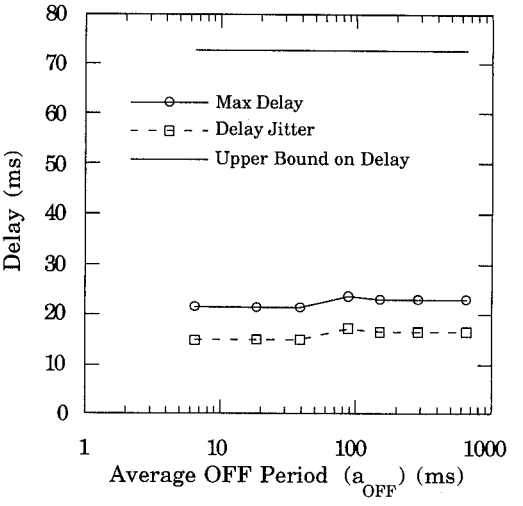

Figure 7: The maximum delay and delay jitter of a five-hop ON-OFF session in a MIX traffic configuration of ON_OFF sessions.

End-to-End Delay Jitter

Figure 8 demonstrates the effectiveness of delay jitter control. This is a 10 minute run of the network for a CROSS traffic configuration. Figure 8 shows the delay distribution of two ON-OFF five-hop sessions (one with delay jitter control and the other without delay jitter control) with one Poisson cross traffic (i.e. one Poisson traffic for each one-hop route in the CROSS configuration). The ON-OFF sessions have a_{OFF} = 650ms and the Poisson sessions have a_P = 0.28804ms and a reserved rate of 1472kbits/s. This experiment shows a reduction in delay jitter from 59.7ms (the upper bound is 66.25ms) for the session without delay jitter control to 12.4ms (the upper bound is 13.25ms) for the session with delay jitter control. Note that delay jitter control increases the average delay of packets, since the delay jitter control mechanism reduces delay jitter by forcing packets to experience a delay closer to the upper bound on delay.

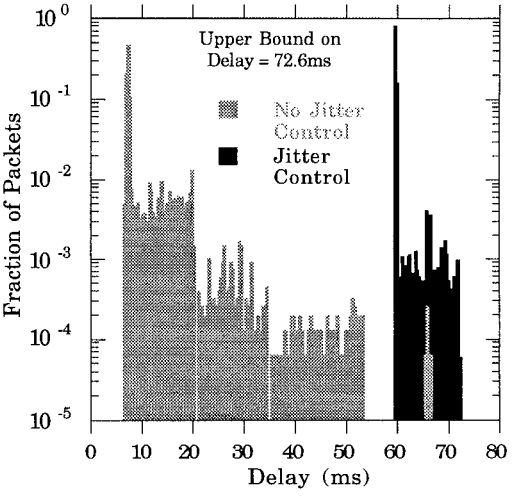

Figure 8: Delay distribution of two ON-OFF sessions with Poisson cross traffic.

Probability Distribution of End-to-End Delays

Figures 9 to 11 compare the obtained probability distribution of delays to the analytical upper bound on the probability distribution of delays. To make this comparison interesting, we needed sessions that would provide an unbounded delay distribution, and that would be amenable to analysis to be able to calculate the analytical upper bound. For these reasons, we chose Poisson sessions.

The analytical upper bound is calculated using inequality (16). For this calculation, we need to know the probability distribution of delays of the session in its reference server, which is equivalent to a M/D/1 system in these experiments. We calculated the probability distribution of delays in a M/D/1 system following the results presented in [16, 21].

We also show in Figures 9 to 11 the upper bound on the probability distribution which is obtained when we use inequality (16) with the results obtained with a simulation of a fixed-rate server serving our traffic. This "simulated upper bound" is provided to show how one can obtain an estimate for the upper bound on the delay distribution in the network even for sessions that are not amenable to analysis.

Figure 9 shows the results of an experiment with a five-hop Poisson session with a_p = 1.5143ms and reserved rate of 400kbits/s (which represents a utilization factor of 0.7) in a CROSS traffic configuration with one Poisson cross traffic (i.e. one Poisson traffic for each one-hop route in the CROSS configuration) in a 10 minute run of the network. Each Poisson cross traffic has a_p = 0.3929ms and reserved rate of 1136kbits/s.

This experiment shows that the analytical upper bound can be used to estimate percentiles. For example, using the analytical curve, one can estimate that about 0.01% of all packets are delayed by more than 26ms. From the experimentally obtained distribution, the delay is about 23ms for this case.

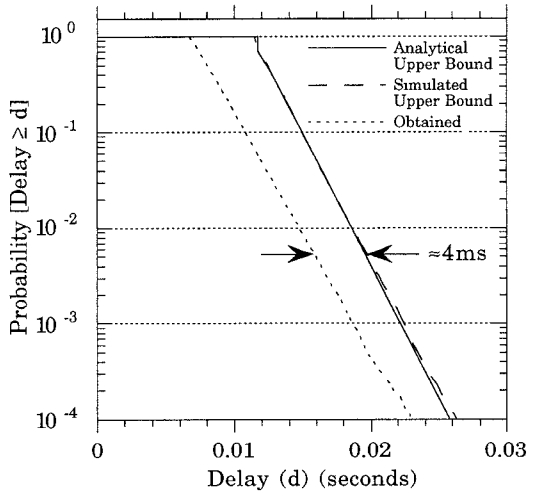

Figure 9: Probability distribution of delays of a Poisson session (utilization factor of 0.7) with Poisson cross traffic

Figure 10 is an experiment similar to the previous one except that the five-hop Poisson session has a_p = 40ms and reserved rate of 32kbits/s (which represents a utilization factor of 0.33) and each Poisson cross traffic has a reserved rate of 1472kbits/s and a_p = 0.28804ms.

This experiment shows that, for a low reserved rate, the analytical upper bound on the probability of delays can be loose (this happens in part because the constant $\beta_s^{1,N}$ in (12) increases for a low rate session). However, this is not always the case for a session with low reserved rate. Consider an experiment in which each one-hop cross traffic is a set of 47 32kbits/s Deterministic sessions. The result of this experiment is shown in Figure 11.

In general, the worst case bounds will often be somewhat loose in a specific and limited time experiment, since worst case bonds consider all possible scenarios (no matter how unlikely they may be).

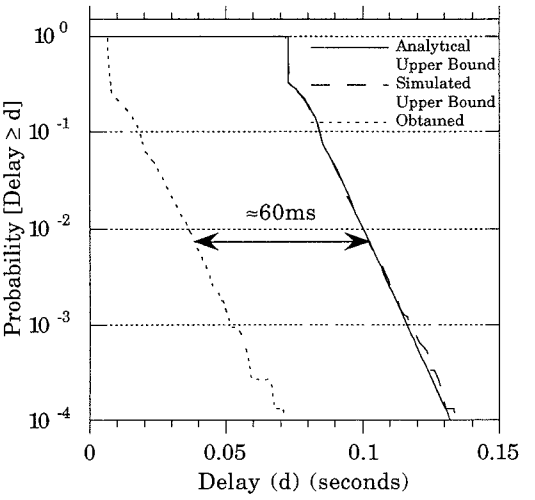

Figure 10: Probability distribution of delays of a Poisson session (utilization factor of 0.33 -- reserved rate of 32kbits/s) with Poisson cross traffic.

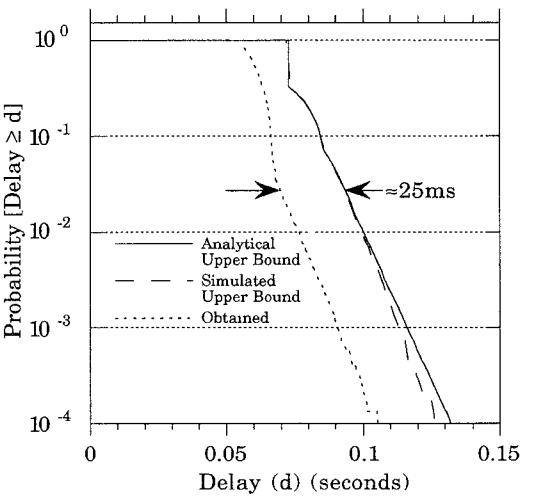

Figure 11: Probability distribution of delays of a Poisson session (utilization factor of 0.33 -- reserved rate of 32kbits/s) with Deterministic cross traffic.

Buffer Space Requirements

Figures 12 and 13 show the distribution of buffer space used by a session without delay jitter control and a session with delay jitter control, respectively. These results were obtained from the same experiment used to generate Figure 8, i.e. a 10 minute run of the network for a CROSS traffic configuration with two ON-OFF

five-hop sessions (32Kbits/s and a_{OFF} = 650ms) and Poisson cross traffic (1472kbits/s and a_p = 0.28804). The buffer space measurement in our experiment is always a multiple of 424 bits (the packet length), since we measure the buffer space at the moment the last bit of a packet arrives at a server node, and we consider the packet under transmission in our total. This explains the staircase shape of the obtained distributions.

Figures 12 and 13 show the buffer space and the calculated upper bound at the first and last server nodes in the session's route. Note that the observed maximum value is below the calculated upper bound by at most about 2 packet lengths for both sessions.

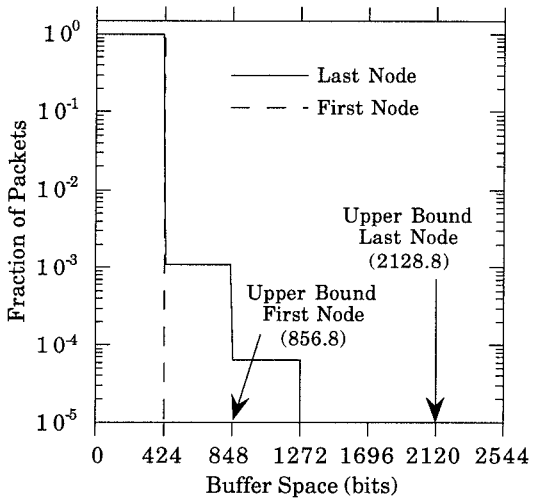

Figure 12: Buffer space of an ON-OFF session *without* delay jitter control with Poisson cross traffic.

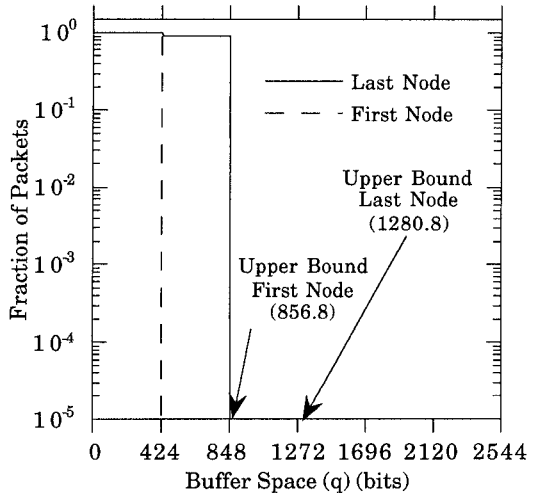

Figure 13: Buffer space of an ON-OFF session *with* delay jitter control with Poisson cross traffic.

Admission Control Procedure 2 With Two Classes

We now simulate admission control procedure 2 with two classes: class 1 with R_1 = 640kbits/s and σ_1 = 2.77ms, and class 2 with R_2 = 1536kbits/s and σ_2 = 13.25.

Figures 14 to 17 refer to a single experiment in which we measure the maximum delay and the delay jitter of four five-hop sessions: two sessions in class 1 (one with delay jitter control and the other without delay jitter control) and two sessions in class 2 (one with delay jitter control and the other without delay jitter control). The sessions in class 1 have $d_{i,s}$ = 2.77ms (10 sessions: 5 five-hop and 5 four-hop sessions), and the sessions in class 2 have $d_{i,s}$ = 18.8ms (all the other sessions). The experiment is a MIX traffic configuration with ON-OFF sessions with a_{OFF} varying from 6.5ms to 650ms in a 5 minute run of the network.

Figures 14 to 17 show the advantage of the class hierarchy, i.e. sessions in low numbered classes experience less delay (and delay jitter) than sessions in high numbered classes.

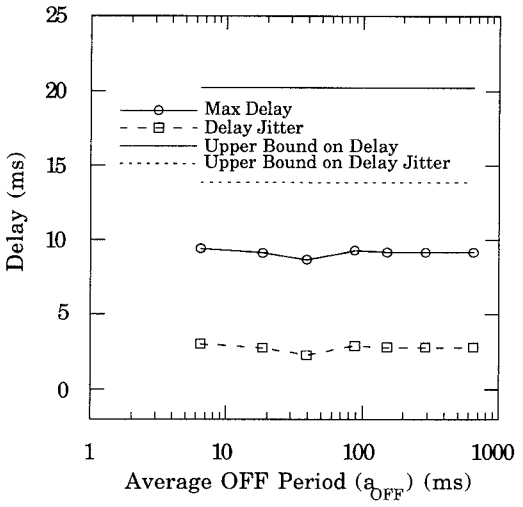

Figure 14: The maximum delay and delay jitter of a session *without* delay jitter control in class 1 in a MIX traffic configuration of ON_OFF sessions.

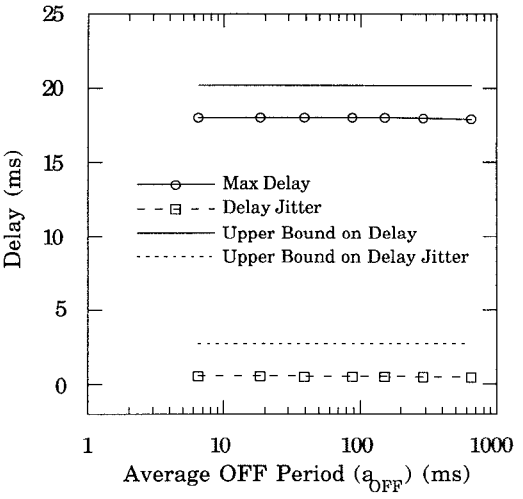

Figure 15: The maximum delay and delay jitter of a session *with* delay jitter control in class 1 in a MIX traffic configuration of ON_OFF sessions.

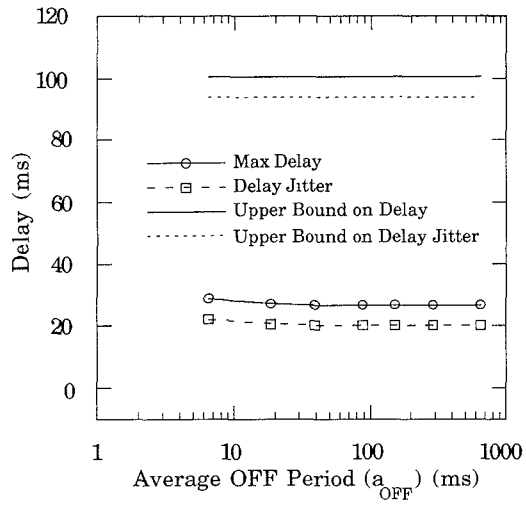

Figure 16: The maximum delay and delay jitter of a session *without* delay jitter control in class 2 in a MIX Traffic configuration of ON_OFF sessions.

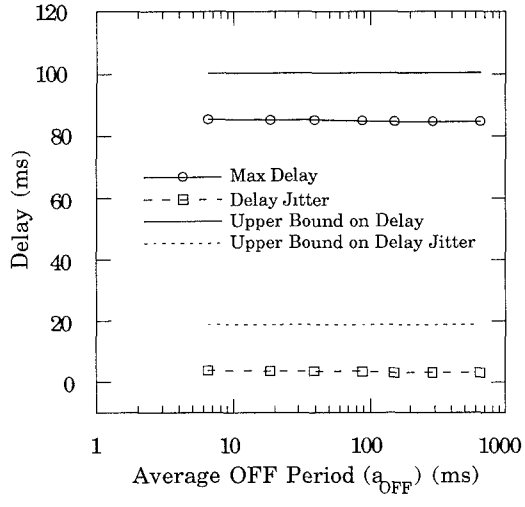

Figure 17: The maximum delay of a session *with* delay jitter control in class 2 in a MIX traffic configuration of ON_OFF sessions.

These experiments were also done using admission control procedure 1; we do not show the detailed results due to lack of space. The main result is that admission control procedure 2 allows class 1 sessions to have a lower upper bound on delay (and upper bound on delay jitter) than class 1 sessions with admission control procedure 1. But as explained earlier, this benefit comes with the drawback of having to assign a large enough value to σ_P such that all the bandwidth of the server can be exploited, effectively causing an increase in end-to-end delay for session in class P. This problem does not occur in admission control procedure 1.

4 Comparing Leave-in-Time to Other Schemes

Leave-in-Time is most related to VirtualClock [29] and Jitter-EDD [22]. In VirtualClock, each packet is assigned a transmission deadline and packets are served in increasing order of deadline. The transmission deadlines are calculated by an equation equivalent to equation (2). The Leave-in-Time service discipline builds on equation (2) through two generalizations. One generalization allows a session to reduce its end-to-end delay jitter, and the other allows a session to reduce its end-to-end delay. Many of the results we presented for the Leave-in-Time service discipline can be translated into results for the VirtualClock service discipline by considering a special case of Leave-in-Time's operation, i.e. admission control procedure 1 with one class and no jitter control. In this case, an upper bound on end-to-end delay for VirtualClock is obtainable from inequality (12) with $\alpha_s^N = 0$ and $d^n_{max,s} = L_{max,s}/r_s$ in $\beta_s^{1,N}$. In exactly the same manner, upper bounds on the end-to-end delay distribution, end-to-end delay jitter, and buffer space requirements are obtained. Except for the upper bound on end-to-end delay (presented in [7]), these results are new for the VirtualClock service discipline.

In Delay-EDD [5], and its extension Jitter-EDD [22] (EDD stands for earliest-due-date), packets are assigned deadlines and transmitted in order of increasing deadline. The deadline of a packet is not directly coupled to the reserved bandwidth of its session as in the Leave-in-Time scheme (see equation (11)). This leads to a schedulability test at connection establishment time [5] to avoid scheduling saturation, which can occur even if bandwidth is not overbooked [5, 28]. The schedulability test is then a compromise on the looser coupling between reserved rate and delay bound. The Leave-in-Time scheme needs an admission control procedure for the same reason.

In order to provide performance guarantees, the input traffic in Delay-EDD and Jitter-EDD (and RCSP, see below) must be constrained to a scheme more restrictive than a token-bucket filter. The traffic characterization specifies a minimum packet interarrival time $x_{\min}$, a minimum average packet interarrival time x_{ave} over an averaging interval of time I, and a maximum packet length P. In [26], bandwidth is reserved at the peak rate implied by $x_{\min}$. This admission control is refined in [27], where both $x_{\min}$ and x_{ave} are taken into consideration. In the Leave-in-Time scheme, bandwidth may be reserved at the average rate, although a session may need to reserve more bandwidth than its average rate in order to reduce the end-to-end delay, due to the coupling between delay bound and bandwidth allocation.

Stop-and-Go Queueing is proposed in [9,10,11] and uses a *framing strategy*. Stop-and-Go guarantees a minimum and maximum delay for packets of a session traversing the network. Stop-and-Go is a non-work-conserving service discipline, since arriving packets must wait until the beginning of the next frame to be served, even when the server is idle. The admission control of Stop-and-Go requires that during any time frame of size T, the arrived packets collectively have no more than r_sT bits, where r_s is the bandwidth given to session s. In [9], a session is called (r_s, T)-smooth if it follows this traffic constraint. This is more restrictive than a token bucket filter. The delay under Stop-and-Go is equal to $\alpha HT \pm T$, where α is a constant in the interval [1, 2), and H is the number of links traversed by a session. This creates a trade-off between queueing delays and flexibility in bandwidth allocation. Consider a session with all packets of fixed length L. For this session, the incremental step of bandwidth allocation is L/T, since the session must conform to the admission control. This means that one cannot have a low upper bound on delay and fine granularity of bandwidth allocation at the same time. Besides this, all sessions must be classified according to a finite number of connection types, since the set of possible frame sizes is generally predefined. Consequently, Stop-and-Go offers less flexibility in bandwidth allocation than the Leave-in-Time scheme.

The Leave-in-Time scheme also suffers from a coupling

between delay bound and bandwidth allocation, since one has to give more bandwidth to a session in order to reduce the value of $D^{ref}_{max,s}$ (see inequality (12)) and the value of $\beta^{1,N}_s$ when using admission control procedures 1 or 2. However, the coupling is not to the same extent as in Stop-and-Go. To see this, consider a session that generates at most 10 packets of length $0.01TC$ in any interval of T seconds, where C is the rate of the outgoing link of all servers. The average rate of this session is then $0.1C$, independent of the value of T. Now, suppose that both schemes allocate a bandwidth of $0.1C$ to this session. In Stop-and-Go, the delay will be $\alpha HT \pm T$. In the Leave-in-Time scheme, the delay will be at most $D^{ref}_{max,s} + \beta^{1,N}_s + \alpha^N_s$ (from equation (12)), where $D^{ref}_{max,s} = T$, since this session conforms to a token bucket filter $(r_s, b_{0,s}) = (0.1C, 0.1CT)$ (see inequality (14)). If the Leave-in-Time network uses admission control procedure 1 with one class and $d^n_{i,s} = L_{i,s}/r_s = 0.1T$, the upper bound on delay will be $T + \beta^{1,N}_s$, since $\alpha^N_s = 0$. The per-link increase[2] in the upper bound on delay is αT for Stop-and-Go (where $1 \le \alpha < 2$) and $L_{MAX}/C + 0.1T$ (inequality (12) with $d^n_{max,s} = L_{max,s}/r_s = 0.1T$) for the Leave-in-Time scheme. This shows that the delay bound in Stop-and-Go can be much larger than the delay bound in the Leave-in-Time scheme.

The delay jitter bound provided by the Leave-in-Time scheme is $J^{1,N}_s < D^{ref}_{max,s} + \delta^N_{max,s} - d^N_{max,s} + \alpha^N_s$ (inequality (17)), while in Stop-and-Go it is $2T$. A (r_s, T)-smooth session conforms to a token bucket filter (r_s, r_sT). Thus, from (14), $J^{1,N}_s < T + \delta^N_{max,s} - d^N_{max,s} + \alpha^N_s$, which is competitive with the result of Stop-and-Go.

The reader is referred to [25] which compares the delay distribution seen by sessions under FCFS multiplexing with the delay bound computed with Cruz's method [2, 3], the delay bound using Stop-and-Go, and the delay bound using PGPS (presented later in this section).

Hierarchical Round Robin (HRR) [13] also uses a framing strategy and is a non-work-conserving service discipline. It offers the same upper bound on delay as Stop-and-Go, but does not guarantee a lower bound on delay. The same arguments in the discussion of Stop-and-Go also apply.

Rate-Controlled Static-Priority Queueing (RCSP) [26] is a service discipline that avoids both framing strategies (as in Stop-and-Go and HRR) and sorted priority queues (that are used in all the other service disciplines studied here), by the separation of rate-control and delay-control in the design of the server, which allows it to provide throughput, delay, delay jitter, and loss free guarantees. Leave-in-Time uses an approximate sorted priority queue algorithm which runs in O(1) time with a small cost in emulation error [6].

Weighted Fair Queueing (WFQ) is proposed in [4] and is a service discipline that tries to emulate the service provided by a bit-by-bit round robin server. Each packet is stamped with the finishing round number (or *virtual time finishing time*, as called by Parekh in [17]) at which the packet would have finished transmission, had the server been doing a bit-by-bit round robin. Packets are served in increasing order of finishing round number. Packet-by-Packet Generalized Processor Sharing (PGPS) is the name given in [17, 18, 19, 20] to a method for computing delay bounds under WFQ when sessions conform to a token bucket filter. In Section 2, the upper bound on end-to-end delay for sessions conforming to a token bucket filter was presented, and it is the same as that found using PGPS for Leave-in-Time with admission control procedure 1 with one class.

The most significant difference between PGPS and the Leave-in-Time scheme is in the calculus of the transmission deadline of arrived packets. In the Leave-in-Time scheme, the transmission deadline of a packet is a function only of the past behavior of the session to which it belongs, while in PGPS, the finishing round number of a packet depends on the set of sessions with queued packets at the server at the time the packet arrives. The transmission deadline in the Leave-in-Time scheme is expressed in units of real time, while in PGPS, it is expressed in round units. This makes the Leave-in-Time scheme easier to analyze, and easier to implement. However, the Leave-in-Time service discipline does not conform to the notion of fairness attributed to PGPS. PGPS is called a *fair queueing scheme* because it closely emulates the service provided by a bit-by-bit round robin server. The reader is referred to [12] for a relevant work on fair queueing systems.

Others have addressed the possibility of providing per-session bounds on delay and delay distribution in a network setting [2, 3, 15, 24]. In [2, 3], Cruz uses a non-probabilistic approach to characterize each session entering the network -- the *burstiness constraint*, which is in principle very similar to a token bucket filter. Under this assumption, a methodology is proposed to calculate per-session upper bounds on delay and buffer requirements. Kurose in [15], and Yaron and Sidi in [24] describe methods to calculate bounds on the distribution of delay and buffer occupancy when all sessions entering the network are stochastically bounded. The work in [2, 3, 15, 24] differs from our work in that the Leave-in-Time scheme provides a function to calculate an upper bound on the delay distribution for a session, while the methodology in [2, 3, 15, 24] provides the upper bound on the delay distribution directly. However, the Leave-in-Time service discipline is able to provide this function for sessions *with any kind* of dynamic traffic behavior, and this function depends only on the session in question, i.e. the dynamic traffic behavior of other sessions does not enter into consideration (i.e. the Leave-in-Time service discipline provides isolation between sessions), while the methodology in [2, 3, 15, 24] is based on the traffic characterization of all sessions sharing network servers with the session in question, and the dynamic traffic behavior of all sessions enters in the calculation of the delay distribution.

5 Conclusions

We have presented the Leave-in-Time service discipline which provides sessions with upper bounds on delay, delay distribution, and delay jitter, all end-to-end, and an upper bound on server buffer space. The requirements are that a session declare its minimum required bandwidth which must be reserved for each link along the path carrying the session, and that the maximum size of the session's packets be bounded. Leave-in-Time's performance bounds depend only on the dynamic traffic behavior of that session, and are not affected by the behavior of other sessions being transported over the same links and servers. The bound on delay distribution is especially useful in supporting "tolerant" applications as defined in [1].

A special case of Leave-in-Time's operation reduces to that of VirtualClock. Since the performance bounds we have presented apply to the general case of Leave-in-Time's operation, they also apply to VirtualClock. In the case of bounds on end-to-end delay jitter, end-to-end delay distribution, and server buffer space, these bounds are new results for VirtualClock. Leave-in-Time extends

2. Propagation delay is not considered here since it increases the delay by the same amount for both disciplines

VirtualClock by operating in a non-work-conserving mode similar to Jitter-EDD that results in significantly reduced jitter. In effect, Leave-in-Time exploits the good properties of VirtualClock and Jitter-EDD while maintaining efficiency and flexibility, and providing desirable performance bounds.

Leave-in-Time goes beyond VirtualClock and Jitter-EDD by supporting delay shifting. Delay shifting provides flexibility over end-to-end delay bounds by allowing parts of the delays of some sessions, on a per-server basis, to be traded off with those of others. We presented a set of admission control procedures that provide an intuitive framework of priority classes such that delay shifting is done in a systematic way. These admission control procedures represent different degrees of flexibility versus computational complexity.

References

[1] 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] R. L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *In IEEE Transactions on Information Theory*, Vol. 37, No. 1, pp. 114-131, January 1991.

[3] R. L. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *In IEEE Transactions on Information Theory*, Vol. 37, No. 1, pp. 132-141, January 1991.

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *In Proceedings of ACM SIGCOMM '89*, pp. 1-12, September 1989.

[5] D. Ferrari and D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *In IEEE JSAC*, Vol. 8, No. 4, pp. 368-379, April 1990.

[6] N. R. Figueira, "A New Approach to the Control of Real-Time Traffic in Packet Switching Data Networks," Ph.D. Dissertation, Department of Computer Science and Engineering, University of California, San Diego, June 1995.

[7] N. R. Figueira and Joseph Pasquale, "An Upper Bound on Delay for the VirtualClock Service Discipline," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, August 1995, (in press).

[8] N. R. Figueira and Joseph Pasquale, "Leave-in-Time: A Service Discipline for Real-Time Communications in a Packet Switching Network," Technical Report CSE95-426, University of California, San Diego, May 1995.

[9] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management," *In Proceedings of ACM SIGCOMM '90*, pp. 8-18, September 1990.

[10] S. J. Golestani, "Congestion-Free Transmission of Real-Time Traffic in Packet Networks," *In Proceedings of IEEE INFOCOM '90*, pp. 527-536, June 1990.

[11] S. J. Golestani, "Duration-Limited Statistical Multiplexing of Delay-Sensitive Traffic in Packet Networks," I*n Proceedings of IEEE INFOCOM '91*, pp. 323-332, April 1991.

[12] S. J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," *In Proceedings of IEEE INFOCOM '94*, pp. 636-646, June 1994.

[13] C. Kalmanek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," *In Proceedings of IEEE GlobeCom '90*, pp. 300.3.1-300.3.9, December 1990.

[14] L. Kleinrock, *Queueing Systems, Vol. 1*. New York: Wiley, 1975.

[15] J. Kurose, "On Computing Per-session Performance Bounds in High-Speed Multi-hop Computer Networks," *In ACM Sigmetrics '92*, pp. 128-139, June 1992.

[16] A. M. Lee, *Applied Queueing Theory*. London: Macmillan, New York: St. Martin's Press, 1966.

[17] A. K. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," Ph.D. Dissertation, Massachusetts Institute of Technology, LIDS-TH-2089, February 1992.

[18] A. K. Parekh, and G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks -- The Single Node Case," *In Proceedings of IEEE INFOCOM '92*, Vol. 2, pp. 915-924, May 1992.

[19] A. K. Parekh, and G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks -- The Multiple Node Case," *In Proceedings of IEEE INFOCOM '93*, Vol. 2, pp. 521-530, March 1993.

[20] A. K. Parekh, and G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *In IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, pp. 344-357, June 1993.

[21] J. R. Shelton, "Solution Methods for Waiting Line Problems," *Journal of Industrial Engineering*, pp. 293-303, July-August 1960.

[22] D. Verma, H. Zhang, and D. Ferrari, "Delay Jitter Control for Real-Time Communication in a Packet Switching Network," *In Proceedings of IEEE TriCom '91*, pp. 35-43, April 1991.

[23] R. W. Wolff, *Stochastic Modelling and the Theory of Queues*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[24] O. Yaron and M. Sidi, "Calculating Performance Bounds in Communication Networks," *In Proceedings of IEEE INFOCOM '93*, Vol. 2, pp. 539-545, March 1993.

[25] D. Yates, J. Kurose, D. Towsley, and M. G. Hluchyj, "On Per-session End-to-End Delay Distributions and The Call Admission Problem for Real-Time Applications with QOS Requirements," *In Proceedings of ACM SIGCOMM '93*, pp. 2-12, September 1993.

[26] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing," *In Proceedings of IEEE INFOCOM '93*, pp. 227-236, March 1993.

[27] H. Zhang and D. Ferrari, "Improving Utilization for Deterministic Service in Multimedia Communication," *In Proceedings of the International Conference on Multimedia Computing and Systems*, pp. 295-304, May 1994.

[28] H. Zhang and S. Keshav, "Comparison of Rate-Based Service Disciplines," *In Proceedings of ACM SIGCOMM '91*, pp. 113-121, August 1991.

[29] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *In ACM Transactions on Computer Systems*, Vol. 9, No. 2, pp. 101-124, May 1991. Also in *Proceedings of ACM SIGCOMM '90*, pp. 19-29, September 1990.