A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case.[1]

Abhay K. Parekh [2] [3] and Robert G. Gallager [4]
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

Abstract

The problem of allocating network resources to the users of an integrated services network is investigated in the context of rate-based flow control. The network is assumed to be a virtual circuit, connection-based packet network. We show that the use of Generalized Processor Sharing (GPS), when combined with leaky bucket admission control, allows the network to make a wide range of worst-case performance guarantees on throughput and delay. The scheme is flexible in that different users may be given widely different performance guarantees, and is efficient in that each of the servers is work conserving. We present a practical packet-by-packet service discipline, PGPS (first proposed by Demers, Shenker and Keshav [7] under the name of Weighted Fair Queueing), that closely approximates GPS. This allows us to relate results for GPS to the packet-by-packet scheme in a precise manner.

In this paper, the performance of a single server GPS system is analyzed exactly from the standpoint of worst-case packet delay and burstiness when the sources are constrained by leaky buckets. The worst-case session backlogs are also determined. In the sequel to this paper, these results are extended to arbitrary topology networks with multiple nodes.

[1] An abridged version of the results in this paper was presented at IEEE INFOCOM '92.

[2] Research partly funded by a Vinton Hayes Fellowship and a Center for Intelligent Control Systems Fellowship.

[3] Current Address: IBM T. J. Watson Research Center, PO Box 704, Yorktown Heights NY 10598.

[4] Research Funded by the National Science Foundation under 8802991-NCR and by the Army Research Office under DAAL03-86-K-0171.

1 Introduction

This paper and its sequel [17] focus on a central problem in the control of congestion in high speed integrated services networks. Traditionally, the flexibility of data networks has been traded off with the performance guarantees given to its users. For example, the telephone network provides good performance guarantees but poor flexibility, while packet switched networks are more flexible, but only provide marginal performance guarantees. Integrated services networks must carry a wide range of traffic types and still be able to provide performance guarantees to real-time sessions such as voice and video. We will investigate an approach to reconcile these apparently conflicting demands when the short-term demand for link usage frequently exceeds the usable capacity.

We propose the combined use of a packet service discipline based on Generalized Processor Sharing and leaky bucket rate control to provide flexible, efficient and fair use of the links. Neither Generalized Processing Sharing, nor its packet based version, PGPS, are new. Generalized Processor Sharing is a natural generalization of uniform processor sharing [14], and the packet-based version (while developed independently by us) was first proposed in [7] under the name of Weighted Fair Queueing. Our contribution is to suggest the use of PGPS in the context of integrated services networks and to combine this mechanism with leaky bucket admission control in order to provide performance guarantees in a flexible environment.

A major part of our work is to analyze networks of arbitrary topology using these specialized servers, and to show how the analysis leads to implementable schemes for guaranteeing worst-case packet delay. In this paper, however, we will restrict our attention to sessions at a single node, and postpone the analysis of arbitrary topologies to the sequel.

Our approach can be described as a strategy for rate-based flow control. Under rate-based schemes, a source's traffic is parametrized by a set of statistics such as average rate, maximum rate, burstiness etc., and is assigned a vector of values corresponding to these parameters. The user also requests a certain quality of service, that might be characterized, for example, by tolerance to worst-case or average delay. The network checks to see if a new source can be accommodated, and if so, it takes actions (such as reserving transmission links or switching capacity) to ensure the quality of service desired. Once a source begins sending traffic, the network ensures that the agreed upon values of traffic parameters are not violated.

Our analysis will concentrate on providing guarantees on throughput and worst-case packet delay. While packet delay in the network can be expressed as the sum of the processing, queueing, transmission and propagation delays, we will focus exclusively on how to limit *queueing* delay.

We will assume that rate admission control is done through *leaky buckets* [20]. An important advantage of using leaky buckets is that this allows us to separate the packet delay into two components–delay in the leaky bucket and delay in the network. The first of these components is *independent* of the other active sessions, and can be estimated by the user if the statistical characterization of the incoming data is sufficiently simple (see Section 6.3 of [1] for an example). The traffic entering the network has been "shaped" by the leaky bucket in a manner that can be succinctly characterized (we will do this in Section 5), and so the network can upper bound the second component of packet delay through this characterization. This upper bound is independent of the statistics of the incoming data, which is helpful in the usual case where these statistics are either complex or unknown. A similar approach to the analysis of interconnection networks has been taken by Cruz [5]. From this point on, we will not consider the delay in the leaky bucket.

Generalized Processor Sharing (GPS) is defined and explained in Section 2. In Section 3 we present the packet-based scheme, PGPS, and show that it closely approximates GPS. Results obtained in this section allow us to translate session delay and buffer requirement bounds derived for a GPS server system to a PGPS server system. We propose a virtual time implementation of PGPS in the next subsection. Then PGPS is compared to weighted round robin, virtual clock multiplexing [21] and stop-and-go queueing [9, 10, 11].

Having established PGPS as a desirable multiplexing scheme we turn our attention to the rate enforcement function in Section 5. The leaky bucket is described and proposed as a desirable strategy for admission control. We then proceed with an analysis, in Sections 6 to 8, of a single GPS server system in which the sessions are constrained by leaky buckets. The results obtained here are crucial in the analysis of arbitrary topology, multiple node networks, which we will present in the sequel to this paper.

2 GPS Multiplexing

The choice of an appropriate service discipline at the nodes of the network is key to providing effective flow control. A good scheme should allow the network to treat users differently, in accordance with their desired quality of service. However, this *flexibility* should not compromise the *fairness* of the scheme, i.e. a few classes of users should not be able to degrade service to other classes, to the extent that performance guarantees are violated. Also, if one assumes that the demand for high bandwidth services is likely to keep pace with the increase in usable link bandwidth, time and frequency multiplexing are too wasteful of the network resources to be considered as candidate multiplexing disciplines. Finally, the service discipline must be *analyzable* so that performance guarantees can be made in the first place. We now present a flow-based multiplexing discipline called Generalized Processor Sharing that is efficient, flexible, and analyzable, and that therefore seems very appropriate for integrated services networks. However, it has the significant drawback of not transmitting packets as entities. In Section 3 we will present a packet-based multiplexing discipline that is an excellent approximation to GPS even when the packets are of variable length.

A Generalized Processor Sharing (GPS) server is work conserving and operates at a fixed rate r. It is characterized by positive real numbers $\phi_1, \phi_2, ..., \phi_N$. Let $S_i(\tau, t)$ be the amount of session i traffic served in an interval $[\tau, t]$. A session is backlogged at time t if a positive amount of that session's traffic is queued at time t. Then a GPS server is defined as one for which

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}, \quad j = 1, 2, ..., N \tag{1}$$

for any session i that is continuously backlogged in the interval $[\tau, t]$.

Summing over all sessions j:

$$S_i(\tau,t) \sum_j \phi_j \geq (t-\tau) r \phi_i$$

and session i is guaranteed a rate of

$$g_i = \frac{\phi_i}{\sum_j \phi_j} r. \tag{2}$$

GPS is an attractive multiplexing scheme for a number of reasons:

- Define r_i to be the session i average rate. Then as long as $r_i \leq g_i$, the session can be guaranteed a throughput of ρ_i, independent of the demands of the other sessions. In addition to this throughput guarantee, a session i backlog will always be cleared at a rate $\geq g_i$.

- The delay of an arriving session i bit can be bounded as a function of the session i queue length, independent of the queues and arrivals of the other sessions. Schemes such as FCFS, LCFS, and Strict Priority do not have this property.

- By varying the ϕ_i's we have the flexibility of treating the sessions in a variety of different ways. For example, when all the ϕ_i's are equal, the system reduces to uniform processor sharing. As long as the combined average rate of the sessions is less than r, any assignment of positive ϕ_i's yields a stable system. For example, a high bandwidth delay-insensitive session, i, can be assigned g_i much less than its average rate, thus allowing for better treatment of the other sessions.

- Most importantly, it is possible to make worst-case network queueing delay *guarantees* when the sources are constrained by leaky buckets. We will present our results on this later. Thus GPS is particularly attractive for sessions sending real-time traffic such as voice and video.

Figure 1 illustrates generalized processor sharing. Variable length packets arrive from both sessions on infinite capacity links and appear as impulses to the system. For $i = 1, 2$, let $A_i(0,t)$ be the amount of session i traffic that arrives at the system in the interval $(0,t]$, and similarly, let $S_i(0,t)$ be the amount of session i traffic that is served in the interval $(0,t]$. We assume that the server works at rate 1.

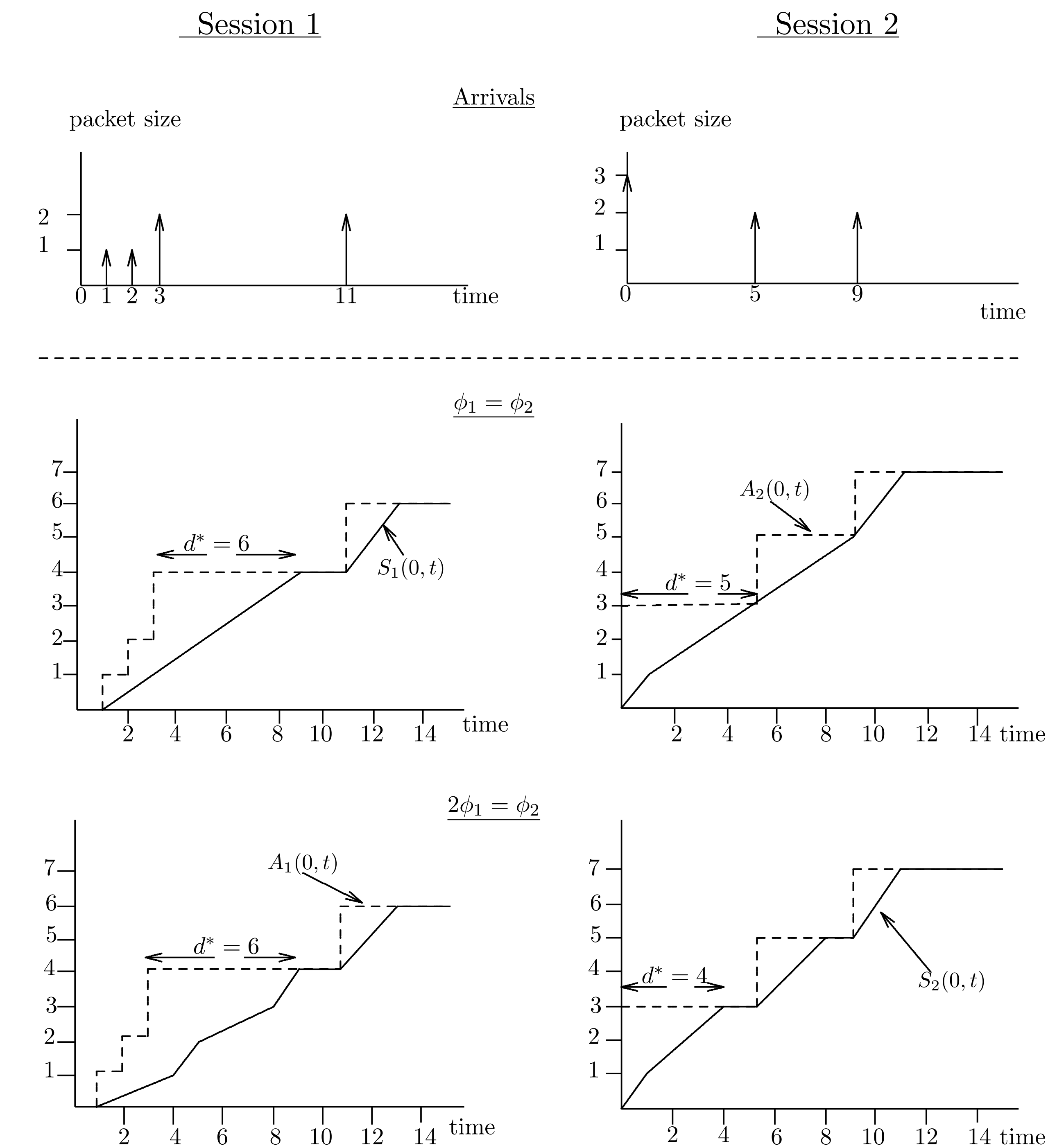
When $\phi_1 = \phi_2$, and both sessions are backlogged, they are each served at rate $\frac{1}{2}$ (eg. the interval $[1,6]$). When $2\phi_1 = \phi_2$, and both sessions are backlogged, session 1 is served at rate $\frac{1}{3}$ and session 2 at rate $\frac{2}{3}$. Notice how increasing the relative weight of ϕ_2 leads to better treatment of that session in terms of both backlog and delay. The delay to session 2 goes down by one time unit, and the delay to session 1 goes up by one time unit. Also, notice that under both choices of ϕ_i, the system is empty at time 13 since the server is work conserving under GPS.

It should be clear from the example that the delays experienced by a session's packets can be reduced by increasing the value of ϕ for that session. But this reduction may be at the expense of *a corresponding* increase in delay for packets from the other sessions. Figure 2 demonstrates that this may not be the case when the better treated session is steady. Thus, when combined with appropriate rate enforcement, the flexibility of GPS multiplexing can be used effectively to control packet delay.

3 A Packet-by-Packet Transmission Scheme–PGPS

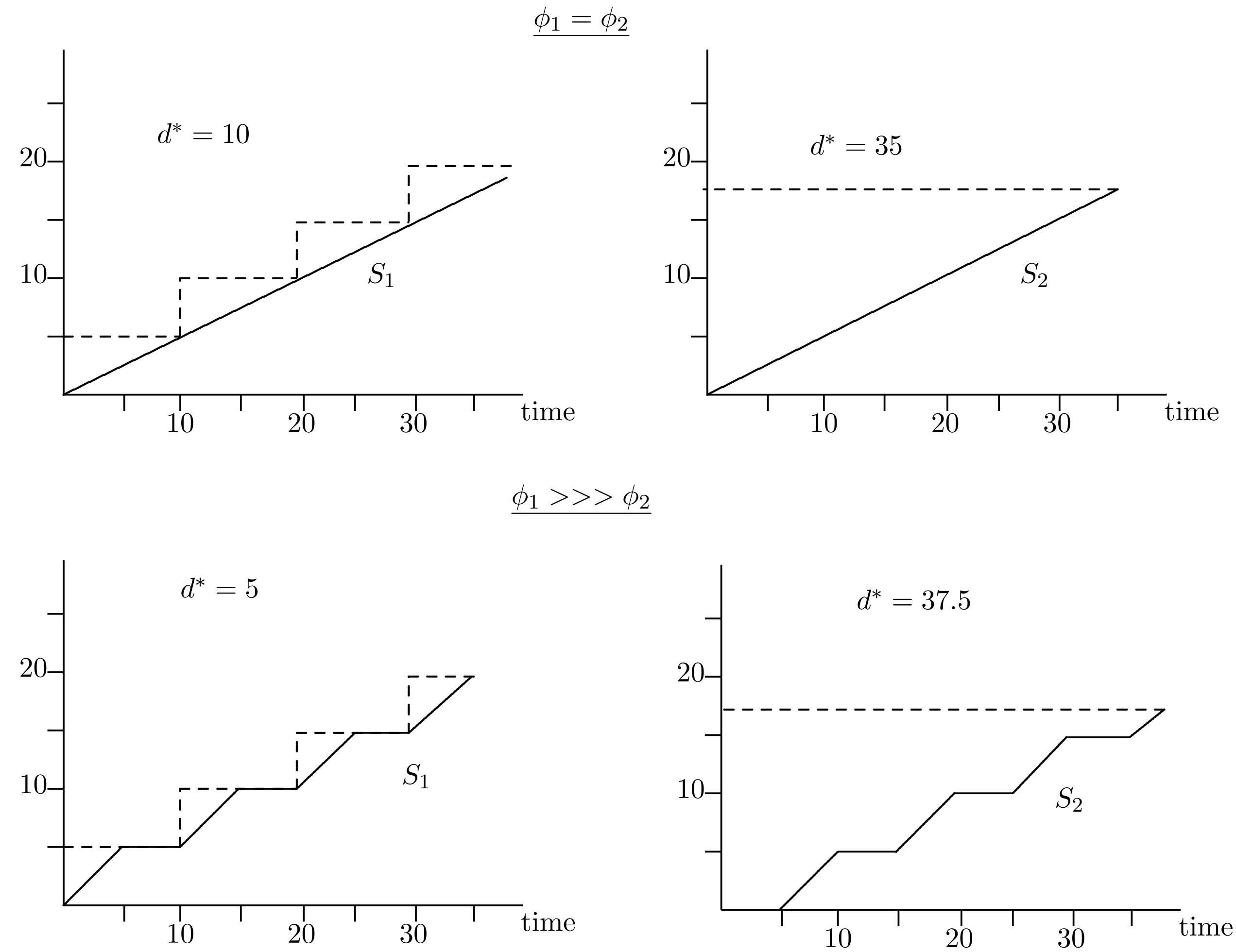
A problem with GPS is that it is an idealized discipline that does not transmit packets as entities. It assumes that the server can serve multiple sessions simultaneously and that the traffic is infinitely divisible. In this section we present a simple packet-by-packet transmission scheme that is an excellent approximation to GPS even when the packets are of variable length. Our idea is identical to the one used in [7]. *We will adopt the convention that a packet has arrived only after its last bit has arrived.*

Let F_p be the time at which packet p will depart (finish service) under generalized processor sharing. Then a very good approximation of GPS would be a work conserving scheme that serves packets in increasing order of F_p. (By work conserving we mean that



The packets arrive on links with infinite speed and are of variable length. Notice that by increasing $\frac{\phi_2}{\phi_1}$, we can give session 2 better service.

Figure 1: An example of generalized processor sharing.



The dotted curves represent cumulative arrivals. Session 1 is a steady session that is also delay sensitive (perhaps it is a video session). Its worst-case packet delay can be cut in half with minimal performance degradation to other sessions. In the figure ϕ_1 is increased to infinity, but session 2 delay goes up by only 2.5 time units.

Figure 2: The effect of increasing ϕ_i for a steady session i

		Session 1				Session 2		
packet information	Arrival Size	1 1	2 1	3 2	11 2	0 3	5 2	9 2
$\phi_1 = \phi_2$	GPS	3	5	9	13	5	9	11
	PGPS	4	5	7	13	3	9	11
$2\phi_1 = \phi_2$	GPS	4	5	9	13	4	8	11
	PGPS	4	5	9	13	3	7	11

The lower portion of the table gives the packet departure times under both schemes.

Figure 3: How GPS and PGPS compare for the example in Figure 1.

the server is always busy when there are backlogged packets in the system.) Now suppose that the server becomes free at time τ. The next packet to depart under GPS *may not have arrived* at time τ, and since the server has no knowledge of when this packet will arrive, there is no way for the server to be *both* work conserving and to serve the packets in increasing order of F_p. The server picks the first packet that would complete service in the GPS simulation if no additional packets were to arrive after time τ. Let us call this scheme PGPS for *packet-by-packet* generalized processor sharing. As stated earlier, this mechanism was originally called Weighted Fair Queueing [7].

Figure 3 shows how PGPS performs for the example in Figure 1. Notice that when $\phi_1 = \phi_2$, the first packet to complete service under GPS is the session 1 packet that arrives at time 1. However, the PGPS server is forced to begin serving the long session 2 packet at time 0, since there are no other packets in the system at that time. Thus the session 1 packet arriving at time 1 departs the system at time 4, i.e. 1 time unit later than it would depart under GPS.

A natural issue to examine at this point is how much later packets may depart the system under PGPS relative to GPS. First we present a useful property of GPS systems.

Lemma 1 *Let p and p' be packets in a GPS system at time τ and suppose that packet p completes service before packet p' if there are no arrivals after time τ. Then packet p will also complete service before packet p' for any pattern of arrivals after time τ.*

Proof. The sessions to which packets p and p' belong are both backlogged from time τ until one completes transmission. By (1), the ratio of the service received by these sessions is independent of future arrivals. □

A consequence of this Lemma is that if PGPS schedules a packet p at time τ before another packet p' that is also backlogged at time τ, then in the simulated GPS system, packet p

cannot leave later than packet p'. Thus, the only packets that are delayed more in PGPS, are those that arrive too late to be transmitted in their GPS order. Intuitively, this means that only the packets that have small delay under GPS are delayed more under PGPS.

Now let $\hat{F}_p$ be the time at which packet p departs under PGPS. We show that

Theorem 1 *For all packets p,*

$$\hat{F}_p - F_p \leq \frac{L_{\max}}{r},$$

where $L_{\max}$ is the maximum packet length, and r is the rate of the server.

Proof. Since both GPS and PGPS are work conserving disciplines, their busy periods coincide i.e. the GPS server is in a busy period iff the PGPS server is in a busy period. Hence it suffices to prove the result for each busy period. Consider any busy period and let the time that it begins be time zero. Let p_k be the k^{th} packet in the busy period to depart under PGPS and let its length be L_k. Also let t_k be the time that p_k departs under PGPS and u_k be the time that p_k departs under GPS. Finally, let a_k be the time that p_k arrives. We now show that

$$t_k \leq u_k + \frac{L_{\max}}{r}$$

for $k = 1, 2, ...$ Let m be the largest integer that satisfies both $0 < m \leq k-1$ and $u_m > u_k$. Thus

$$u_m > u_k \geq u_i \quad \text{for } m < i < k. \tag{3}$$

Then packet p_m is transmitted before packets $p_{m+1} \ldots, p_k$ under PGPS, but after all these packets under GPS. If no such integer m exists then set $m = 0$. Now for the case $m > 0$, packet p_m begins transmission at $t_m - \frac{L_m}{r}$, so from from Lemma 1:

$$\min\{a_{m+1}, ..., a_k\} > t_m - \frac{L_m}{r} \tag{4}$$

Since $p_{m+1}, ..., p_{k-1}$ arrive after $t_m - \frac{L_m}{r}$ and depart before p_k does under GPS:

$$u_k \geq \frac{1}{r}(L_k + L_{k-1} + L_{k-2} + \ldots + L_{m+1}) + t_m - \frac{L_m}{r}$$

$$\Rightarrow u_k \geq t_k - \frac{L_m}{r}.$$

If $m = 0$, then $p_{k-1}, ..., p_1$ all leave the GPS server before p_k does, and so

$$u_k \geq t_k.$$

$\square$

Note that that if N maximum size packets leave simultaneously in the reference system they can be served in arbitrary order in the packet-based system. Thus $F_p - \hat{F}_p \geq (N-1)\frac{L_{\max}}{r}$ even if the reference system is tracked perfectly.

Let $S_i(\tau, t)$ and $\hat{S}_i(\tau, t)$ be the amount of session i traffic (in bits, not packets) served under GPS and PGPS in the interval $[\tau, t]$.

Theorem 2 *For all times τ and sessions i.*

$$S_i(0, \tau) - \hat{S}_i(0, \tau) \leq L_{\max}.$$

Proof. The slope of $\hat{S}_i$ alternates between r when a session i packet is being transmitted, and 0 when session i is not being served. Since the slope of S_i also obeys these limits, the difference $S_i(0,t) - \hat{S}_i(0,t)$ reaches its maximal value when session i packets begin transmission under PGPS. Let t be some such time, and let L be the length of the packet going into service. Then the packet completes transmission at time $t + \frac{L}{r}$. Let τ be the time at which the given packet completes transmission under GPS. Then since session i packets are served in the same order under both schemes:

$$S_i(0, \tau) = \hat{S}_i(0, t + \frac{L}{r}).$$

From Theorem 1:

$$\begin{aligned} \tau &\geq (t + \frac{L}{r}) - \frac{L_{\max}}{r} && (5) \\ \Rightarrow S_i(0, t + \frac{L - L_{\max}}{r}) &\leq \hat{S}_i(0, t + \frac{L}{r}) && (6) \\ &= \hat{S}_i(0, t) + L. && (7) \end{aligned}$$

Since the slope of S_i is at most r, the Theorem follows. $\Box$

Let $\hat{Q}_i(\tau)$ and $Q_i(t)$ be the session i backlog (in units of traffic) at time τ under PGPS and GPS respectively. Then it immediately follows from Theorem 2 that

Corollary 1 *For all times τ and sessions i.*

$$\hat{Q}_i(0, \tau) - Q_i(0, \tau) \leq L_{\max}.$$

Theorem 1 generalizes result shown for the uniform processing case by Greenberg and Madras [12]. Notice that

- Theorem 1 and Corollary 1 can be used to translate bounds on GPS worst-case packet delay and backlog to corresponding bounds on PGPS.

- Variable packet lengths are easily handled by PGPS. This is not true of weighted round robin.

- The results derived so far can be applied to present an alternative solution to a problem studied in [4, 19, 2, 8, 3]: There are N input links to a multiplexer; the peak rate of the i^{th} link is C_i, and the rate of the multiplexer is $C \geq \sum_{i=1}^{N} C_i$. Since up to $L_{\max}$ bits from a packet may be queued from any link before the packet has "arrived'," at least $L_{\max}$ bits of buffer must be allocated to each link. In fact, in [3] it is shown that at least $2L_{\max}$ bits are required, and that a class of buffer policies called Least Time to Reach Bound (LTRB) meets this bound. It is easy to design a PGPS policy that meets this bound as well: Setting $\phi_i = C_i$, it is clear that the resulting GPS server ensures that no more than than $L_{\max}$ bits are ever queued at any link. The bound of Corollary 1 guarantees that no more than $2L_{\max}$ bits need to be allocated per link under PGPS. In fact, if L_i is the maximum allowable packet size for link i, then the bound on the link i buffer requirement is $L_i + L_{\max}$. Further, various generalizations of the problem can be solved: For example, suppose the link speeds are arbitrary, but no more than $f_i(t) + r_i t$ bits can arrive on link i in any interval of length t (for each i). Then if $\sum_i r_i \leq C$, setting $\phi_i = r_i$ for each i yields a PGPS service discipline for which the buffer requirement is $L_{\max} + \max_{t \geq 0}(f_i(t) - r_i t)$ bits for each link i.

- There is no constant $c \geq 0$ such that

$$\hat{S}_i(0,t) - S_i(0,t) \leq cL_{\max} \tag{8}$$

holds for all sessions i, over all patterns of arrivals. To see this, let $K = \lfloor c + 2 \rfloor$, $\phi_1 = K$, $\phi_2 = ... = \phi_N = 1$ and fix all packets sizes at $L_{\max}$. At time zero, $K - 1$ session 1 packets arrive and one packet arrives from each of the other sessions. No more packets arrive after time zero. Denote the $K - 1^{st}$ session 1 packet to depart GPS (and PGPS) as packet p. Then $F_p = \frac{K-1}{K}(N + K - 1)\frac{L_{\max}}{r}$, and $S_i(0, F_p) = \frac{K-1}{K} L_{\max}$ for $i = 2, ..., N$. Thus the first $K - 1$ packets to depart the GPS system are the session 1 packets, and packet p leaves PGPS at time $(K-1)\frac{L_{\max}}{r}$. Consequently,

$$\hat{S}_1(0, (K-1)\frac{L_{\max}}{r}) = (K-1)L_{\max}$$

and

$$S_1(0, (K-1)\frac{L_{\max}}{r}) = \frac{K(K-1)L_{\max}}{N - K + 1}.$$

This yields

$$\hat{S}_1(0,(K-1)\frac{L_{\max}}{r}) - S_1(0,(K-1)\frac{L_{\max}}{r}) = (K-1)L_{\max}(1-\frac{K}{N-K+1}). \quad (9)$$

For any given K, the RHS of (9) can be made to approach $(K-1)L_{\max}$ arbitrarily closely by increasing N.

3.1 A Virtual Time Implementation of PGPS

In this section we will use the concept of Virtual Time to track the progress of GPS that will lead to a practical implementation of PGPS. Our interpretation of virtual time generalizes the innovative one considered in [7] for uniform processor sharing. In the following we assume that the server works at rate 1.

Denote as an <u>event</u> each arrival and departure from the GPS server, and let t_j be the time at which the j^{th} event occurs (simultaneous events are ordered arbitrarily). Let the time of the first arrival of a busy period be denoted as $t_1 = 0$. Now observe that for each $j = 2, 3, ...$, the set of sessions that are busy in the interval (t_{j-1}, t_j) is fixed, and we may denote this set as B_j. Virtual time $V(t)$ is defined to be zero for all times when the server is idle. Consider any busy period, and let the time that it begins be time zero. Then $V(t)$ evolves as follows:

$$\begin{aligned} V(0) &= 0 \\ V(t_{j-1}+\tau) &= V(t_{j-1}) + \frac{\tau}{\sum_{i\in B_j}\phi_i}, \quad \tau \le t_j - t_{j-1}, \quad j = 2, 3, ... \end{aligned} \quad (10)$$

The rate of change of V, namely $\frac{\partial V(t_j+\tau)}{\partial \tau}$, is $\frac{1}{\sum_{i\in B_j}\phi_i}$, and each backlogged session i receives service at rate $\phi_i \frac{\partial V(t_j+\tau)}{\partial \tau}$. Thus, V can be interpreted as increasing at the marginal rate at which backlogged sessions receive service.

Now suppose that the k^{th} session i packet arrives at time a_i^k and has length L_i^k. Then denote the virtual times at which this packet begins and completes service as S_i^k and F_i^k respectively. Defining $F_i^0 = 0$ for all i, we have

$$\begin{aligned} S_i^k &= \max\{F_i^{k-1}, V(a_i^k)\} \\ F_i^k &= S_i^k + \frac{L_i^k}{\phi_i}. \end{aligned} \quad (11)$$

There are three attractive properties of the virtual time interpretation from the standpoint of implementation. First, the virtual time finishing times can be determined at the packet arrival time. Second, the packets are served in order of virtual time finishing time. Finally, we need only update virtual time when there are events in the GPS system. However, the price to paid for these advantages is some overhead in keeping track of the sets B_j, which is essential in the updating of virtual time:

Define $\text{Next}(t)$ to be the *real* time at which the next packet will depart the GPS system after time t if there are no more arrivals after time t. Thus the next virtual time update after t will be performed at $Next(t)$ if there are no arrivals in the interval $[t, \text{Next}(t)]$. Now suppose a packet arrives at some time, t (let it be the j^{th} event), and that the time of the event just prior to t is τ (if there is no prior event, i.e. if the packet is the first arrival in a busy period, then set $\tau = 0$). Then, since the set of busy sessions is fixed between events, $V(t)$ may be computed from (10), and the packet stamped with its virtual time finishing time. $\text{Next}(t)$ is the real time corresponding to the smallest virtual time packet finishing time at time t. This real time may be computed from (10) since the set of busy sessions, B_j, remains fixed over the interval $[t, Next(t)]$: Let $F_{\min}$ be the smallest virtual time finishing time of a packet in the system at time t. Then from (10):

$$F_{\min} = V(t) + \frac{\text{Next}(t) - t}{\sum_{i \in B_j} \phi_i}$$

$$\Rightarrow \text{Next}(t) = t + (F_{\min} - V(t)) \sum_{i \in B_j} \phi_i.$$

Given this mechanism for updating virtual time, PGPS is defined as follows: When a packet arrives, virtual time is updated and the packet is stamped with its virtual time finishing time. The server is work conserving and serves packets in increasing order of time-stamp.

4 Comparing PGPS to other schemes

Under weighted round robin, every session i, has an integer weight, w_i associated with it. The server polls the sessions according a *precomputed sequence* in an attempt to serve session i at a rate of $\frac{w_i}{\sum_j w_j}$. If an empty buffer is encountered, the server moves to the next session in the order instantaneously. When an arriving session i packet just misses its slot in a frame it cannot be transmitted before the next session i slot. If the system is heavily loaded in the sense that almost every slot is utilized, the packet may have to wait almost N slot times to be served, where N is the number of sessions sharing the server. Since PGPS approximates GPS to within one packet transmission time regardless of the arrival patterns, it is immune to such effects. PGPS also handles variable length packets in a much more systematic fashion than does weighted round robin. However, if N or the packets sizes are small then it is possible to approximate GPS well by weighted round robin. Hahne [13] has analyzed round robin in the context of providing fair rates to users of networks that utilize hop-by-hop window flow control.

Zhang proposes an interesting scheme called virtual clock multiplexing [21]. Virtual clock multiplexing allows guaranteed rate and (average) delay for each session, independent of the behavior of other sessions. However, if a session produces a large burst of data, even while the system is lightly loaded, that session can be "punished" much later when the other sessions become active. Under PGPS the delay of a session i packet can be bounded in terms of the session i queue size seen by that packet upon arrival, even in the absence of any rate control. This enables sessions to take advantage of lightly loaded network conditions. We illustrate this difference with a numerical example:

Suppose there are two sessions that submit fixed size packets of 1 unit each. The rate of the server is one, and the packet arrival rate is $\frac{1}{2}$ for each session. Starting at time zero, 1000 session 1 packets begin to arrive at a rate of 1 packet/second. No session 2 packets arrive in the interval $[0, 900)$, but at time 900, 450 session 2 packets begin to arrive at a rate of 1 packet/second. Now if the sessions are to be treated equally, the virtual clock for each session will tick at a rate of $\frac{1}{2}$, and the PGPS weight assignment will be $\phi_1 = \phi_2$. Since both disciplines are work-conserving, they will serve session 1 continuously in the interval $[0, 900)$.

At time 900^- there are no packets in queue from either session; the session 1 virtual clock will read 1800 and the session 2 virtual clock will read 900. The 450 session 2 packets that begin arriving at this time will be stamped $900, 902, 904, ..., 1798$, while the 100 session 1 packets that arrive after time 900 will be stamped $1800, 1804, ..., 1998$. Thus, **all** of the session 2 packets will be served under Virtual Clock before any of these session 1 packets are served. The session 1 packets are being punished since the session used the server exclusively in the interval $[0, 900)$. But note this exclusive use of the server was *not at the expense of any session* 2 *packets.* Under PGPS the sessions are served in round robin fashion from time 900 on, which results in much less delay to the session 1 packets.

The lack of a punishment feature is an attractive aspect of PGPS since in our scheme the admission of packets is regulated at the network periphery through leaky buckets, and it does not seem necessary to punish users at the internal nodes as well. Note however, that in this example PGPS guarantees a throughput of $\frac{1}{2}$ to each session even in the absence of access control.

Stop-and-Go Queueing is proposed in [9, 10, 11], and is based on a network-wide time slot structure. It has two advantages over our approach: it provides better jitter control and is probably easier to implement. A finite number of connection types are defined, where a type g connection is characterized by a fixed frame size of T_g. Since each connection must conform to a predefined connection type, the scheme is somewhat less flexible than PGPS. The admission policy under which delay and buffer size guarantees can be made is that no more than $r_i T_g$ bits may be submitted during any type g frame. If sessions $1, 2, ..., N$ are served by a server of capacity 1 it is stipulated that $\sum_{i=1}^{N} r_i \leq 1$, where the sum is only taken over the real-time sessions. The delay guarantees grow linearly with T_g, so in order to provide low delay one has to use a small slot-size. The service discipline is not work conserving and is such that each packet may be delayed up to $2T_g$ time units even when there is only one active session at the server. Observe that for a single session PGPS system in which the peak rate does not exceed the rate of the server, each arriving packet is served *immediately* upon arrival. Also, since it is work-conserving, PGPS will provide better average delay than stop-and-go for a given access control scheme.

It is clear that r_i is the average rate at which the source i can send data over a single slot. The relationship between delay and slot size may force Stop-and-Go to allocate bandwidth by peak to satisfy delay senstive sessions. This may also happen under PGPS but not to the same degree. To see this, consider an on-off periodic source that fluctuates between the values $C - \epsilon$ and 0. (As usual, ϵ is small.) The on period is equal to the off period, say they are B seconds in duration. We assume that B is large. Clearly the average rate of this session is $.5(C - \epsilon)$. We are interested in providing this session low delay under Stop-and-Go and PGPS. To do this, one has to pick a slot size is smaller than B, which forces $r = C - \epsilon$. The remaining capacity of the server that can be allocated is ϵ. Under PGPS we allocate a large value of ϕ to the session to bring its delay down to the desired level, however, now the remaining capacity that can be allocated is $.5(C + \epsilon)$.

Now observe that if there is a *second* on-off session with identical on and off periods as the first sesision, but which is relatively less delay sensitive, then PGPS can carry both sessions (since the combined sustainable rate is less than C) whereas Stop-and-Go cannot.

5 Leaky Bucket

Figure 4 depicts the leaky bucket scheme [20] that we will use to describe the traffic that enters the network. Tokens or permits are generated at a fixed rate, ρ, and packets can be released into the network only after removing the required number of tokens from the token bucket. There is no bound on the number of packets that can be buffered, but the *token* bucket contains at most σ bits worth of tokens. In addition to securing the required number of tokens, the traffic is further constrained to leave the bucket at a maximum rate

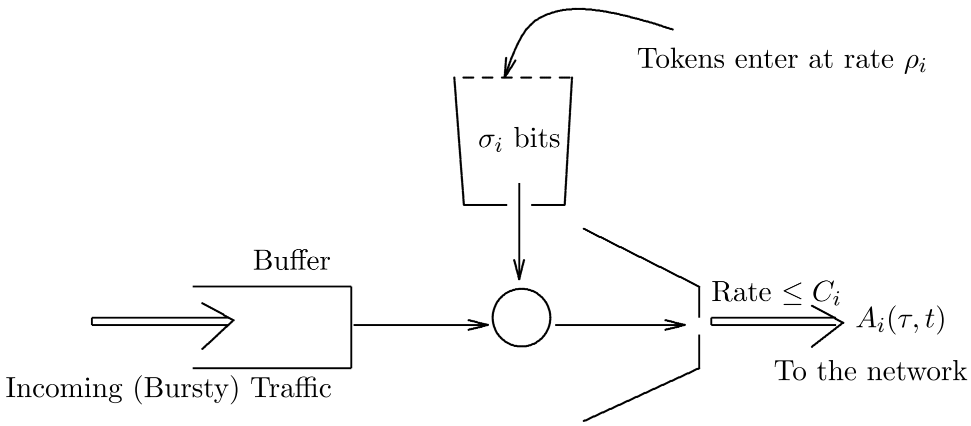


Figure 4: A Leaky Bucket

of $C > \rho$.

The constraint imposed by the leaky bucket is as follows: If $A_i(\tau, t)$ is the amount of session i flow that leaves the leaky bucket and enters the network in the time interval $(\tau, t]$, then

$$A_i(\tau, t) \leq \min\{(t-\tau)C_i, \sigma_i + \rho_i(t-\tau)\}, \quad \forall t \geq \tau \geq 0, \tag{12}$$

for every session i. We say that session i conforms to (σ_i, ρ_i, C_i), or $A_i \sim (\sigma_i, \rho_i, C_i)$.

This model for incoming traffic is essentially identical to the one recently proposed by Cruz [5], [6], and it has also been used in various forms to represent the inflow of parts into manufacturing systems by Kumar [18], [15]. The arrival constraint is attractive since it restricts the traffic in terms of average sustainable rate (ρ), peak rate (C), and burstiness (σ and C). Figure 5 shows how a fairly bursty source might be characterized using the constraints.

Represent $A_i(0, t)$ as in Figure 5. Let there be $l_i(t)$ bits worth of tokens in the session i token bucket at time t. We assume that the session starts out with a full bucket of tokens. If $K_i(t)$ is the total number of tokens accepted at the session i bucket in the interval $(0, t]$ (it does not include the full bucket of tokens that session i starts out with, and does not include arriving tokens that find the bucket full), then

$$K_i(t) = \min_{0 \leq \tau \leq t}\{A_i(0, \tau) + \rho_i(t-\tau)\}. \tag{13}$$

Thus for all $\tau \leq t$:

$$K_i(t) - K_i(\tau) \leq \rho_i(t-\tau). \tag{14}$$

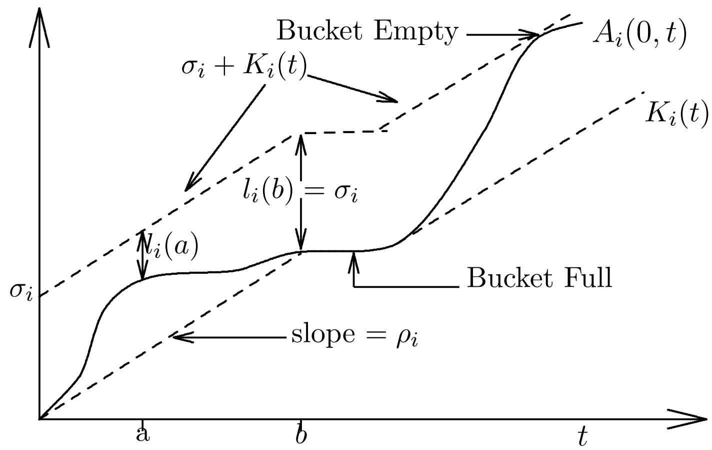


Figure 5: $A_i(t)$ and $l_i(t)$.

We may now express $l_i(t)$ as

$$l_i(t) = \sigma_i + K_i(t) - A_i(0,t). \tag{15}$$

From (15) and (14): we obtain the useful inequality:

$$A_i(\tau,t) \le l_i(\tau) + \rho_i(t-\tau) - l_i(t). \tag{16}$$

6 Analysis

In this section we analyze the worst-case performance of single node GPS systems for sessions that operate under leaky bucket constraints, i.e., the session traffic is constrained as in (12).

There are N sessions, and the only assumptions we make about the incoming traffic are that $A_i \sim (\sigma_i, \rho_i, C_i)$ for $i = 1, 2, ..., N$, and that the system is empty before time zero. The server is work conserving (i.e. it is never idle if there is work in the system), and operates at the fixed rate of 1. Let $S_i(\tau,t)$ be the amount of session i traffic served in the interval $(\tau,t]$. Note that $S_i(0,t)$ is continuous and non-decreasing for all t (see Figure 6). The session i backlog at time τ is defined to be

$$Q_i(\tau) = A_i(0,\tau) - S_i(0,\tau).$$

The session i delay at time τ is denoted by $D_i(\tau)$, and is the amount of time that it

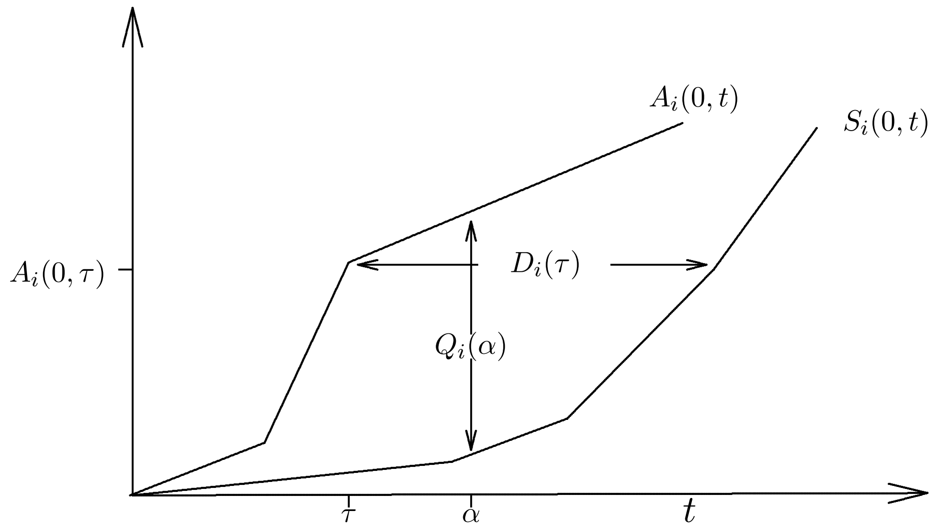


Figure 6: $A_i(0,t)$, $S_i(0,t)$, $Q_i(t)$ and $D_i(t)$

would take for the session i backlog to clear if no session i bits were to arrive after time τ. Thus

$$D_i(\tau) = \inf\{t \geq \tau \ : \ S_i(0,t) = A_i(0,\tau)\} \ - \ \tau. \tag{17}$$

From Figure 6 we see that $D_i(\tau)$ is the horizontal distance between the curves $A_i(0,t)$ and $S_i(0,t)$ at the ordinate value of $A_i(0,\tau)$.

Clearly, $D_i(\tau)$ depends on the arrival functions $A_1, ..., A_N$. We are interested in computing the maximum delay over all time, and over all arrival functions that are consistent with (12). Let D_i^* be the maximum delay for session i. Then

$$D_i^* = \max_{(A_1,\dots,A_N)} \max_{\tau \geq 0} D_i(\tau).$$

Similarly, we define the maximum backlog for session i, Q_i^*:

$$Q_i^* = \max_{(A_1,\dots,A_N)} \max_{\tau \geq 0} Q_i(\tau).$$

The problem we will solve in the following sections is: Given $\phi_1, ..., \phi_N$ for a GPS server of rate 1 and given (σ_j, ρ_j, C_j), $j = 1, ..., N$, what are D_i^* and Q_i^* for every session i? We will also be able to characterize the burstiness of the output traffic for every session i, which will be especially useful in our analysis of GPS networks in the sequel.

6.1 Definitions and Preliminary Results

We introduce definitions and derive inequalities that are helpful in our analysis. Some of these notions are general enough to be used in the analysis of any work-conserving service discipline (that operates on sources that are leaky bucket constrained).

Given $A_1, ..., A_N$, let σ_i^τ be defined for each session i and time $\tau \geq 0$ as

$$\sigma_i^\tau = Q_i(\tau) \ + l_i(\tau) \tag{18}$$

where $l_i(\tau)$ is defined in (15). Thus σ_i^τ is the sum of the number of tokens left in the bucket and the session backlog at the server at time τ. If $C_i = \infty$ we can think of σ_i^τ as the maximum amount of session i backlog at time τ^+, over all arrival functions that are identical to $A_1, ..., A_N$ up to time τ.

Observe that $\sigma_i^0 = \sigma_i$ and that

$$Q_i(\tau) = 0 \Rightarrow \sigma_i^\tau \leq \sigma_i. \tag{19}$$

Recall (16):

$$A_i(\tau, t) \leq l_i(\tau) + \rho_i(t - \tau) - l_i(t).$$

Substituting for l_i^τ and l_i^t from (18):

$$Q_i(\tau) + A_i(\tau, t) - Q_i(t) \leq \sigma_i^\tau - \sigma_i^t + \rho_i(t - \tau). \tag{20}$$

Now notice that

$$S_i(\tau, t) = Q_i(\tau) + A_i(\tau, t) - Q_i(t). \tag{21}$$

Combining (20) and (21) we establish the following useful result:

Lemma 2 *For every session i, $\tau \leq t$:*

$$S_i(\tau, t) \leq \sigma_i^\tau - \sigma_i^t + \rho_i(t - \tau). \tag{22}$$

Define a **system busy period** to be a maximal interval B such that for any $\tau, t \in B$, $\tau \leq t$:

$$\sum_{i=1}^{N} S_i(\tau, t) = t - \tau.$$

Since the system is work conserving, if $B = [t_1, t_2]$, then $\sum_{i=1}^N Q_i(t_1) = \sum_{i=1}^N Q_i(t_2) = 0$.

Lemma 3 *When $\sum_j \rho_j < 1$, the length of a system busy period is at most*

$$\frac{\sum_{i=1}^N \sigma_i}{1 - \sum_{i=1}^N \rho_i}.$$

Proof. Suppose $[t_1, t_2]$ is a system busy period. By assumption,

$$\sum_{i=1}^N Q_i(t_1) = \sum_{i=1}^N Q_i(t_2) = 0.$$

Thus

$$\sum_{i=1}^N A_i(t_1, t_2) = \sum_{i=1}^N S_i(t_1, t_2) = t_2 - t_1.$$

Substituting from (12) and rearranging terms:

$$t_2 - t_1 \leq \frac{\sum_{i=1}^N \sigma_i}{1 - \sum_{i=1}^N \rho_i}.$$

$\square$

A simple consequence of this Lemma is that all system busy periods are bounded. Since session delay is bounded by the length of the largest possible system busy period, the session delays are bounded as well. Thus the interval B is finite whenever $\sum_{i=1}^N \rho_i < 1$ and may be infinite otherwise.

We end this section with some comments valid only for the GPS system: Let a **session** i **busy period** be a maximal interval B_i contained in a single system busy period, such that for all $\tau, t \in B_i$:

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{\phi_i}{\phi_j} \quad j = 1, 2, .., N. \tag{23}$$

Notice that it is possible for a session to have zero backlog during its busy period. However, if $Q_i(\tau) > 0$ then τ *must* be in a session i busy period at time τ. We have already shown in (2) that

Lemma 4 *:* For every interval $[\tau, t]$ that is in a session i busy period

$$S_i(\tau, t) \geq (t - \tau) \frac{\phi_i}{\sum_{j=1}^N \phi_j}.$$

Notice that when $\phi = \phi_i$ for all i, the service guarantee reduces to

$$S_i(\tau, t) \geq \frac{t - \tau}{N}.$$

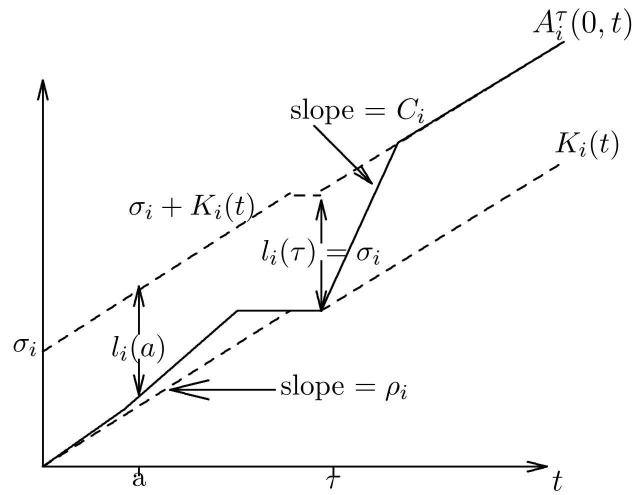


Figure 7: A session i arrival function that is greedy from time τ.

6.2 Greedy Sessions

Session i is defined to be greedy starting at time τ if

$$A_i(\tau, t) = \min\{C_i(t-\tau), l_i(\tau) + (t-\tau)\rho_i\}, \quad \text{for all} \quad t \geq \tau. \tag{24}$$

In terms of the leaky bucket, this means that the session uses as many tokens as possible (i.e. sends at maximum possible rate) for all times $\geq \tau$. At time τ, session i has $l_i(\tau)$ tokens left in the bucket, but it is constrained to send traffic at a maximum rate of C_i. Thus it takes $\frac{l_i^\tau}{C_i - \rho_i}$ time units to deplete the tokens in the bucket. After this, the rate will be limited by the token arrival rate, ρ_i.

Define A_i^τ as an arrival function that is greedy starting at time τ (see Figure 7). From inspection of the figure (and from (24)), we see that if a system busy period starts at time zero, then

$$A_i^0(0,t) \geq A(0,t), \quad \forall A \sim (\sigma_i, \rho_i, C_i), \quad t \geq 0.$$

The major result in this section will be the following:

Theorem 3 *Suppose that $C_j \geq r$ for every session j, where r is the rate of a GPS server. Then for every session i, D_i^* and Q_i^* are achieved (not necessarily at the same time) when every session is greedy starting at time zero, the beginning of a system busy period.*

This is an intuitively pleasing and satisfying result. It seems reasonable that if a session sends as much traffic as possible at all times, it is going to impede the progress of packets arriving from the other sessions. But notice that we are claiming a worst case result, which implies that it is never more harmful for a subset of the sessions to "save up" their bursts, and to transmit them at a time greater than zero.

While there are many examples of service disciplines for which this "all-greedy regime" does not maximize delay, the amount of work required to establish Theorem 3 is still somewhat surprising. Our approach is to prove the Theorem for the case when $C_i = \infty$ for all i—this implies that the links carrying traffic to the server have infinite capacity. This is the easiest case to visualize since we do not have to worry about the input links, and further, it bounds the performance of the finite link speed case, since any session can "simulate" a finite speed input link by sending packets at a finite rate over the link. After we have understood the infinite capacity case it will be shown that a simple extension in the analysis yields the result for finite link capacities as well.

6.3 Generalized Processor Sharing with Infinite Incoming Link Capacities

When all the input link speeds are infinite, the arrival constraint (12) is modified to

$$A_i(\tau, t) \leq \sigma_i + \rho_i(t - \tau), \quad \forall 0 \leq \tau \leq t, \tag{25}$$

for every session i. We say that session i conforms to (σ_i, ρ_i), or $A_i \sim (\sigma_i, \rho_i)$. Further, we stipulate that $\sum_i \rho_i < 1$ to ensure stability.

By relaxing our constraint, we allow step or jump arrivals, which create discontinuities in the arrival functions A_i. Our convention will be to treat the A_i as *left-continuous* functions (i.e. continuous from the left). Thus a session i impulse of size Δ at time 0 yields $Q_i(0) = 0$ and $Q_i(0^+) = \Delta$. Also note that $l_i(0) = \sigma_i$, where $l_i(\tau)$ is the maximum amount of session i traffic that could arrive at time τ^+ without violating (25). When session i is greedy from time τ, the infinite capacity assumption ensures that $l_i(t) = 0$ for all $t > \tau$. Thus (16) reduces to

$$A_i^\tau(\tau, t) = l_i(\tau) + (t - \tau)\rho_i, \quad \text{for all} \quad t > \tau. \tag{26}$$

Note also that if session is greedy after time τ, $l_i(t) = 0$ for any $t > \tau$.

Defining σ_i^τ as before (from 18), we see that it is equal to $Q_i(\tau^+)$ when session i is greedy starting at time τ.

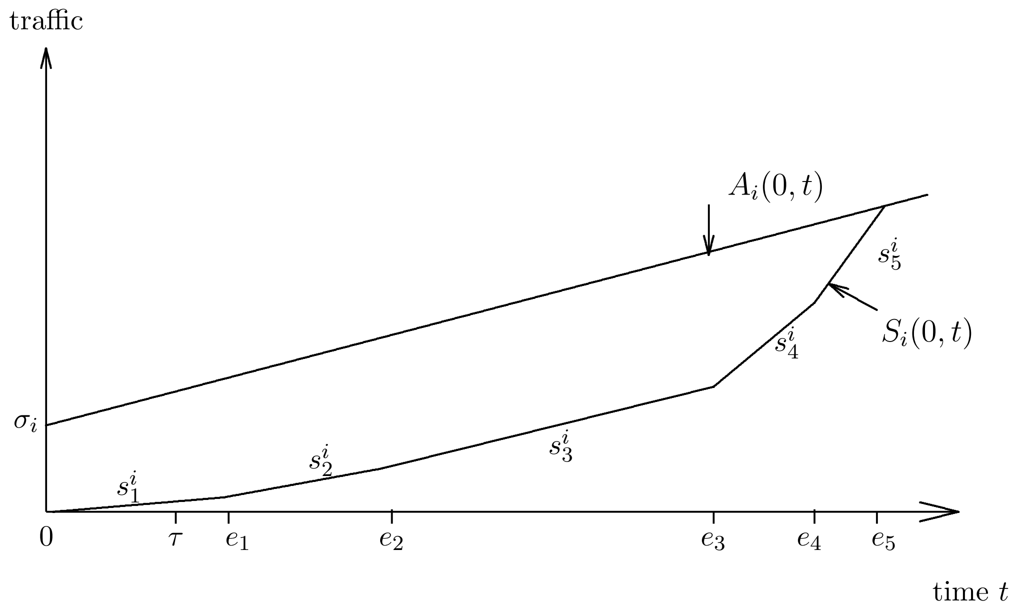


Figure 8: Session i arrivals and departures after 0, the beginning of a system busy period.

6.3.1 An All-greedy GPS system

Theorem 3 suggests that we should examine the dynamics of a system in which all the sessions are greedy starting at time 0, the beginning of a system busy period. This is illustrated in Figure 8.

From (26), we know that

$$A_i(0,\tau) = \sigma_i + \rho_i \tau, \quad \tau \geq 0,$$

and let us assume for clarity of exposition, that $\sigma_i > 0$ for all i. Define e_1 as the first time at which one of the sessions, say $L(1)$, ends its busy period. Then in the interval $[0, e_1]$, each session i is in a busy period (since we assumed that $\sigma_i > 0$ for all i), and is served at rate $\frac{\phi_i}{\sum_{k=1}^{N} \phi_k}$. Since session $L(1)$ is greedy after 0, it follows that

$$\rho_{L(1)} < \frac{\phi_i}{\sum_{k=1}^{N} \phi_k},$$

where $i = L(1)$. (We will show that such a session must exist in Lemma 5.) Now each session j still in a busy period will be served at rate

$$\frac{(1 - \rho_{L(1)})\phi_j}{\sum_{k=1}^{N} \phi_k - \phi_{L(1)}}$$

until a time e_2 when another session, $L(2)$, ends its busy period. Similarly, for each k:

$$\rho_{L(k)} < \frac{(1-\sum_{j=1}^{k-1}\rho_{L(j)})\phi_i}{\sum_{j=1}^{N}\phi_j - \sum_{j=1}^{k-1}\phi_{L(j)}}, \quad k=1,2,...,N \quad ,i=L(k). \tag{27}$$

As shown in Figure 8, the slopes of the various segments that comprise $S_i(0,t)$ are $s_1^i, s_2^i, ...$ From (27):

$$s_k^i = \frac{(1-\sum_{j=1}^{k-1}\rho_{L(j)})\phi_i}{\sum_{j=1}^{N}\phi_j - \sum_{j=1}^{k-1}\phi_{L(j)}}, \quad k=1,2,...,L(i).$$

It can be seen that $\{s_k^i\},\ k=1,2,...,L(i)$ forms an increasing sequence.

Note that

- We only require that
$$0 \le e_1 \le e_2 \le ... \le e_N,$$
allowing for several e_i to be equal.)
- We only care about $t \le e_{L(i)}$ since the session i buffer is always empty after this time.
- Session $L(i)$ has exactly one busy period–the interval $[0, e_i]$.
- e_N is the maximum busy period length, i.e. it meets the bound of Lemma 3.

Any ordering of the sessions that meets (27) is known as a **feasible ordering**. Thus, sessions $1,...,N$ follow a feasible ordering if and only if:

$$\rho_k < \frac{(1-\sum_{j=1}^{k-1}\rho_j)\phi_k}{\sum_{j=k}^{N}\phi_j}, \quad k=1,2,...,N. \tag{28}$$

Lemma 5 *At least one feasible ordering exists if* $\sum_{i=1}^{N}\rho_i < 1$.

Proof. By contradiction. Suppose there exists an index i, $1 \le i \le N$ such that we can label the first $i-1$ sessions of a feasible ordering $\{1,...,i-1\}$, but (28) does not hold for any of the remaining sessions when $k=i$. Then denoting $L_{i-1} = \{1,...,i-1\}$ we have for every session $k \notin L_{i-1}$:

$$\rho_k \ge (1 - \sum_{j\in L_{i-1}}\rho_j)(\frac{\phi_k}{\sum_{j\notin L_{i-1}}\phi_j}).$$

Summing over all such k we have:

$$\sum_{k\notin L_{i-1}}\rho_k \ge 1 - \sum_{j\in L_{i-1}}\rho_j \Rightarrow \sum_{j=1}^{N}\rho_j \ge 1$$

which is a contradiction, since we assumed that $\sum_{j=1}^{N} \rho_j < 1$. Thus no such index i can exist and the Lemma is proven. $\Box$

In general there are many feasible orderings possible, but the one that comes into play at time 0 depends on the σ_i's. For example if $\rho = \rho_j$ and $\phi = \phi_j,\ j = 1, 2, ..., N$, then there are $N!$ different feasible orderings. Similarly, there are $N!$ different feasible orderings if $\rho_i = \phi_i$ for all i. To simplify the notation let us assume that the sessions are labeled so that $j = L(j)$ for $j = 1, 2, ..., N$. Then for any two sessions i, j indexed greater than k we can define a "universal slope" s_k by:

$$s_k = \frac{s_k^i}{\phi_i} = \frac{s_k^j}{\phi_j} = \frac{1 - \sum_{j=1}^{k-1} \rho_j}{\sum_{j=k}^{N} \phi_k}, \quad i, j > k,\ k = 1, 2, ..., N.$$

This allows us to describe the behavior of **all** the sessions in a single figure as is depicted in Figure 9. Under the all-greedy regime, the function $V(t)$ (described in Section 3.1), *corresponds exactly* to the universal service curve, $S(0, t)$, shown in Figure 9. It is worth noting that the virtual time function $V(t)$ captures this notion of generalized service for arbitrary arrival functions.

In the remainder of this section we will prove a tight lower bound on the amount of service a session receives when it is in a busy period: Recall that for a given set of arrival functions $A = \{A_1, ..., A_N\}$, $A^\tau = \{A_1^\tau, ..., A_N^\tau\}$ is the set such that for every session k, $A_k^\tau(0, s) = A_k(0, s)$ for $s \in [0, \tau)$, and session k is greedy starting at time τ.

Lemma 6

Assume that session i is in a busy period in the interval $[\tau, t]$. Then (i) For any subset M of m sessions, $1 \leq m \leq N$, and any time $t \geq \tau$:

$$S_i(\tau, t) \geq \frac{(t - \tau - (\sum_{j \notin M} \sigma_j^\tau + \rho_j(t - \tau)))\phi_i}{\sum_{j \in M} \phi_j}. \tag{29}$$

(ii) Under A^τ, there exists a subset of the sessions, M^t, for every $t \geq \tau$ such that equality holds in (29).

Proof. For compactness of notation, let $\phi_{ji} = \frac{\phi_j}{\phi_i},\ \ \forall i, j$.

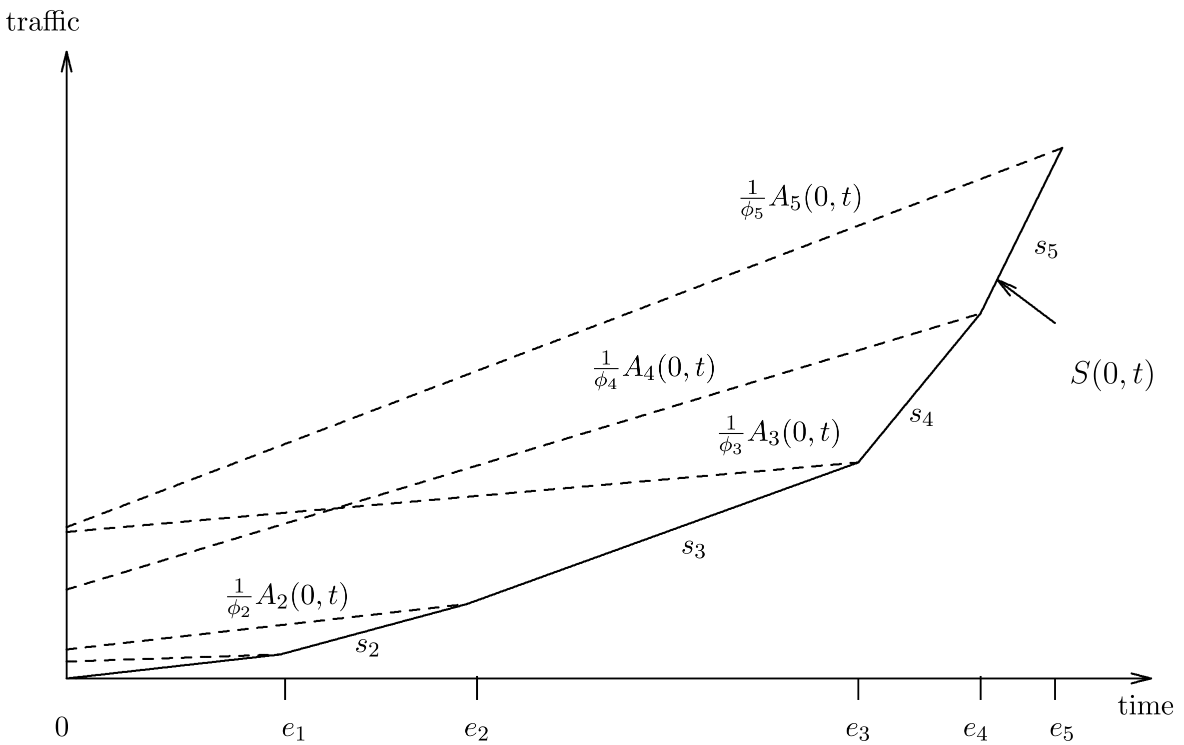
(i) From (22)

$$S_j(\tau, t) \leq \sigma_j^\tau + \rho_j(t - \tau)$$

for all j.

Also, since the interval $[\tau, t]$ is in a session i busy period:

$$S_j(\tau, t) \leq \phi_{ji} S_i(\tau, t).$$



The arrival functions are scaled so that a universal service curve, $S(0,t)$, can be drawn. After time e_i, session i has a backlog of zero until the end of the system busy period, which is at time e_5. The vertical distance between the dashed curve corresponding to session i and $S(0,\tau)$ is $\frac{1}{\phi_i}Q_i(\tau)$, while the horizontal distance yields $D_i(\tau)$ just as it does in Figure 8.

Figure 9: The dynamics of an all-greedy GPS system.

Thus

$$S_j(\tau,t) \leq \min\{\sigma_j^\tau + \rho_j(t-\tau), \phi_{ji}S_i(\tau,t)\}.$$

Since the system is in a busy period, the server serves exactly $t-\tau$ units of traffic in the interval $[\tau,t]$. Thus

$$t-\tau \leq \sum_{j=1}^{N} \min\{\sigma_j^\tau + \rho_j(t-\tau), \phi_{ji}S_i(\tau,t)\}$$

$$\Rightarrow t-\tau \leq \sum_{j\notin M} \sigma_j^\tau + \rho_j(t-\tau) + \sum_{j\in M} \phi_{ji}S_i(\tau,t)$$

for any subset of sessions, M. Rearranging the terms yields (29).
(ii) Since all the sessions are greedy after τ under A^τ, every session j will have a session busy period that begins at τ and lasts up to some time e_j. As we showed in the discussion leading up to Figure 9, $Q_j(t) = 0$, for all $t \geq e_j$. The system busy period ends at time $e^* = \max_j e_j$. Define

$$M^t = \{j : e_j \geq t\}.$$

By the definition of GPS we know that session $j \in M^t$ receives exactly $\phi_{ji}S_i(\tau,t)$ units of service in the interval $(\tau,t]$. A session k is not in M^t only if $e_k < t$, so we must have $Q_k(t) = 0$. Thus, for $k \notin M^t$,

$$S_k(\tau,t) = \sigma_k^\tau + \rho_k(t-\tau),$$

and equality is achieved in (29). □

6.4 An Important Inequality

In the previous section we examined the behavior of the GPS system when the sessions are greedy. Here we prove an important inequality that holds for any arrival functions that conform to the arrival constraints (25).

Theorem 4 *: Let $1,...,N$ be a feasible ordering. Then for any time t and session p:*

$$\sum_{k=1}^{p} \sigma_k^t \leq \sum_{k=1}^{p} \sigma_k.$$

We want to show that at the beginning of a session p busy period, the collective burstiness of sessions $1,...,p$ will never be more than what it was at time 0. The interesting aspect of this theorem is that it holds for **every** feasible ordering of the sessions. When $\rho_j = \rho$, and $\phi_j = \phi$ for every j, it says that the collective burstiness of *any* subset of sessions is no less than what it was at the beginning of the system busy period.

The following three Lemmas are used to prove the theorem. The first says (essentially), that if session p is served at a rate *smaller* than its average rate, ρ_p, during a session p busy period, then the sessions indexed lower than p will be served correspondingly *higher* than their average rates. Note that this lemma is true even when the sessions are not greedy.

Lemma 7 *Let* $1, ..., N$ *be a feasible ordering, and suppose that session* p *is busy in the interval* $[\tau, t]$. *Further, define* x *to satisfy*

$$S_p(\tau, t) = \rho_p(t - \tau) - x \tag{30}$$

Then

$$\sum_{k=1}^{p-1} S_k(\tau, t) > \sum_{k=1}^{p-1}(t - \tau)\rho_k + x(1 + \sum_{j=p+1}^{N} \frac{\phi_j}{\phi_p}). \tag{31}$$

Proof. For compactness of notation, let $\phi_{ij} = \frac{\phi_i}{\phi_j}, \quad \forall i, j$. Now because of the feasible ordering,

$$\rho_p < \frac{1 - \sum_{j=1}^{p-1} \rho_i}{\sum_{i=p}^{N} \phi_{ip}}.$$

Thus

$$S_p(\tau, t) < (t - \tau)\left(\frac{1 - \sum_{j=1}^{p-1} \rho_i}{\sum_{i=p}^{N} \phi_{ip}}\right) - x. \tag{32}$$

Also, $S_j(\tau, t) \le \phi_{jp} S_p(\tau, t)$, for all j. Thus

$$\sum_{j=p}^{N} S_j(\tau, t) \le S_p(\tau, t) \sum_{j=p}^{N} \phi_{jp}.$$

Using (32)

$$\sum_{j=p}^{N} S_j(\tau, t) < (t - \tau)(1 - \sum_{j=1}^{p-1} \rho_j) - x \sum_{j=p}^{N} \phi_{jp}.$$

Since $[\tau, t]$ is in a system busy period:

$$\sum_{j=p}^{N} S_j(\tau, t) = (t - \tau) - \sum_{j=1}^{p-1} S_j(\tau, t).$$

Thus

$$(t - \tau) - \sum_{j=1}^{p-1} S_j(\tau, t) < (t - \tau)(1 - \sum_{j=1}^{p-1} \rho_j) - x \sum_{j=p}^{N} \phi_{jp}$$

$$\Rightarrow \sum_{j=1}^{p-1} S_j(\tau,t) > (t-\tau)\sum_{j=1}^{p-1}\rho_j + x(1+\sum_{j=p+1}^{N}\phi_{jp}),$$

since $\phi_{pp} = 1$. □

Lemma 8 *Let $1,...,N$ be a feasible ordering, and suppose that session p is busy in the interval $[\tau, t]$. Then if $S_p(\tau,t) \leq \rho_p(t-\tau)$:*

$$\sum_{k=1}^{p} S_k(\tau,t) > (t-\tau)\sum_{k=1}^{p}\rho_k, \tag{33}$$

Proof. Let

$$S_p(\tau,t) = \rho_p(t-\tau) - x,$$

$x \geq 0$. Then from (31) we have are done, since $x\sum_{j=p+1}^{N}\frac{\phi_j}{\phi_p} \geq 0$. □

Lemma 9 *Let $1,...,N$ be a feasible ordering, and suppose that session p is busy in the interval $[\tau, t]$. Then if $S_p(\tau,t) \leq \rho_p(t-\tau)$:*

$$\sum_{k=1}^{p}\sigma_k^t \leq \sum_{k=1}^{p}\sigma_k^\tau.$$

Proof. From Lemma 2, for every k,

$$\sigma_k^\tau + \rho_k(t-\tau) - S_k(\tau,t) \geq \sigma_k^t.$$

Summing over k, and substituting from (33), we have the result. □

If we choose τ to be the beginning of a session p busy period, then Lemma 9 says that if $S_p(\tau,t) \leq \rho_p(t-\tau)$ then

$$\sigma_p^t + \sum_{k=1}^{p-1}\sigma_k^t \leq \sigma_p + \sum_{k=1}^{p-1}\sigma_k^\tau. \tag{34}$$

Now we will prove Theorem 4:

Proof. (Of Theorem 4): We proceed by induction on the index of the session p.

Basis: $p = 1$. Define τ to be the last time at or before t such that $Q_1(\tau) = 0$. Then session 1 is in a busy period in the interval $[\tau, t]$, and we have

$$S_1(\tau,t) \geq \frac{(t-\tau)\phi_1}{\sum_{k=1}^{N}\phi_k} > (t-\tau)\rho_1.$$

The second inequality follows since session 1 is first in a feasible order, implying that $\rho_1 < \frac{\phi_1}{\sum_{k=1}^{N} \phi_k}$. From Lemma 2:

$$\sigma_1^t \le \sigma_1^\tau + \rho_1(t-\tau) - S_1(\tau,t) < \sigma_1^\tau \le \sigma_1.$$

This shows the basis.

Inductive Step: Assume the hypothesis for $1, 2, ..., p-1$ and show it for p. Observe that if $Q_i(t) = 0$ for any session i then $\sigma_i^t \le \sigma_i$. Now consider two cases:

Case 1: $\sigma_p^t \le \sigma_p$: By the induction hypothesis:

$$\sum_{i=1}^{p-1} \sigma_i^t \le \sum_{i=1}^{p-1} \sigma_i.$$

Thus

$$\sum_{i=1}^{p} \sigma_i^t \le \sum_{i=1}^{p} \sigma_i.$$

Case 2: $\sigma_p^t > \sigma_p$: Session p must be in a session p busy period at time t, so let τ be the time at which this busy period begins. Also, from (22): $S_p(\tau,t) < \rho_p(t-\tau)$. Applying (34):

$$\sigma_p^t + \sum_{k=1}^{p-1} \sigma_k^t \le \sigma_p + \sum_{k=1}^{p-1} \sigma_k^\tau \le \sum_{k=1}^{p} \sigma_k, \tag{35}$$

where in the last inequality, we have used the induction hypothesis. □

6.5 Proof of the main result

In this section we will use Lemma 6 and Theorem 4 to prove Theorem 3 for infinite capacity incoming links.

Let $\hat{A}_1, ..., \hat{A}_N$ be the set of arrival functions in which all the sessions are greedy from time 0, the beginning of a system busy period. For every session p, let $\hat{S}_p(\tau,t)$, and $\hat{D}_p(t)$ be the session p service and delay functions under $\hat{A}$. We first show

Lemma 10 *Suppose that time t is contained in a session p busy period that begins at time τ: Then*

$$\hat{S}_p(0, t-\tau) \le S_p(\tau,t). \tag{36}$$

Proof. Define $\mathcal{B}$ as the the set of sessions that are busy at time $t-\tau$ under $\hat{A}$. From Lemma 6:

$$S_p(\tau,t) \ge \frac{(t-\tau-\sum_{i\notin\mathcal{B}}(\sigma_i^\tau + \rho_i(t-\tau)))\phi_i}{\sum_{j\in\mathcal{B}}\phi_j}$$

Since the order in which the sessions become inactive is a feasible ordering, Theorem 4 asserts that:

$$S_p(\tau,t) \geq \frac{(t-\tau-\sum_{i\notin\mathcal{B}}(\sigma_i+\rho_i(t-\tau)))\phi_i}{\sum_{j\in\mathcal{B}}\phi_j}$$
$$= \hat{S}_i(0,t-\tau),$$

(from Lemma 6), and (36) is shown. □

Lemma 11 *For every session i, D_i^* and Q_i^* are achieved (not necessarily at the same time) when every session is greedy starting at time zero, the beginning of a system busy period.*

Proof. We first show that the session i backlog is maximized under $\hat{A}$: Consider any set of arrival functions, $A = \{A_1, ..., A_N\}$ that conforms to (25), and suppose that for a session i busy period that begins at time τ:

$$Q_i(t^*) = \max_{t\geq\tau} Q_i(t).$$

From Lemma 10:

$$\hat{S}_i(0,t^*-\tau) \leq S_i(\tau,t^*),$$

Also,

$$A_i(\tau,t^*) \leq \sigma_i + \rho_i(t-\tau) = \hat{A}_i(0,t^*-\tau).$$

Thus

$$\hat{A}_i(0,t^*-\tau) - \hat{S}_i(0,t^*-\tau) \geq A_i(\tau,t^*) - S_i(\tau,t^*)$$

i.e.

$$\hat{Q}_i(t^*-\tau) \geq Q_i(t^*).$$

The case for delay is similar: Consider any set of arrival functions, $A = \{A_1, ..., A_N\}$ that conforms to (25), and for a session i busy period that begins at time τ, let t^* be the smallest time in that busy period such that:

$$D_i(t^*) = \max_{t\geq\tau} D_i(t).$$

From the definition of delay in equation (17):

$$A_i(\tau,t^*) - S_i(\tau,t^* + D_i(t^*)) = 0.$$

Let us denote $d_i^* = t^* - \tau$. From Lemma 10:

$$\hat{S}_i(0, d_i^* + D_i(t^*)) \leq S_i(\tau, t^* + D_i(t^*))$$

and since $\sigma_i \geq \sigma_i^\tau$:

$$\hat{A}_i(0, d_i^*) \geq A_i(\tau, t^*).$$

Thus

$$\hat{A}_i(0, d_i^*) - \hat{S}_i(0, d_i^* + D_i(t^*)) \geq A_i(\tau, \tau + t^*) - S_i(\tau, t^* + D_i(t^*)) = 0$$

$$\Rightarrow \hat{D}_i(d_i^*) \geq D_i(t^*).$$

□

Thus we have shown Theorem 3 for infinite capacity incoming links.

7 Generalized Processor Sharing with Finite Link Speeds

In the infinite link capacity case we were able to take advantage of the fact that a session could use up all of its outstanding tokens instantaneously. In this section we include the maximum rate constraint, i.e. for every session i, the incoming session traffic can arrive at a maximum rate of $C_i \geq 1$. Although this can be established rigorously [16], it is not hard to see that Theorem 3 still holds: Consider a given set of arrival functions for which there is no peak rate constraint. Now consider the intervals over which the a particular session i is backlogged when the arrivals reach the server through (a) infinite capacity input links, and (b) input links such that $1 \leq C_j$ for all j and $C_k < \infty$ for at least one session k. Since the server cannot serve any session at a rate of greater than 1, the set of intervals over which session i is backlogged is identical for the two cases. This argument holds for every session in the system, implying that the session service curves are identical for cases (a) and (b). Thus Lemma 10 continues to hold, and Theorem 3 can be established easily from this fact. We have not been able to show that Theorem 3 holds when $C_j < 1$ for some sessions j, but delay bounds calculated for the case $C_j = 1$ (or $C_j = \infty$) apply to such systems, since any link of capacity 1 (or ∞) can simulate a link of capacity less than 1.

8 The Output Burstiness, σ_i^{out}

In this section we focus on determining for every session i, the least quantity σ_i^{out} such that

$$S_i \sim (\sigma_i^{\text{out}}, \rho_i, r)$$

where r is the rate of the server. This definition of output burstiness is due to Cruz [5]. (To see that this is the best possible characterization of the output process, consider the case in which session i is the only active session, and is is greedy from time zero. Then, a peak service rate of r, and a maximum sustainable average rate of ρ_i are both achieved.) By characterizing S_i in this manner we can begin to analyze networks of servers, which is the focus of the sequel to this paper. Fortunately there is a convenient relationship between σ_i^{out} and Q_i^*:

Lemma 12 *If $C_j \geq r$ for every session j, where r is the rate of the server, then for each session i:*

$$\sigma_i^{\mathrm{out}} = Q_i^*.$$

Proof. First consider the case $C_i = \infty$. Suppose that Q_i^* is achieved at some time t^*, and session i continues to send traffic at rate ρ_i after t^*. Further, for each $j \neq i$, let t_j be the time of arrival of the last session j bit to be served before time t^*. Then Q_i^* is also achieved at t^* when the arrival functions of all the sessions $j \neq i$ are truncated at t_j, i.e., $A_j(t_j, t) = 0$, $j \neq i$. In this case, all the other session queues are empty at time t^*, and beginning at time t^*, the server will exclusively serve session i at rate 1 for $\frac{Q_i^*}{1-\rho_i}$ units of time, after which session i will be served at rate ρ_i. Thus

$$S_i(t^*, t) = \min\{t - t^*, Q_i^* + \rho_i(t^* - t)\}, \quad \forall t \geq t^*.$$

From this we have

$$\sigma_i^{\mathrm{out}} \geq Q_i^*.$$

We now show that the reverse inequality holds as well: For any $\tau \leq t$:

$$\begin{aligned} S_i(\tau, t) &= A_i(\tau, t) + Q_i(\tau) - Q_i(t) \\ &\leq l_i^\tau + \rho_i(t - \tau) + Q_i(\tau) - Q_i(t) \\ &= \sigma_i^\tau - Q_i(t) + \rho_i(t - \tau) \end{aligned}$$

(since $C_i = \infty$.) This implies that

$$\sigma_i^{\mathrm{out}} \leq \sigma_i^\tau - Q_i(t) \leq \sigma_i^\tau \leq Q_i^*.$$

Thus

$$\sigma_i^{\mathrm{out}} = Q_i^*.$$

Now suppose that $C_i \in [r, \infty)$. Since the traffic observed under the all-greedy regime is indistinguishable from a system in which all the incoming links have infinite capacity, we must have $\sigma_i^{\text{out}} = Q_i^*$ in this case as well. $\Box$

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[2] A. Birman, P. C. Chang, J. S. C. Chen, and R. Guerin, *Buffer sizing in an ISDN frame relay switch*, Tech. Rep. RC14386, IBM Research, August 1989.

[3] A. Birman, H. R. Gail, S. L. Hantler, Z. Rosberg, and M. Sidi, *An optimal policy for buffer systems*, Tech. Rep. RC16641, IBM Research, March 1991.

[4] I. Cidon, I. Gopal, G. Grover, and M. Sidi, *Real time packet switching: A performance analysis*, IEEE Journal on Selected Areas in Communications, SAC-6 (1988), pp. 1576–1586.

[5] R. L. Cruz, *A calculus for network delay, Part I: Network elements in isolation*, IEEE Transactions on Information Theory, 37 (1991), pp. 114–131.

[6] ——, *A calculus for network delay, Part II: Network analysis*, IEEE Transactions on Information Theory, 37 (1991), pp. 132–141.

[7] A. Demers, S. Keshav, and S. Shenkar, *Analysis and simulation of a fair queueing algorithm*, Internetworking Research and Experience, 1 (1990).

[8] H. R. Gail, G. Grover, R. Guerin, S. L. Hantler, Z. Rosberg, and M. Sidi, *Buffer size requirements under longest queue first*, Tech. Rep. RC14486, IBM Research, January 1991.

[9] S. J. Golestani, *Congestion-free transmission of real-time traffic in packet networks*, in Proceedings of IEEE INFOCOM '90, San Fransisco, CA, 1990, pp. 527–536.

[10] ——, *A framing strategy for connection managment*, in Proceedings of SIGCOMM '90, 1990.

[11] ——, *Duration-limited statistical multiplexing of delay sensitive traffic in packet networks*, in Proceedings of IEEE INFOCOM '91, 1991.

[12] A. C. Greenberg and N. Madras, *How fair is fair queueing?*, Jounal of the ACM, 3 (1992).

[13] E. Hahne, *Round Robin Scheduling for Fair Flow Control*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, December 1986.

[14] L. Kleinrock, *Queueing Systems Volume 2: Computer Applications*, John Wiley and Sons, New York, 1976.

[15] C. Lu and P. R. Kumar, *Distributed scheduling based on due dates and buffer prioritization*, Tech. Rep., University of Illinois, 1990.

[16] A. K. Parekh, *A Generalized Processor Sharing Aproach to Flow Control in Integrated Services Networks*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, February 1992.

[17] A. K. Parekh and R. G. Gallager, *A Generalized Processor Sharing approach to flow control—The Multiple Node Case*, Tech. Rep. 2076, Laboratory for Information and Decision Systems, MIT, 1991.

[18] J. R. Perkins and P. R. Kumar, *Stable distributed real-time scheduling of flexible manufacturing systems*, IEEE Transactions on Automatic Control, AC-34 (1989), pp. 139–148.

[19] G. Sasaki, *Input buffer requirements for round robin polling systems*, in Allerton Conference on Communication, Control and Computing, 1989.

[20] J. Turner, *New directions in commnunications (or Which way to the information age)*, IEEE Communications Magazine, 24 (1986), pp. 8–15.

[21] L. Zhang, *A New Architecture for Packet Switching Network Protocols*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, August 1989.