A Generalized Processor Sharing Approach to Flow Control In Integrated Services Networks

by

Abhay Kumar J. Parekh

B.E.S., The Johns Hopkins University
S.M., Massachusetts Institute of Technology

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1992

Author .....
Department of Electrical Engineering and Computer Science
January 24, 1992

Certified by .....
Robert G. Gallager
Fujitsu Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

A Generalized Processor Sharing Approach to Flow Control In Integrated Services Networks

by

Abhay Kumar J. Parekh

Submitted to the Department of Electrical Engineering and Computer Science
on January 24, 1992, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The problem of how to assign network resources to the new users of an integrated services network is investigated in the context of rate based flow control. In packet networks that support varied traffic types such as voice, video, image and interactive data, many of the services may have stringent delay requirements on *every packet* submitted to the network. Thus, just providing average performance guarantees may not be adequate for these services. We propose a flexible and efficient multiplexing scheme called Generalized Processor Sharing (GPS) that allows the network to make worst-case performance guarantees. The scheme is flexible, in that one user may receive a "better" quality of service without adversely affecting the performance guarantees of the others. In addition, GPS is shown to be *fair* to *every* packet that enters the system unlike other multiplexing schemes such as First-Come-First-Serve. A practical packet-by-packet transmission scheme that closely approximates Generalized Processor Sharing is also presented. This allows us to relate results for GPS to the packet-by-packet scheme in a precise manner. Both GPS and PGPS were suggested earlier in [6], in the context of managing congestion in gateways.

The performance of a single server GPS system is analyzed from the standpoint of worst-case delay and burstiness when the sources are constrained by leaky buckets. The analysis yields a simple resource assignment scheme that allows the server to make worst-case delay and rate guarantees to every session in the system. Other multiplexing schemes such as FCFS and Strict Priority are also analyzed in the context of sources that are leaky bucket constrained.

Next, we extend the results for the single server case to provide worst-case bounds on session delay and backlog for several broad classes of GPS networks. In Rate Proportional Processor Sharing Networks (RPPS), the server parameters are assigned in proportion to the average session rates. We bound the worst-case delay and backlog for every session in terms of its leaky bucket parameters. It is interesting that these bounds are independent of the behavior of the other sessions, and also of the network topology. Extensions of RPPS are discussed in which the same bounds on delay and backlog hold for a subset of the sessions.

We then show how to characterize the internal traffic (in terms of average rate and burstiness) for any acyclic network, and also for a broad class of non-acyclic networks called Consistent Relative Session Treatment networks. Next, we provide a method, based on an independent sessions assumption, to obtain bounds on session delay and backlog for any network for which internal traffic has been characterized. Our method analyzes the session i route as a whole, and yields bounds that are much better than those that result from adding the worst-case delays (backlogs) at each of the servers encountered by a session. We bound delay and backlog for each session, based on an efficiently computable universal service curve, and show that a "staggered" greedy regime achieves the worst-case bounds under the independent sessions assumption. Propagation delay is also incorporated into the model.

Finally, we relate our analysis of arbitrary topology GPS networks to arbitrary topology PGPS networks. We find that for small packet sizes, the behavior of the two schemes is virtually identical, and demonstrate the effectiveness of PGPS in guaranteeing worst-case session delay under Rate Proportional Processor Sharing assignments.

Thesis Supervisor: Robert G. Gallager
Title: Fujitsu Professor of Electrical Engineering

The voyage of discovery consists not in seeking new landscapes,
but in seeing through new eyes....
Marcel Proust, Remembrance of Things Past

Acknowledgements

Robert Gallager enabled my return to MIT, arranged for my funding, and supervised this thesis with care, consideration and interest. Undaunted by my obstinacy, he waged a somewhat quixotic campaign to make me write more clearly, for which I am now very grateful. His remarkable insight contributed substantially to the results of this thesis, and his high standards of scholarship and intellectual integrity inspired me to challenge myself much more than I would have otherwise. I will always be grateful to him for believing in my abilities, and for helping make my first significant research experience an extremely enjoyable one.

I would like to thank the members of my thesis committee—David Clark, Dimitri Bertsekas and Pierre Humblet, for taking a genuine interest in my work. Dr. Clark inspired me with his enthusiasm and made sure that I did not lapse into theoretical irrelevancy. When he was trapped with me on a flight to Urbana, Prof. Bertsekas made best of the situation by taking some of my results and rewriting them so clearly, that I actually understood them better when he was done. I would also like to thank him for his words of encouragement. Professor Humblet took the time to almost provide a simple proof of the major result in Chapter 2—I am grateful that he did not take *enough* time to actually construct this proof! The helpful discussions I have had on academic matters with Professors Sanjoy Mitter and John Tsitsiklis are much appreciated.

The lively social and intellectual atmosphere of the Laboratory for Information Decision Systems helped make my stay at MIT a lot of fun. I would like to thank every individual who killed an hour in a "reading-room argument" with me. I have thoroughly enjoyed my interactions with Rick Barry, Chee-seng Chow, Sheila Hegarty, Diane Ho, Rick Hughes, Sanjeev Kulkarni, Chongwan Lee, Rajesh Pankaj, Tom and Karen Richardson, Serap Sevari, Jane Simmons, Bruno Suard, Emre Telatar, David Tse, Paul Tseng, Manos Varvarigos, Roy Yates, John Young and many others, whose omission can only be attributed to egregious oversight. In addition, I would like to thank Kathleen O'Sullivan and Nancy Young for shielding me from the morass of MIT's bureaucracy.

I am grateful to my wife Kadambari for putting up with a hyperactive and nocturnal graduate student for four long years without so much as batting an eyelid. Besides supporting me financially, she has given me all the understanding, love and friendship that I could ever desire. She is truly wonderful!

Finally, and most importantly, I would like to acknowledge the immeasurable contribution of my parents, whose love, sacrifices and support have enabled me to live my dreams. It is seeing through their eyes that that has made discovery possible in a hundred different universes. This thesis is dedicated to them with the utmost love and respect.

Sponsor Acknowledgements

This thesis was funded by a Vinton Hayes Fellowship and a Center for Intelligent Control Fellowship under contract DAAL03-86-K-0171 with the Army Research Office. These funds gave me great independence to pursue my ideas, and I am thankful for the support.

Contents

List of Figures

Chapter 1

Introduction

The problem of how to assign the resources of a wide area high speed network to its users is central to providing real-time services such as voice and video. This problem becomes especially difficult when sessions that have stringent performance requirements contend for these resources. In this thesis we will investigate the issue of resource allocation from the standpoint of rate-based flow control in a packet switched integrated services network. We will provide a framework based on a multiplexing discipline called generalized processor sharing, within which it is possible for the network to meet the differing performance requirements of a wide variety of traffic types. Further, these requirements can be met even when the short term demand for link usage frequently exceeds usable link capacity.

By a high speed wide area network we mean one that is geographically distributed over a fairly large area, and for which the underlying topology does not have a regular structure such as a ring, or a mesh. The network is composed of heterogeneous switching and transmission equipment, but a significant part of it utilizes optical technology, which makes it possible to support high bandwidth services such as video.

There are two ways in which high speed wide area networks are likely to emerge. The first is as a backbone network for local area networks. LANs allow a variety of services such as load balancing and distributed file storage to be implemented, and

as users get used to these services the market for such interconnected LANs can only grow. The second way in which such wide area networks can become a reality is through the implementation of a standard such as Broadband ISDN, under which the telephone network will be transformed into a high speed wide area network that provides voice, video, data and image transport. Without speculating on the exact form that wide area high speed networks will take, it is clear that there is a demand for them, and that people will try to build them in the near future.

Historically, transmission bandwidth has been viewed as a scarce resource in wide area networks. However, the enormous bandwidth offered by optical fiber relative to the speeds of the electronic devices, has shifted the bottleneck to processing overhead in a dramatic fashion. Network layer protocols now attempt to limit computation in order to get the data through processing nodes in a timely manner, and may even "waste bandwidth" in order to do this. Another way in which optical communication technology has affected protocol design is in the area of error control. The negligible bit error rates of optical fiber has made it possible to implement error control as an end-to-end function, while data link control is an important function in non-optical data networks.

However, the most fundamental difference between traditional and high speed networks *is the different ways in which they are to be used.* High speed networks will offer services that span a staggeringly wide range of rates–from about 5b/s to 500 Mb/s. Further, the throughput and delay requirements of these services will differ widely as well. Applications such as voice and video will have stringent constraints on message delay, for *all* messages. In fact, the primary source of errors in high speed networks may be from packets for which the network does not meet these delay requirements. Buffer overflow might also continue to be significant even though the cost of storage continues to decrease rapidly with time.

The range of service requirements appears even more daunting when one considers what *today's* wide area networks offer in terms of performance. Datagram based

networks carry traffic that spans a fairly wide range of rates, but their performance guarantees are of the meagre "best effort" variety. Circuit switched voice networks provide excellent performance guarantees, but are extremely inflexible in the range of services they offer. Thus, traditional networks trade off performance guarantees with flexibility, but *integrated services networks require both.* It is tempting to resolve this problem by providing transport on more than one kind of network. For example, all the bursty traffic with lax performance requirements could be packet switched, and the more delay sensitive, steady traffic could be circuit switched on a separate network. However, it seems clear that if a single network can support a large diversity of services, and can also realize the economies of scale that go with integrated solutions, then there is no point in building several specialized networks that cost more. In short, it seems reasonable to attempt a design of an integrated services network before passing judgement on the issue.

If the network is to be an integrated services network, the next question that arises is how the data should be switched. We will assume it is packet switched for two reasons: First, circuit switching is inherently inflexible at the access points of the network since it forces session bandwidths to be multiples of a base slot size—for example data-rates provided under narrow band ISDN are all multiples of 64 Kb/s. Second, circuit switching does not use bandwidth efficiently, and we believe it is important to understand the problem of resource allocation when there is a significant amount of *contention* for the resources. This is based on the assumption that the increase in demand of high bandwidth services will keep pace with the supply of usable link capacity.

Since quality of service is so important, we can rule out datagram delivery for real time traffic such as voice, since it provides only "best effort" performance. From now on, we will assume that the network is a virtual circuit packet network.[1]

[1] Of course, datagram service can also be provided on such a network.

1.1 Flow Control

The level of performance that can be assured to users is intimately related to the methods employed to admit traffic into the network. In the telephone network, voice calls get "blocked" unless resources are available at call set up that can be *dedicated* for the duration of the call. This results in good performance guarantees for the calls that go through. On the other hand, data networks attempt to control the session input rate in "real-time" depending on the level of congestion in the network. Since these schemes are designed to utilize the network as fully as possible, a session can receive very bad service when the network is heavily loaded. Thus in building a network that accommodates a large variety of traffic types, it is crucial to examine the problem of how to regulate the incoming traffic. This network function is known as Flow Control.

The purpose of flow control is to regulate the admission of traffic into the network in order to minimize network overload, without unduly restraining sessions when the network is lightly loaded. Since the level of network congestion may vary across different regions of the network, flow control mechanisms must be able to react in accordance with the traffic conditions local to the routes taken by the sessions being actively controlled. Flow control is particularly important in high speed networks for two reasons: First, more is demanded of the network in terms of performance guarantees, and second, many more packets are submitted to the network, leading to potentially worse problems of buffer overflow.

Since a flow control scheme limits access to the network, it is important that it be "fair" to all the users. Fairness problems fall under three major categories:

- Fairness to each user attempting to establish a session: it may be possible for a few large sessions to seize so many network resources that new sessions cannot be established.

- Fairness to each session already established: A few of the sessions already es-

tablished may be able to achieve high data rates (or low delay) at the expense of another, poorly treated session.

- Fairness to each packet of an established session: The largest delay faced by a packet of a given session may be much greater than the average delay faced by the packets of the session. This kind of fairness is particularly important when performance guarantees apply to each packet of a real-time session.

Despite the large body of analytical work in these areas, the notion of fairness remains essentially a subjective one. What is fair is largely a function of what purpose the network is designed to serve. When worst-case performance guarantees are to be given to real-time sessions it is desirable to limit the the variability in delay over the packets of a session—i.e. the third category of fairness assumes considerable importance.

Of the many approaches to flow control that have been suggested in the literature we will summarize just two. For a more complete treatment the reader is referred to [1] and [9].

1.1.1 Window-based Schemes

Most traditional data networks use window-based flow control. In these schemes, there is a stipulated upper bound (the window size) on the number of unacknowledged packets submitted to the network at any given time. Under end-to-end schemes, the window is set up between source and destination, while in node-by-node schemes it is set up for every link traversed by the session traffic. The level of network congestion, is reflected in the time taken for acknowledgments to reach the source, and consequently slows the source down. On the other hand, a source can take advantage of lightly loaded conditions, since acknowledgments will be relatively unhindered by queueing delays. Thus the goal of a window-based scheme is to control the source in real-time through feedback that reflects the level of network congestion.

There are two important advantages of window based schemes. First, they are

easy to implement. Second, there are few known mechanisms that respond faster to congestion in the network than do windows. Thus, if flow control is to be done through feedback, it seems clear that window-based schemes should be leading candidates.

However the effectiveness of feedback based congestion control schemes in high speed wide area networks is severely limited by propagation delay. This is best seen by an example: Suppose a 1 Gb/s transmission line connects nodes A and B which are 1000 miles apart. Assume that traffic flows from A to B, and that the feedback (permits in the case of windows) from B travel on another 1 Gb/s link. The propagation delay is about 10 ms, which implies that there will be about $10Mb$ of traffic in "flight" when the link is being used. Now suppose that B is congested and wants to cut back on the amount of traffic received from A. Then B will receive 20 Mb of traffic before any feedback to A can take effect. This can result in buffers being over-run, and packets being dropped. In fact, a single dropped packet may result in the entire message (of which it is a part) being retransmitted, which further increases congestion and the likelihood of buffers being over-run.

More importantly, windows are not well suited to providing performance guarantees. This is particularly true when no other access control mechanism is used in conjunction with windows. When the required session rates are high, window sizes must be large—otherwise throughput is severely limited by round trip delay. When many such large sessions are active at the same time, congestion sets in and slows down all the packets in the network. In this case, high throughput may be achieved at the expense of large packet delays. Since it is difficult to predict the number of large sessions that will be active during the duration of a newly arriving session, even average rate guarantees cannot be made.

A number of adjustable window schemes have been suggested, in which the window size fluctuates with time. Typically, the window size starts out by being small, and is expected to stabilize with time to a level that is consistent with the degree of network congestion. While there are merits to this approach, the feedback delay

problem may lead to oscillating window sizes. Since the effects of feedback delay on congestion control do not appear to be well understood either in theory or simulation, it is difficult to determine under what general conditions these oscillations will lead to severe congestion, and so adjustable window schemes have drawbacks when being used to provide performance guarantees.

1.1.2 Rate based Schemes

The need to accommodate traffic with stringent rate and delay requirements limits us to consider schemes in which guarantees can be made on meeting these requirements (with high probability). 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 *Resource Assignment* function of the flow control scheme 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. Resource assignment is done by allocating resources within the network—it is a complicated function and currently very poorly understood.[2] Yet, without a method for resource assignment, rate based flow control is really quite ineffective and in fact, seems pointless. In [3] the lack of an effective resource assignment function is cited as "an important hurdle that must be crossed before rate control can be used in production protocols." *This thesis will concentrate on understanding and providing insight on the resource assignment problem.*

Once the session begins submitting packets, the network ensures that the source behavior is consistent with the agreed upon values of the traffic parameters. This

[2]Of course in a network with abundant resources (transmission and switching), one could use bandwidth inefficient schemes such as time or frequency division multiplexing to do resource assignment, but we will assume in this thesis that such wide area networks are not likely to be the norm for the foreseeable future.

network function is known as *Rate Enforcement.* The most popular form of rate enforcement, and the one we will use in this thesis is called Leaky Bucket admission control [23]. 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 1.7), 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 [4]. In this thesis we will explore ways in which *the packet delay in the network* can be estimated given that input traffic is constrained by leaky buckets, and will not consider the delay in the leaky bucket. Leaky buckets are discussed in more detail in Section 1.7.

1.1.3 Resource Assignment

As explained in the Section 1.1.2, a flow control scheme that guarantees network performance will have to rely on how resources are allocated within the network. Network resources consist of links, switches, and devices that process packet headers, service requests etc. A packet spends a certain amount of time using a resource, but typically must wait for it to become available. Thus there is an intimate connection between delay and how the resource assignment is done. Packet delay in the network can be expressed as the sum of the following four components:

- Processing delay.
- Queueing delay.

- Transmission delay.
- Propagation delay.

In this thesis we will focus exclusively on how to limit queueing delay through resource assignment. We view the resources as links and investigate the effects of various generalized multiplexing techniques on delay. Thus our study of resource assignment is a study of multiplexing in the network.

1.2 Multiplexing

Multiplexing is the allocation of link capacity to competing traffic streams–it is the function that enables the sharing of a link. The manner in which multiplexing is performed has "a profound effect on packet delay" [1], and consequently forms an important component of the resource assignment function. We have already pointed out that given our focus on resource contention, regular time and frequency multiplexing are too wasteful of the network resources to be considered as candidate multiplexing disciplines. In addition to being efficient, a good scheme should allow the network to treat users differently, in accordance with their negotiated quality of service. However, this flexibility should not compromise the *integrity* 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. Finally, the scheme should be analyzable since performance guarantees are to be given. Thus the challenge is to find schemes that are *flexible*, *efficient* and *analyzable*.

1.3 Outline of the Chapter

So far in this chapter, we have established the connection between rate-based flow control and performance guarantees. We have also argued that an integrated services network should be flexible and use its links efficiently, *in addition* to providing per-

formance guarantees. Understanding the resource assignment function was identified as the focus of this thesis.

In the remainder of this chapter we will focus on the two ingredients of our approach to flow control—Generalized Processor Sharing (GPS) and Leaky Bucket admission control. GPS is efficient, fair and flexible, but it assumes a fluid traffic model, and is not packet-based. In Section 1.5 we will describe a packet-by-packet scheme, called Packet-by-Packet Generalized Processor Sharing (PGPS), and prove that it closely approximates GPS. A practical implementation of this scheme based on Virtual Time is presented in Section 1.5.1. Next, PGPS is compared to a wide variety of other service disciplines—this comparison is rather long, but necessary in order to understand the merits of PGPS.

The Leaky Bucket is described in Section 1.7. The output of the bucket (which is the input to the network) is succinctly characterized in the form of an arrival constraint. The validity of worst-case analysis, and other issues pertaining to models of delay in Integrated Services networks are discussed in Section 1.8. Finally, the structure of the rest of this thesis is outlined in Section 1.9.

1.4 Generalized Processor Sharing

A generalized processor sharing (GPS) server is characterized by N positive real numbers, $\phi_1, \phi_2, ..., \phi_N$. The server operates at a fixed rate r and is work conserving. Let $S_i(\tau, t)$ be the amount of session i traffic served in an interval $[\tau, t]$. Then a GPS server is defined as one for which

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}, \quad j = 1, 2, ..., N \tag{1.1}$$

for any session i that is backlogged throughout 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

$$g_i = \frac{\phi_i}{\sum_j \phi_j} r. \tag{1.2}$$

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

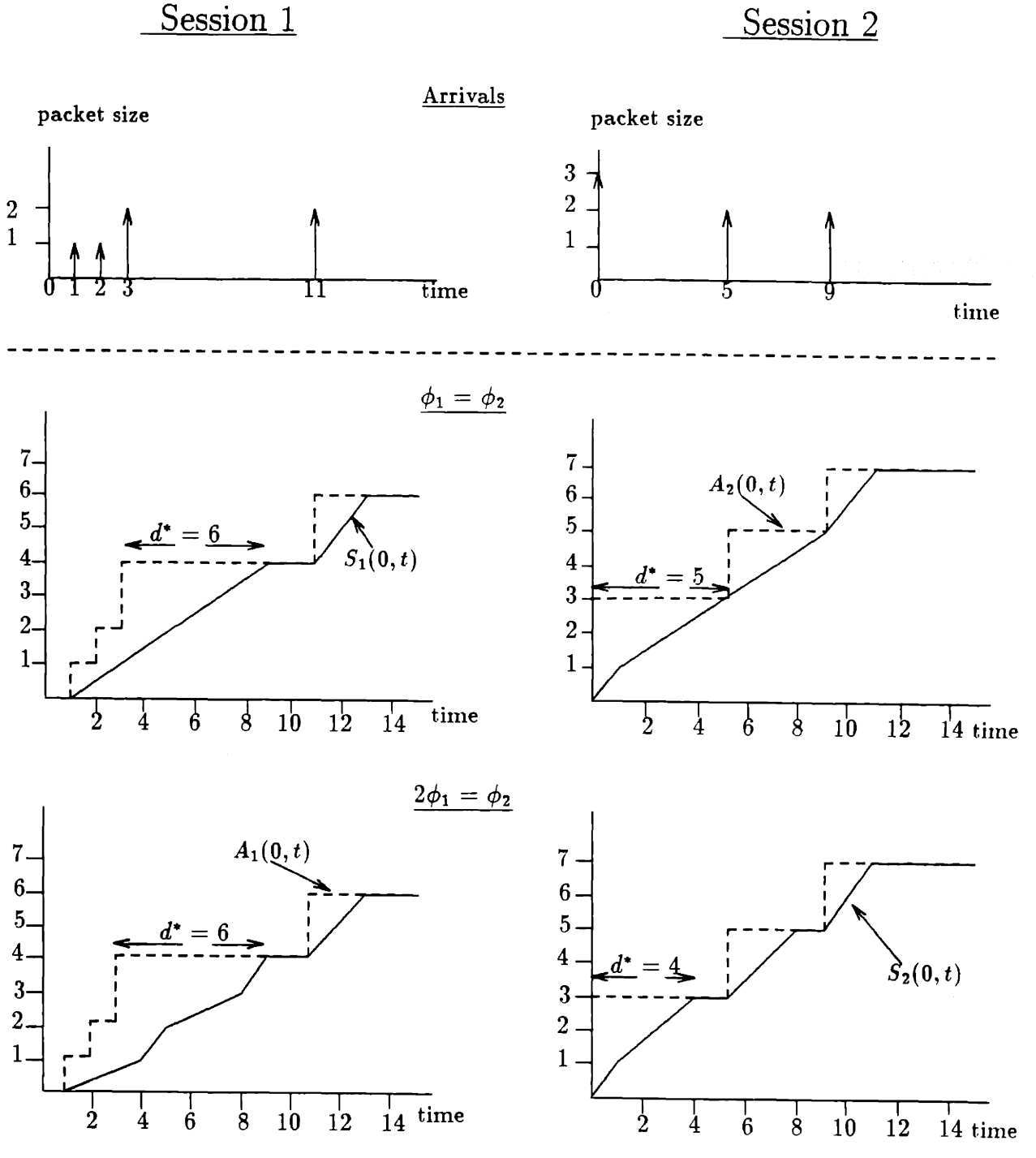
- Define ρ_i to be the session i average rate. Then as long as $\rho_i < g_i$, the session can be guaranteed a throughput of ρ_i, independent of the demands of the other sessions.

- 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. We will discuss this point more fully in Section 1.6

- 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.

- It is possible to make worst-case network queueing delay *guarantees* when the sources are constrained by leaky buckets. Most of Chapters 2 and 3 consist of an analysis of GPS that makes this possible. Thus GPS is particularly attractive for sessions sending real-time traffic such as voice and video.

Figure 1-1 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



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 1-1: An example of generalized processor sharing.

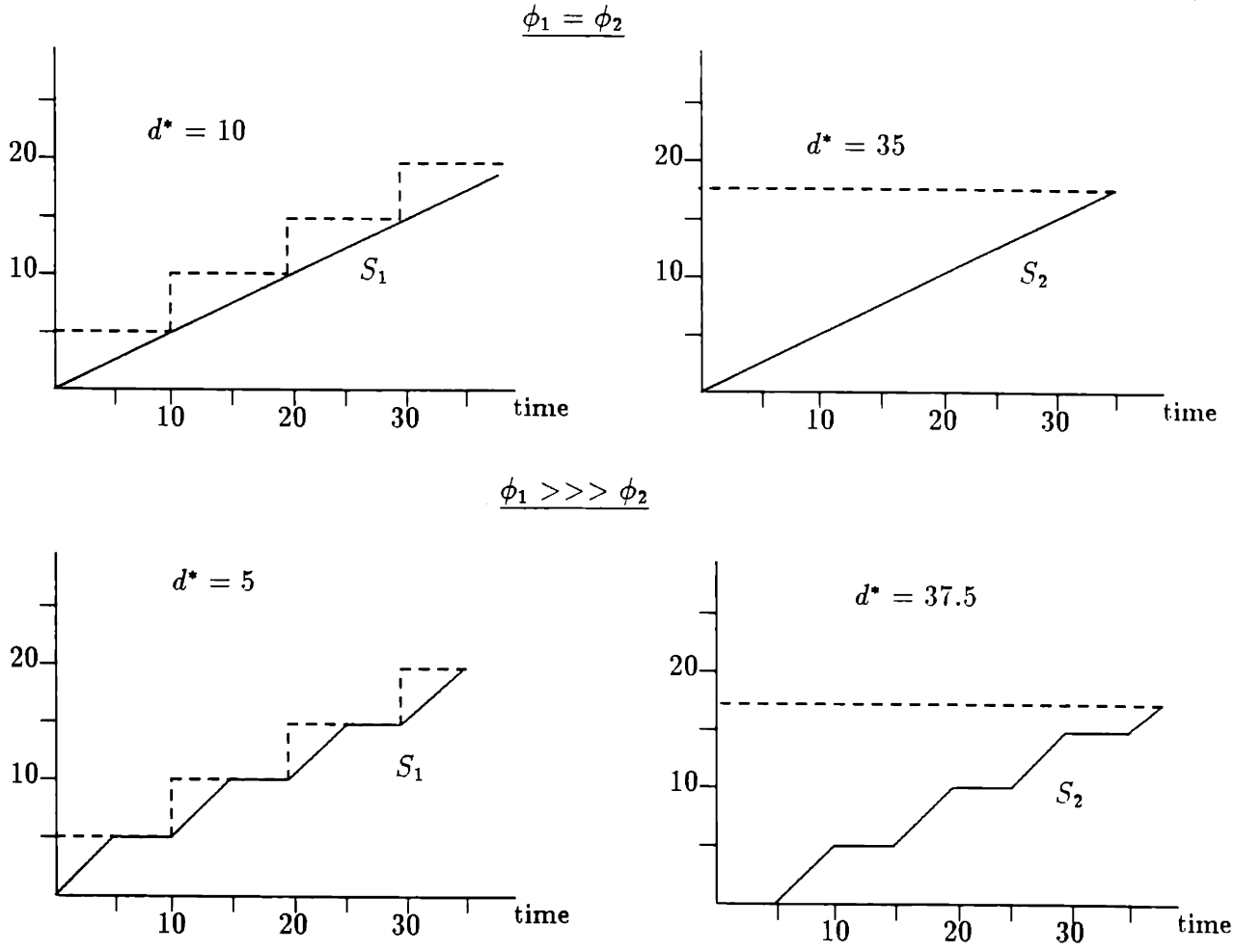
in the interval $(0,t]$, and similarly, let $S_i(0,t)$ be the amount of session i traffic that is served in the interval $(0,t]$. We assume that the server works at rate 1. Notice that the vertical distance between $A_i(0,\tau)$ and $S_i(0,\tau)$ for each i, is the backlog at time τ. To find the delay of a packet that arrives at t, measure the horizontal distance between the curves $A_i(0,\tau)$ and $S_i(0,\tau)$ at the ordinate value of $A_i(0,t)$—this is the amount of time that the packet must wait in the system before it departs. The maximum delay for packets arriving in the interval $[0,11]$ is marked (as d^*) on each of the four graphs in Figure 1-1.

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. 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. The point of Figure 1-2 is to show 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.

1.5 A Packet-by-Packet Transmission Scheme–PGPS

The 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 propose a simple packet-by-packet transmission scheme that is an excellent approximation to GPS even



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 1-2: The effect of increasing ϕ_i for a steady session i

		Session 1				Session 2		
packet	Arrival	1	2	3	11	0	5	9
information	Size	1	1	2	2	3	2	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 1-3: How GPS and PGPS compare for the example in Figure 1-1.

when the packets are of variable length. We have recently dicovered that our scheme is identical to the non-uniformly weighted version of Fair Queueing in [6]. *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 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.

Figure 1-3 shows how PGPS performs for the example in Figure 1-1. 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 1.1 *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 backlogged at time τ. By (1.1), 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 packet p cannot leave later than packet p' in the simulated GPS system.

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

Theorem 1.1 *For all packets p,*

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

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 given busy period and denote the time that it begins as 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

$$u_m > u_k \geq u_i \quad \text{for } m < i < k. \tag{1.3}$$

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 Lemma 1.1:

$$\min\{a_{m+1}, ..., a_k\} > t_m - \frac{L_m}{r} \tag{1.4}$$

Since $p_{m+1}, ..., p_{k-1}$ arrive after $t_m - \frac{L_m}{r}$, they receive all their service under GPS after time $t_m - \frac{L_m}{r}$. Similarly, they receive all their service before p_k departs at time u_k. Thus

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

implying that

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

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

$$u_k \geq t_k.$$

□

From the structure of the proof of Theorem 1.1 it should be clear that the only packets that are delayed more in PGPS are those that arrive too late to be transmitted in their natural order. Intuitively, this means that only the packets that have small delay under GPS are delayed more under PGPS.

Let $S_i(\tau,t)$ and $\hat{S}_i(\tau,t)$ be the amount of session i traffic served under GPS and PGPS in the interval $[\tau,t]$. Then we can use Theorem 1.1 to show:

Theorem 1.2 *For all times τ and sessions i.*

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

Proof. The slope of the second argument 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 lies within 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 1.1:

$$\tau \geq (t + \frac{L}{r}) - \frac{L_{\max}}{r}, \quad \text{implying that} \tag{1.5}$$

$$S_i(0, t + \frac{L - L_{\max}}{r}) \leq \hat{S}_i(0, t + \frac{L}{r}) \tag{1.6}$$

$$= \hat{S}_i(0,t) + L. \tag{1.7}$$

Since the slope of S_i is at most r, the Theorem follows. □

Let $\hat{Q}_i(\tau)$ and $Q_i(t)$ be the session i backlog at time τ under PGPS and GPS respectively. Then it immediately follows from Theorem 1.2 that

Corollary 1.1 *For all times τ and sessions i.*

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

Notice that

- We can use Theorem 1.1 and Corollary 1.1 to translate bounds on GPS worst-case packet delay and backlog to the corresponding bounds on PGPS.
- Variable packet lengths are easily handled by PGPS. This is not true of weighted round robin.

In Section 1.5.1 we will provide a way to implement PGPS using the concept of virtual time, that will allow us to simulate the evolution of the GPS process efficiently.

1.5.1 A Virtual Time Implementation of PGPS

In Section 1.5 we described PGPS but did not provide an efficient way to implement it. 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 is a generalization of the one considered in [6] for uniform processor sharing. In the following we assume that the server works at rate r.

Denote as an event 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 denote the virtual time that it begins as time zero. Then $V(t)$ evolves as follows:

$$V(0) = 0$$

$$V(t_{j-1}+\tau) = V(t_{j-1}) + \frac{\tau}{\sum_{i\in B_j}\phi_i}, \quad \tau \le t_j - t_{j-1}, \quad j = 2,3,... \tag{1.8}$$

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 $r\phi_i\frac{\partial V(t_j+\tau)}{\partial \tau}$. Thus, V can be interpreted as increasing proportionally to 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

$$\begin{aligned} S_i^k &= \max\{F_i^{k-1}, V(a_i^k)\} \\ F_i^k &= S_i^k + \frac{L_i^k}{\phi_i}. \end{aligned} \tag{1.9}$$

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 be 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 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, 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 (1.8), and the packet stamped with its virtual time finishing time. 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

(1.8) since the set of busy sessions 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 (1.8):

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

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

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.

1.6 Comparing PGPS to other service disciplines

We argued in Section 1.2 that a service discipline should be flexible, efficient and analyzable. In addition, given our focus on worst-case performance, the largest delay faced by a packet of a given session should not be much more than the average delay faced by the packets of the session. In this section we will examine the merits of PGPS relative to other service disciplines.

1.6.1 First Come First Serve and Strict Priority

Let N sessions share a server that operates at rate r. The sessions are not constrained by rate enforcement. Now suppose a session i packet p, arrives at some time, τ. Under *FCFS* all of the backlogged packets found by packet p will be served before p. It is reasonable to serve backlogged session i packets before p but what of the backlogged packets from the *other* sessions? Notice that a burst of session $j(\neq i)$ arrivals just before time τ may force p to wait a very long time before it is served. Thus p may receive arbitrarily worse treatment than do the other session i packets that are lucky enough not to be held up by such a burst.

Strict Priority schemes suffer from the same problem, since p may have to wait arbitrarily long for higher priority traffic to be served.

For the schemes examined so far, it is not possible to make average rate guarantees for the packets of a particular session, unless the average incoming rate for every other session is restricted. More importantly, the worst-case delay faced by a packet may be arbitrarily high even when these *average rates are restricted.*

In marked contrast, PGPS has the following property, which we call worst-case packet fairness: Let $Q_i(\tau)$ be the session i backlog in bits at time τ:

Definition: *A service discipline is* worst-case packet fair *if for every session i, and time τ, the delay of a packet arriving at τ is bounded above by $g_i^{-1}Q_i(\tau) + c$, where g_i is the throughput guarantee to session i, and c is a constant independent of the queues of the other sessions sharing the multiplexer.*

Notice that for a worst-case packet fair service discipline, a session i packet arriving at time τ cannot be delayed by an arbitrarily large amount even if $Q_j(\tau^-) = \infty$ for some $j \neq i$. Clearly, FCFS and strict priority are not worst-case packet fair.

1.6.2 Round Robin

The drawbacks of FCFS and strict priority schemes are mitigated considerably by Round Robin service. Newly arriving packets queue up by session, so that there are N different queues. The server polls each session queue in cyclic order and serves a packet from any non-empty buffer encountered. Since the server differentiates among packets by session, this scheme may be called *session-based* round robin.

The major advantage of cyclical disciplines over FCFS and strict priority schemes is that no single session can dominate the attention of the server at the expense of the other sessions. Round robin is an attempt to treat all sessions equally, and to ensure each session a service rate of at least $\frac{1}{N}$. However, this results in a lack of *flexibility* which is essential if certain sessions are supposed to be treated better than others.

Suppose a session i packet, p, arrives at time τ and sees session i backlog of

$P_i(\tau)$ packets. (Assume fixed packet lengths here.) Then p will wait for no more than $P_i(\tau)$ rounds before being served, where a round of service may take between 0 and N packet transmission times, depending on the number of backlogged sessions encountered by the server in the round. While p's delay does depend on how active the other sessions are, no other session i packet that sees the same session i backlog of $P_i(\tau)$ can receive *arbitrarily* better treatment than p as long as N is bounded above. In general, the *variability* in delay experienced by the packets of session i is bounded as long as $P_i(t)$ and N are bounded. However, since the number of sessions sharing a link may fluctuate over a large range, session i packets that arrive when N is small may receive much better treatment than those that arrive when N is large.

Matters are complicated if one assumes *variable packet sizes* since sessions with small packet sizes may be unduly penalized. To alleviate this problem one may stipulate that no more than Δ units of a session's traffic are served each time the session is polled. As $\Delta \rightarrow 0$ the system reduces to uniform processor sharing.

The fact that the server moves in a fixed polling cycle makes the scheme difficult to analyze. Session based systems have been studied extensively in the queueing theory literature, but most of the analysis assumes that all sources have identical traffic statistics–this is known as a symmetric system. Session based round robin systems are a special case (called limited service systems in the queueing theory literature) of polling systems, which are covered comprehensively by Takagi [21]. He mentions in [21] that "it appears unlikely that exact solutions will be found for asymmetric systems of N (> 2) queues." This is a discouraging prediction, given the importance and wide applicability of round robin.

The difficulty in analyzing session based systems is compounded by the fact that they are not likely to be quasi-reversible. When a queue *is* quasi-reversible, its steady state distribution can be established exactly, without resorting to complicated, and mostly uninsightful manipulations in the transform domain. In what follows, we define quasi-reversibility and show why it does not hold for session-based round robin:

Definition [16] *Suppose that associated with each customer is a class c chosen from a countable set $\mathcal{C}$. Then a queue is quasi-reversible if its state $\mathbf{x}(t)$ is a stationary Markov Process with the property that the state of the queue at time t_0, $\mathbf{x}(t_0)$ is independent of*

(i) the arrival times of class c customers, for each $c \in \mathcal{C}$, subsequent to time t_0;

(ii) the departure times of class c customers, for each $c \in \mathcal{C}$ prior to time t_0;

Conjecture 1.1 *Session based round robin is not quasi-reversible.*

The motivation for this conjecture comes from Theorem 3.6 of Kelly's highly readable book [16]. If one makes (the necessary) assumption that the class of a packet corresponds to its session then Kelly's result can be stated as follows:

Theorem 1.3 *If a session-based queue with N sessions is quasi-reversible then:*

(i) the arrival times of session i packets for $i = 1, 2, ..., N$ form N independent Poisson processes;

(ii) the departure times of session i packets for $i = 1, 2, ..., N$ form N independent Poisson processes.

Although we have not been able to prove it, we consider it unlikely that the session departure processesunder GPS are independent processes (even when the arrival processes are). The issue of how one could analyze GPS stochastically is discussed further in Section 2.8, where we set up the state transition diagram for the case of exponentially distributed packet lengths and $N = 2$.

A more tractable form of cyclical service is based on queues differentiated by *messages* rather than by sessions. Each arriving packet forms its own queue and the server moves in cyclic order over this queue, serving up to Δ units from each queue polled. When the messages arrive according to a Poisson process, both message

based round robin and uniform processor sharing can be shown to be Quasi-Reversible queues, which makes them amenable to elegant analysis.

Yates [24] has analyzed queueing networks of message based round robin servers, and has found a steady state product form distribution for the number of customers in each server of the network. His results also hold for processor sharing in the limit, as packet lengths go to zero. Telatar [22] has examined message based processor sharing from an interesting angle. The server does not follow a cyclic order, but chooses a new message to serve at random. This message is then given Δ units of service unless the message has been completely served. The queue is shown to be quasi-reversible, and average delay and backlog found. When $\Delta \rightarrow 0$, this system reduces to (messaged-based) uniform processor sharing. An interesting consequence of this work is that processor sharing is shown to be a more general service discipline than round robin with infinitesimal packet sizes. Unfortunately, message-based service disciplines are not suitable for our purposes.

1.6.3 Weighted Round Robin

The round robin schemes just discussed are not flexible since they attempt to treat each session/message equally. For integrated services networks, a *weighted* round robin scheme seems more appropriate. Consider a round robin system in which every session i, has an integer weight w_i associated with it. When the server polls session i it will serve up to w_i packets before moving to the next session in the round. This scheme is flexible since one can assign the weights corresponding to the amount of service required. In particular, the scheme attempts to give session i a rate of

$$g_i = \frac{w_i}{\sum_j w_j}. \tag{1.10}$$

However, there is a tradeoff between flexibility and packet delay, since a large range of weights leads to large service cycle times, and consequently to large worst case

packet delays.

A much better implementation of weighted round robin operates according to a frame structure. The server attempts to enforce the weights over several (short) polling rounds that comprise a frame. The idea is best explained by example: Suppose there are two sessions A and B, and the weights are $w_A = 3$ and $w_B = 7$. Then the frame has 10 slots and has the structure: $ABBABBABBB$[3]. So there are four rounds in this frame. Sometimes the frame length can be made much less than than the sum of the weights. For example if $w_A = 20$ and $w_B = 60$, the frame is $ABBB$, i.e. it has only size four. In general, if the weights are reduced so that they do not have a common factor greater than 1, the frame structure must be at least $\sum_i w_i$ long. It is easy to come up with examples for which the frame size goes up exponentially with the number of sessions. For example, if $w_1 = 1$ and $w_i = 2 * w_{i-1}$ for $i = 2, ..., N$, the the frame size grows as $O(2^N)$.

Another advantage of PGPS is that it handles variable length packets in a much more systematic manner.

Next, we examine the issue of how well weighted robin approximates GPS for the simple case: $w_i = w$ for all i. Thus this is an example of uniform round robin, and the frame structure for the given weights is simply $1, 2, 3, ..., N$. Assume that the number of sessions, N, is much larger than 3, let the first frame begin at time zero, i.e., the server first polls session 1 at time zero, and assume that it takes one unit of time to serve a packet.

We focus our attention on a session N packet, p, that arrives at time $N - 1^+$, i.e. the instant of time just *after* time $N - 1$. All other sessions, $i = 1, 2, ..., N - 1$, have arrivals at times $i - 1$ and $N + i - 2$. Notice that under weighted round robin, p arrives just after the server has polled session N and so misses its slot. It departs the system at time $2N - 1$.

Now suppose that the server were a GPS server. In the interval $(N - 1, N)$ there

[3]Assume fixed length packets here.

are only two backlogged sessions, so that half of packet p will be served by time N. Similarly, an additional one third of the packet will be served by time $N+1$, and p will depart at time $N+\frac{5}{3}$. Thus, p spends about $N-3$ additional units of time in the system under round robin. For large N this can be a significant amount of time. Recall that under PGPS p can depart at most 1 unit of time later than it does under GPS.

The example illustrates that PGPS approximates GPS (in the worst case) much better than does weighted round robin. Stated differently, if the goal of weighted round robin is to satisfy (1.10), then PGPS meets this goal better than weighted round robin does in the worst case. However, when the packet sizes are small, both schemes will approximate GPS well.

Choosing a good frame structure

This minor digression addresses the question of how a frame structure should be chosen when the goal of weighted robin is to satisfy (1.10). PGPS can be used to accomplish this as follows: The frame structure is given by the session order in which the first $\sum_j w_j$ packets would depart the system *under a PGPS system* given by $\phi_i = w_j$ when no empty queue is encountered by the server; if two packets depart simultaneously, give higher priority to the packet belonging to the session of higher weight. For example, if $N = 4$ then the frame structure is

$$(4,3),(4,2),(4,3),(4,3,2,1).$$

Notice that the maximum distance between two successive session i slots is about $\frac{\sum_j w_j}{w_i}$.

1.6.4 A Least Slackness Implementation of PGPS

Many schemes proposed in the literature such as [7], and [19], explicitly manage the packet delays at each switch in the route. Under such schemes the service policy at a node is based on the deadline of each packet (rather than on the session type), where the deadline of a delay constrained packet is the time by which the packet must reach its destination. While this approach allows for considerable flexibility, it does not lend itself to the kind of analyzability afforded by schemes such as PGPS.

As a means for comparing PGPS with such deadline-based service disciplines, we present a "least-slack" implementation of PGPS, in which the due-dates are computed according to a slightly different notion of virtual time than the one used in Section 1.5.1. Every session has a virtual clock that reads time $VC_i(t)$ at real time t. Assume that the server works at rate 1, and suppose that a system busy period begins at time zero. Define $S_i^G(0,t)$ to be the amount of session i traffic served by the GPS server in the interval $[0,t]$. Then

$$VC_i(t) = \frac{S_i^G(0,t)}{\phi_i}.$$

Suppose a session i packet, p, arrives at time t. (Recall that we consider a packet to have arrived only after its last bit has arrived.) Define $A_i(0,t)$ to be the amount of session i traffic that has arrived in the interval $[0,t]$ under GPS. Then the packet is stamped with the value

$$s_p = \frac{A_i(0,t)}{\phi_i}.$$

At any time t, the slack of a backlogged session i packet p is defined as

$$slack_p(t) = s_p - VC_i(t).$$

Now suppose the server has just completed serving a packet at time τ. Then since it is work conserving, it must schedule the next packet transmission at time τ.

The rule is to serve a packet with the *least slack* at time τ, i.e. to schedule a

packet p such that $slack_p(\tau) \leq slack_q(\tau)$ for all packets q that are backlogged at time τ.

We now explain why this rule implements PGPS. Consider a session i packet, p that is backlogged in the PGPS system at some time t. Then $\phi_i slack_p(t)$ is the amount of session i traffic that the GPS server has to serve after time t, for packet p to clear the GPS system. Now consider a session j packet q, that is also backlogged at time t. It follows from the definition of GPS and from Lemma 1.1 that packet p clears the GPS system before packet q if and only if $slack_p(t) < slack_q(t)$. Note that this is true *regardless of any arrivals after time t*. Thus we have an implementation of PGPS in which the F_p's do not have to be computed explicitly.

Notice that

- In the context of traditional scheduling policies, s_p can be viewed as the due-date for packet p in virtual time. Hence the difference between the stamped value and virtual time is naturally referred to as slackness.

- The rate at which the virtual clock for session i ticks is proportional to the rate at which session i is being served by the GPS server. The faster a session's virtual clock ticks, the better service that session gets.

- The session i virtual clock tick rate is also proportional to ϕ_i. This ensures that sessions with large values of ϕ get better service than those with smaller values of ϕ.

- Suppose that all the sessions in a set B are backlogged in the interval $[0, T]$ under GPS. Then for any two sessions, $i, j \in B$, we have $VC_i(t) = VC_j(t)$ for all $t \in [0, T]$, i.e. their virtual clocks read the same time in the interval $[0, T]$.

- When all the ϕ_i's are equal, and the packet sizes are fixed, the rule simplifies to serving the packets in the order in which they appear at the heads of their sessions queues under GPS.

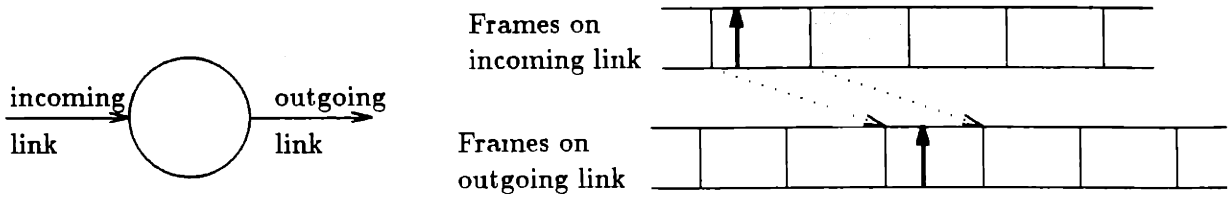


Figure 1-4: Adjacent Frames under Stop-and-Go.

1.6.5 Stop-and-Go Queueing

In [10, 11, 12] Golestani proposes a novel scheme for congestion management that is based on a service discipline called stop-and-go queueing. The scheme allows for tight control of jitter and provides bounds on end-to-end worst-case delay and backlog. In this section we will describe stop-and-go queueing, but postpone to Chapter 4, a detailed comparison of the network bounds with those obtained under PGPS with leaky bucket admission control.

The network supports a finite number of connection types. For every type g, each link divides time into frames of size T_g. The frame times do not have to be synchronized—frames may be defined with respect to different reference points.

The admission policy under which delay and buffer size guarantees can be made is that no more than $r_k T_g$ bits are admitted into the network in any type g frame, where r_k is the transmission rate assigned to session i. A traffic stream that obeys this restriction is said to be (r_k, T_g)-smooth.

The concept of *adjacent frames* is central to defining the stop-and-go discipline: Suppose a type g frame f, is defined on an incoming link (also called an *arriving* frame), and expires at time τ. Then the next type g frame on the outgoing link that begins after time τ is defined as the frame adjacent to f. This is illustrated in Figure 1-4. Any service discipline that follows the following three rules is defined to be stop-and-go:

(a) A packet arriving on a frame f is not eligible for service until the beginning of

the outgoing frame adjacent to f.

(b) $T_g \leq T_{g-1}$ for $g = 2, ..., G$ and any eligible type g packet has non-preemptive priority over eligible packets of type $g' < g$.

(c) The link is never idle if there are eligible packets in queue.

For appropriately constrained allocations of the r_k's, all of the traffic in each arriving (type g) frame can be served during the adjacent departing (type g) frame. Thus the smoothness condition imposed on the incoming traffic is maintained throughout the network. Also, two packets that enter the network in the same frame, will remain in the same frame for every hop traversed, and will leave the network in the same departing frame as well. The delay for each packet of a type g connection is given by $D_p^g - d_p^g$, where D_p^g is a constant for every packet of the connection, and d_p^g varies over the packets but has magnitude less than T_g. Thus, jitter is bounded by $2T_g$. Jitter is introduced because the position of a packet in the arriving frame of the destination node may be different from its position in the arriving frame at the source node.

This tight control over jitter is achieved at the cost of limited flexibility and reduced utilization of the network. The inefficiency results from the fact that bandwidth must be allocated by peak rates rather than average rates. Leaky bucket admission control is less restrictive than the access policy based on (r_k, T_g)-smoothness (this will be evident in Section 1.7). Further, the stop-and-go server is not work-conserving, and so average packet delays are also likely to be higher than they would be under PGPS (with the same kind of access control). Also, the need for fixed slot sizes limits the number of types of connections that are supported by the network. In Chapter 4 we will see how this can also limit the range of performance guarantees that can be given relative to PGPS.

Finally, we note that since stop-and-go queueing can be realized at a node through separate FIFO queues for each connection type, it is easy to implement. However, the virtual time implementation of PGPS does not appear to be difficult to realize

either [2].

1.6.6 Virtual Clock Multiplexing

In [25], Zhang proposes a "Flow" architecture for high speed networks in which both the rate enforcement and the multiplexing functions are performed using a mechanism known as a *Virtual Clock*. This virtual clock differs from the concept of Virtual Time introduced in Section 1.5.1 and from the virtual clock of Section 1.6.4.

Every session has a clock associated with it that ticks at a potentially different rate. When a session i packet arrives it is stamped according to an algorithm that is independent of the arrivals from the other sessions. The stamped packets enter a queue common to all the sessions, and are served in order of stamped value. The packets are stamped as follows: Let AR_i be the average *packet* arrival rate for session i.

- At session set-up time, set $VTick_i = AR_i^{-1}$.
- When the first session i packet is received, set $VirtualClock_i \leftarrow$ real time.
- Upon receiving a session i packet:
  1. $VirtualClock_i \leftarrow \max\{\text{real time}, VirtualClock_i\}$; stamp the packet with the value of $VirtualClock_i$.
  2. $VirtualClock_i \leftarrow VirtualClock_i + VTick_i$.

This scheme is similar to PGPS in that it does not allow bursty users to increase the delay of packets from other "better behaved" sessions, and is also able to treat sessions differently according to their average rates. It also provides good *average* packet delay. However, a significant difference between the two schemes has to do with how they deal with a session that has received better than guaranteed service during some interval of time. Under Virtual Clock multiplexing the packets of this

session are "punished" as illustrated in the following example, whereas under PGPS no such punishment occurs:

Suppose there are two sessions with $AR_1 = AR_2 = \frac{1}{2}$, and all packets require exactly one time unit of service. Starting at time zero, 1000 session 1 packets arrive at a rate of 1 packet/second. No session 2 packets arrive in the interval $[0, 900)$, so the first 900 packets from session 1 are served in the interval $[0, 900)$. At time 900^- there are no packets in queue from either session; the session 1 virtual clock reads 1800 and the session 2 virtual clock reads 900. Now suppose that 450 session 2 packets arrive starting at time 900 at a rate of 1 packet/second. They 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 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 no such punishment occurs which is essentially what permits rate guarantees to be seen by *every* arriving packet even in the absence of rate admission control. Since the admission of packets is regulated at the network periphery, it does not seem necessary to punish users at the internal nodes as well. Note that virtual clock multiplexing is not worst-case packet fair, which in the absence of stringent access control can result in poor worst-case packet delays.

1.7 Leaky Bucket

Consider the leaky bucket scheme [23] of Figure 1-5. 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

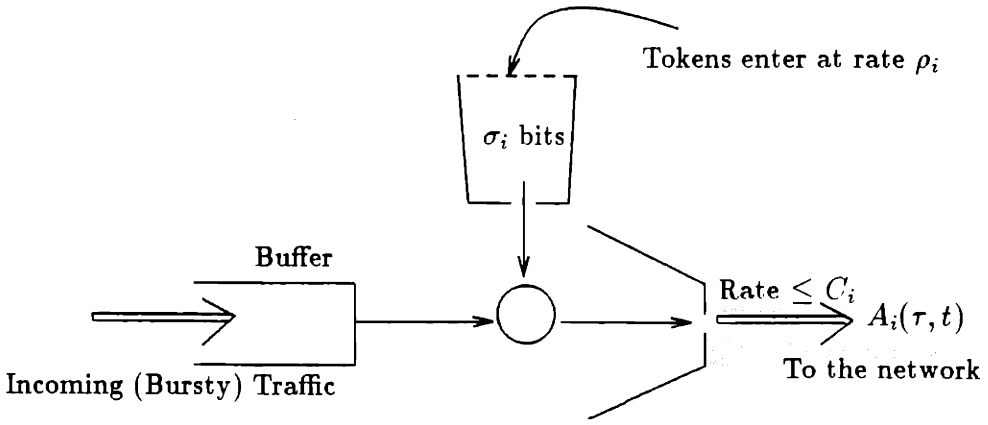


Figure 1-5: A Leaky Bucket

further constrained to leave the bucket at a maximum rate of $C \geq \rho$.

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

$$A_i(\tau, t) \leq \min\{(t-\tau)C_i, \sigma_i + \rho_i(t-\tau)\}, \quad \forall t \geq \tau \geq 0, \tag{1.11}$$

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 very similar to the one recently proposed by Cruz [4], [5], and it has also been used in various forms to represent the inflow of parts into manufacturing systems by Kumar [20], [18]. The arrival constraint is attractive since it restricts the traffic in terms of average rate (ρ), peak rate (C), and burstiness (σ and C).

1.7.1 Properties of the Arrival Constraint

The first concern that the reader might have is that the constraint appears to be so simple that it might not be descriptive enough to model a wide variety of traffic patterns. However, there is more to (1.11) than meets the eye. Figure 1-6 shows how a fairly bursty source might be characterized using the constraints.

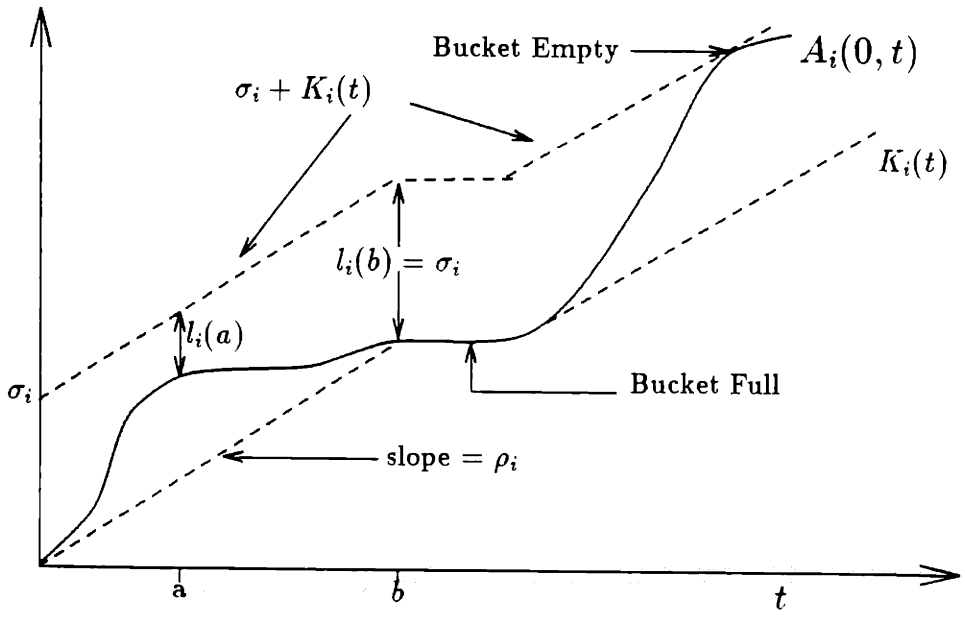


Figure 1-6: $A_i(t)$ and $l_i(t)$.

Suppose we are given a candidate session i arrival function, A_i, and are asked to check whether it conforms to (1.11). Let us assume that the slope of $A_i(0,t)$ never exceeds C_i, since otherwise (1.11) is violated. But since the condition (1.11) applies to *all* subintervals $(\tau, t]$, it is not clear if there is a convenient way of checking whether the burstiness conditions are met:

Represent $A_i(0,t)$ as in Figure 1-6. We assume that the session starts out with a full bucket of tokens. Let $l_i^A(t)$ be defined as the minimum difference between the two sides of (1.11) for $C_i = \infty$. Then

$$l_i^A(t) = \min_{\tau \le t}\{\sigma_i + \rho_i(t-\tau) - A(\tau,t)\}, \quad t \ge 0. \tag{1.12}$$

We can interpret $l_i^A(t)$ to be the number of permits left in the bucket at time t[4] if the token are arriving at rate ρ_i. Then from (1.11) we see that $\hat{A} \sim (\sigma_i, \rho_i, \infty)$ if and only if $l_i^A(t) \le \sigma_i$ for all $t \ge 0$.

These concepts are illustrated in Figure 1-6. $K_i(t)$ represents the total number of

[4]By "number" of permits we mean the number of bits that the session has permission to send.

tokens that will arrive at the session i bucket in the interval $[0, t]$. Thus

$$K_i(t) - K_i(\tau) \leq \rho_i(t - \tau), \quad \tau \leq t \tag{1.13}$$

(we do not have equality since the bucket may be full and reject tokens in some subintervals of $[\tau, t]$. Also, we now have a simpler definition for $l_i^A(t)$:

$$l_i^A(t) = \sigma_i + K_i(t) - A_i(0, t). \tag{1.14}$$

A result that will be useful later is the following:

Lemma 1.2 *For every session i, and session i arrival function A_i:*

$$l_i^A(\tau) - l_i^A(t) \geq A_i(\tau, t) - \rho_i(t - \tau) \tag{1.15}$$

for all $0 \leq \tau \leq t$.

Proof. From (1.14):

$$l_i^A(\tau) - l_i^A(t) = A_i(\tau, t) - (K_i(t) - K_i(\tau)).$$

But $K_i(t) - K_i(\tau) \leq \rho_i(t - \tau)$, so we are done. □

1.8 Delay Models for High Speed Networks

We have seen why the bandwidth assignment function of a rate based flow control scheme must compute (or reliably estimate) end-to-end packet delays in order to provide guarantees on these delays. Since the thrust of this thesis is to analyze such a scheme, a good model of delay in high speed data networks is indispensable to us.

Much work has been done in the past to formulate delay models for data networks. For a clear and insightful description of this work see Chapter 3 of [1]. The network

is modeled as a collection of interconnected queues, and average case analysis is performed for a variety of service disciplines. Packets entering the network are almost always modeled as a Poisson process and standard results in queueing theory are used to do the analysis. However, there are two reasons why the Poisson assumption is inappropriate for the packet arrival process in a high speed network:

First, even if messages are generated according to a Poisson process, in general they will be broken up or segmented into smaller packets. Unless we assume very long packet lengths, the long messages of high bandwidth applications such as video, will be broken up into *many* smaller packets, and the interarrival times of these packets is surely not exponentially distributed. In fact, assuming an exponential distribution is misleading, since it does not allow a realistic study of burstiness in the network. Second, recall that in rate based flow control schemes, the sessions are constrained in a deterministic fashion (for example in the leaky bucket schemes the amount of traffic that enters the network in Δ seconds is no more than $\sigma + \rho\Delta$). Even if the source generates Poisson traffic, the effects of these constraints would have to be taken into account. Thus, the main advantage of the Poisson process, namely its property of renewing at every instant of time may be lost as soon as the traffic enters the network.

Realistic stochastic models of the incoming traffic are very difficult to justify since the nature of the traffic generation is so varied. Recently proposed models are generally hierarchical or layered Markov models (for example [14]) that seem very difficult to analyze. Also note that the action of the leaky bucket precludes us from modeling the queueing problems as Jackson networks, which complicates the analysis further. Recall that in Jackson networks, the service time distributions are chosen independently at every server in the route—for a data network this implies that the packet lengths are rechosen at every node.

In this thesis, we will assume nothing about the traffic except that it is has been constrained in a known deterministic fashion by a rate enforcement function. This will preclude us from doing any average case analysis, which is a disadvantage, but

any of the results we do obtain will be valid for all types of source traffic. This is in keeping with our assumption that one cannot predict ways in which the network will be used in the future.

Another fundamental assumption upon which much of the delay analysis in data network is based (mostly implicitly) is that no single user can significantly alter the average rate at which traffic enters a subnet node. This assumption, known as the "Many Small Sessions Assumption," (Chapter 3, [1]) allows one to reasonably assume that the traffic entering a subnet node can be modeled as a stationary stochastic process (the Quasistatic assumption). While these assumptions have simplified data network performance analysis dramatically (especially of routing), we must be careful in applying them to high speed networks. For example, suppose that most of the sessions coming into a node are voice and low-speed data traffic sessions. The addition of a 50 Mb/s bursty "image" session might dramatically affect the traffic rate into the node. If such scenarios are probable (as we suspect they will be) in high speed networks, the "small sessions" assumption is seriously violated.

Motivated by the stringent delay requirements of real-time traffic, the approach we will take is to compute worst case bounds on quantities such as delay and the buffer size (at each node), rather than determining their distributions. Worst case analysis is sometimes dismissed since it can lead to overly conservative results. However, if performed carefully, it can yield considerable insight. Stochastic analysis of realistic systems, such as the session based generalized processing sharing system, can be virtually intractable. When results are forthcoming, they are often in the transform domain and yield minimal insight. Finally, since bandwidth is an abundant resource of a high speed network, it seems reasonable to err on the side of the user—i.e. to provide firm guarantees to the user by possibly wasting some of the plentiful bandwidth. Fortunately, we will find that even under this conservative approach, excellent performance guarantees may be within the Generalized Processor Sharing framework.

The first step in formulating a methodology for carrying out the task at hand has been taken by Cruz [4],[5] whose work has been most helpful to us.

1.9 Thesis Outline

The goal of this thesis is to understand the tradeoff between flexibility and performance guarantees in an integrated services network when there is contention for the links. In Chapter 1, we proposed analyzing this tradeoff in the context of a system that combines a packet service discipline based on Generalized Processor Sharing at then nodes of the network with Leaky bucket admission control. The rest of this thesis consists of this analysis and is organized as follows:

In Chapter 2 we focus on the single node GPS system and show how to compute bounds efficiently on the worst-case session delay and backlog. We then use the set of tools developed in our analysis to understand other service disciplines as well, when their sources are leaky bucket constrained.

In Chapter 3 we extend the results of the single node system to arbitrary topology networks of GPS servers. We show how to compute bounds on session delay and backlog for a very broad class of GPS assignments called Consistent Relative Session Treatment assignments. We are also able to analyze the session route as a whole, rather than to treat the delay (backlog) along a path as the sum of the worst-case delays (backlogs) at each node in the path. For some assignments, this yields bounds on worst-case session delay that are independent the number of switches in the path.

These results are then further extended in Chapter 4, to arbitrary topology networks of PGPS servers. As an intermediate step, we incorporate packet lengths into the analysis of GPS networks of Chapter 3. The PGPS results are interpreted in the context of Rate Proportional Processor Sharing networks, where the server allocations of each session i, are proportional to ρ_i. We show that for high speed RPPS

networks:

$$D_i^* \leq \frac{\sigma_i + (K-1)L_{\max}}{\rho_i}. \tag{1.16}$$

Chapter 5 contains a summary of the major contributions of this thesis, and also has some suggestions for further work.

Chapter 2

Single Server Generalized Processor Sharing

In this chapter we analyze the performance of single server queueing systems in the worst-case for sessions that operate under leaky bucket constraints i.e., the session traffic is constrained as in (1.11). The performance of the multiplexer is characterized by three parameters–worst case session delay, maximum queue length and maximum burstiness of the outgoing session traffic. While our focus will be on generalized processing sharing servers, the techniques developed in this analysis carry over to systems with other service disciplines such as First Come First Serve and Strict Priority.

In the next section we describe the single server system, and define the quantities of interest—worst case delay, backlog, and burstiness. In Section 2.2 useful definitions that can be used for analyzing any single server system are presented, and some simple results derived. Next, we introduce the notion of a *greedy* session, and state the major result of this chapter—that as long as traffic from any session can arrive at the rate of the server, delay, burstiness and backlog are maximized for every session in a GPS system when all the sessions are greedy from the start of a system busy period. This allows us to bound session delay and backlog for any single node GPS system. In Section 2.4 we prove the result for systems that have infinite capacity incoming links,

and deal with the finite capacity input links case in Section 2.5. Having understood how to analyze a given single GPS system in the worst case, we address the problem of meeting the worst case delay requirements of the sessions. An efficient algorithm is presented in Section 2.6 that enables a server to check if a new session i can meet its delay requirement for a given value of ϕ_i. In Section 2.7 we present some of our ideas on how the ϕ_i's should be assigned, but work still needs to be done in this area. The problem of how to analyze a GPS server stochastically when $N = 2$ is briefly examined in Section 2.8. In Section 2.9 we exploit the insights gained from the GPS system to give simple derivations of delay for a variety of other multiplexers. Some of these results are extensions of the bounds found in [4]. We conclude the chapter with some suggestions for further work.

2.1 The Single Server System

In this section we assume a single server that is work conserving (i.e. it is never idle if there is work in the system), and that operates at the fixed rate of 1. 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 session i traffic that leaves the multiplexer is called the session i outgoing traffic, and is characterized by $(\sigma_i^{\text{out}}, \rho_i, C_i^{\text{out}})$. A major goal of this chapter is to determine the values of these parameters for GPS.

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 in t for all t (see Figure 2-1).

The session i backlog at time τ is defined to be

$$Q_i(\tau) = A_i(0, \tau) - S_i(0, \tau). \tag{2.1}$$

Notice that if $Q_i(\tau) = 0$, then $S_i(\tau, t) \leq A_i(\tau, t)$ for all $t \geq \tau$.

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

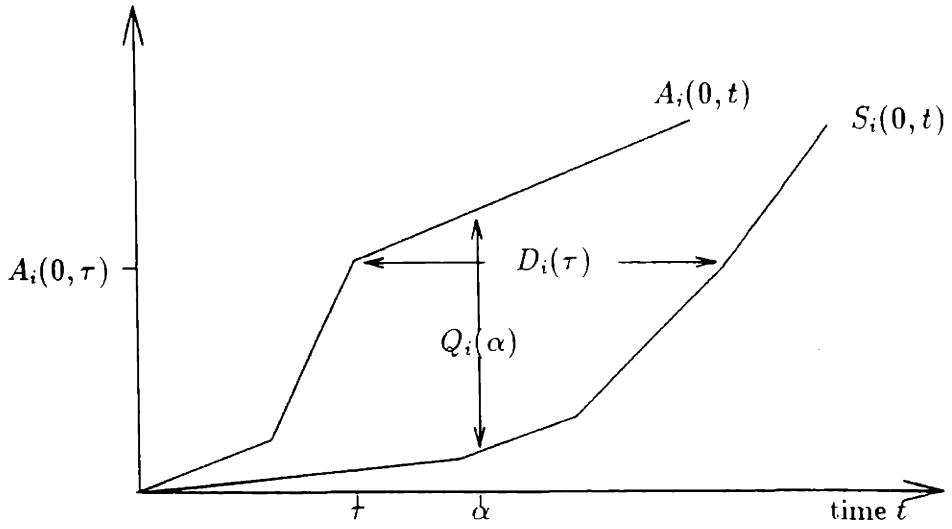


Figure 2-1: $A_i(0,t)$, $S_i(0,t)$, $Q_i(t)$ and $D_i(t)$

session i flow arriving at time τ spends in the system before departing. Thus

$$D_i(\tau) = \inf\{t \geq \tau \ : \ S_i(0,t) = A_i(0,\tau)\} \ - \ \tau. \tag{2.2}$$

From Figure 2-1 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 (1.11). Let D_i^* be the maximum delay for session i. Then

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

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

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

The problem we will analyze 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 following chapters.

2.2 Definitions and Preliminary Results

In this section we introduce notions that are helpful in analyzing GPS, but that are general enough to be used for other service disciplines as well. This will enable us to draw some general conclusions about single server work conserving systems.

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

$$\sigma_i^\tau = Q_i(\tau) + l_i(\tau) \tag{2.3}$$

where $l_i(\tau)$ was defined in 1.14, and is the number of tokens in the leaky bucket at time τ. 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 $\sigma_i^0 = \sigma_i$. Also, if $Q_i(\tau) = 0$ then $\sigma_i^\tau = l_i(\tau)$. But $l_i(\tau) \leq \sigma_i$ since the bucket size is σ_i. Thus

$$Q_i(\tau) = 0 \Rightarrow \sigma_i^\tau \leq \sigma_i. \tag{2.4}$$

Lemma 2.1 *For all time $t \geq \tau$,*

$$A_i(\tau, t) + Q_i(\tau) \leq \sigma_i^\tau + (t - \tau)\rho_i, \tag{2.5}$$

for all sessions, i.

Finally, we relate the amount of session i traffic served in the interval $[\tau, t]$ to σ_i^t and σ_i^τ:

Lemma 2.2 *For every session i, $\tau \leq t$:*

$$S_i(\tau, t) \leq \sigma_i^\tau - \sigma_i^t + \rho_i(t - \tau). \tag{2.6}$$

Proof. From Lemma 1.2:

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

Substituting for l_i^τ and l_i^t (from (2.3)) we have

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

Now notice that

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

Thus,

$$S_i(\tau, t) \leq \sigma_i^\tau - \sigma_i^t + \rho_i(t - \tau).$$

□

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 2.3 *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

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

From (1.11):

$$t_2 - t_1 \leq \sum_{i=1}^{N} (\sigma_i + \rho_i(t_2 - t_1)),$$

which yields

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

□

A simple consequence of this lemma is that every system busy period is bounded. Since session delay is bounded by the length of the largest possible system busy period, the session delays, and session queue lengths are bounded as well.

Finally, let σ_i^{out} be the least quantity such that $S_i \sim (\sigma_i^{\text{out}}, \rho_i, r)$ where r is the rate of the server. Then we may relate the maximum session i delay, D_i^*, to σ_i^{out} through a result by Cruz [4].

Lemma 2.4 *If $\sum_j \rho_j < 1$, then for every session i:*

$$\sigma_i^{\text{out}} \leq \sigma_i + \rho_i D_i^*.$$

Now consider Figure 2-2. It illustrates a single server system in which $\sum_i \rho_i < 1$. We can now draw the following conclusions for session i:

- Every system busy period is bounded. So are D_i^*, Q_i^* and σ_i^{out}.
- $\sigma_i + \rho_i D_i^* \geq \sigma_i^{\text{out}} \geq \sigma_i$: The first inequality follows from Lemma 2.4. For the

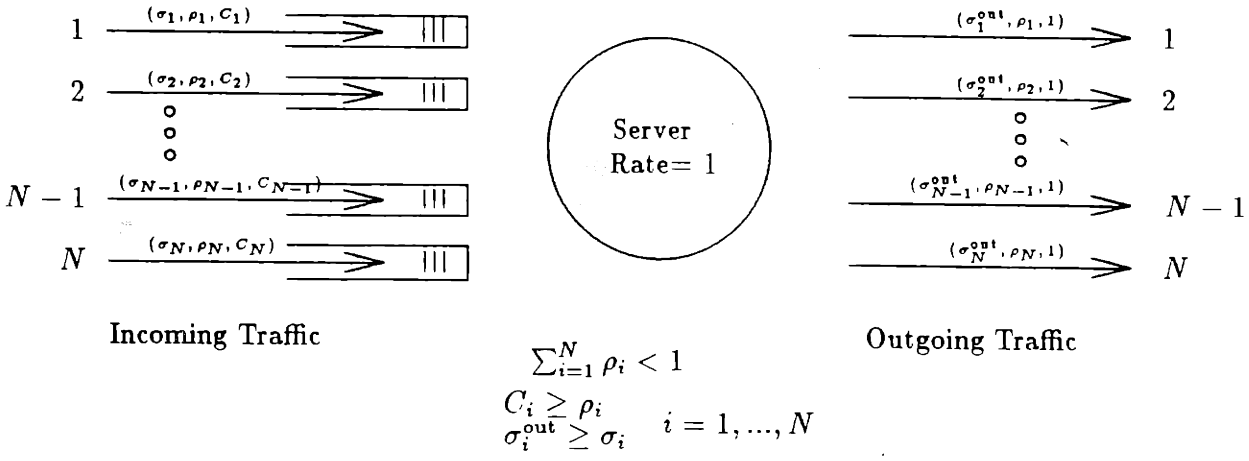


Figure 2-2: A single server system.

second inequality, suppose

$$A_i(0,\tau) = \min\{C_i\tau, \sigma_i + \rho_i\tau\}, \quad \tau \geq 0.$$

Then for large enough values of τ, i.e. for $\tau \geq \tau^*$ we have

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

Now since the system is stable, there must be a time $\tau \geq \tau^*$ at which the session i backlog is zero. Thus the amount served at this time must be $\sigma_i + \rho_i\tau$ implying that $\sigma_i^{\text{out}} \geq \sigma_i$.

- $C_i^{\text{out}} = 1$: The maximum rate at which it can send traffic on the output link for session i is the rate at which it operates, i.e. at rate 1.

Lemma 2.5 *If $C_i = \infty$ then for each session i:*

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

Proof. 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

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

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

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

This implies that

$$\sigma_i^{\text{out}} \leq \sigma_i^\tau - Q_i(t) \leq \sigma_i^\tau \leq Q_i^*,$$

where the last inequality holds since $C_i = \infty$. Thus

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

□

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 $\tau, t \in B_i$:

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j} \quad j = 1,2,..,N. \tag{2.7}$$

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 (1.2) that

Lemma 2.6 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

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

2.3 Greedy Sessions

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

$$A_i(\tau,t) = \min\{C_i(t-\tau), l_i(\tau) + (t-\tau)\rho_i\}, \quad \text{for all} \quad t \geq \tau. \tag{2.8}$$

To distinguish this arrival function from an arbitrary session i arrival function, we denote it as A_i^τ. Thus, for an arbitrary input A_i, the corresponding input A_i^τ that is greedy at time τ is A_i^τ such that $A_i^\tau 0, s = A_i(0,s)$ for $s < \tau$, and is given by (2.8) for times $t \geq \tau$.

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 $t \geq \tau$. 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

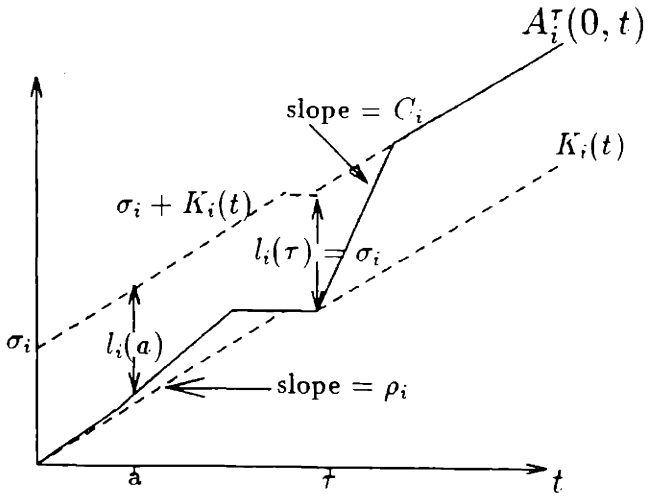


Figure 2-3: A session i arrival function that is greedy from time τ.

this, the rate will be limited by the token arrival rate, ρ_i.

Figure 2-3 depicts the arrival function A_i^τ which is greedy starting at time τ. From inspection of the figure (and from (2.8)), 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 2.1 *Suppose that $C_j \geq r$ for every session j, where r is the rate of the GPS server. Then for every session i, D_i^*, Q_i^* and σ_i^{out} 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 which this "all-greedy regime" does not maximize delay, the amount of work required to establish Theorem 2.1 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.

2.4 Generalized Processor Sharing with Infinite Incoming Link Capacities

When all the input link speeds are infinite, the arrival constraint (1.11) is modified to

$$A_i(t, t+\Delta) \leq \sigma_i + \rho_i \Delta, \quad \forall t \geq 0, \Delta \geq 0, \tag{2.9}$$

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 (2.9). When session i is greedy from time τ, equation (2.9) reduces to

$$A_i^\tau(\tau, t) = l_i(\tau) + (t-\tau)\rho_i, \quad \text{for all} \quad t > \tau. \tag{2.10}$$

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

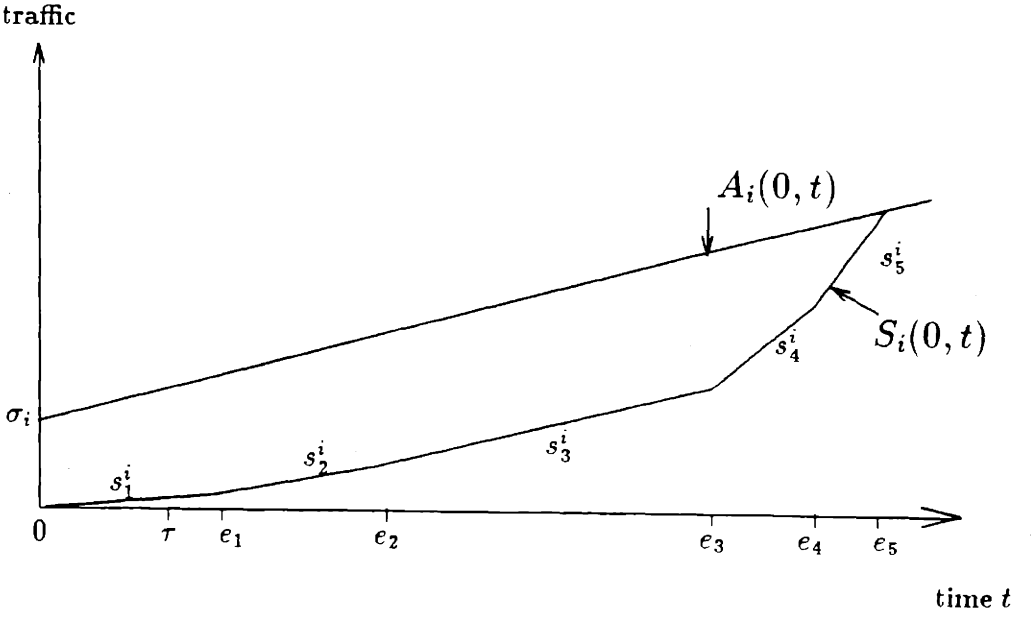


Figure 2-4: Session i arrivals and departures after 0, the beginning of a system busy period.

Defining σ_i^τ as before (from 2.3), we see that it is equal to $Q_i(\tau^+)$ when session i is greedy starting at time τ.

2.4.1 An All-greedy GPS system

Theorem 2.1 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 2-4.

From (2.10), we know that

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

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

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

(We will show that such a session must exist in Lemma 2.8.) 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:

$$\rho_{L(k)} < \frac{(1-\sum_{j=1}^{k-1}\rho_{L(j)})\phi_{L(k)}}{\sum_{j=1}^{N}\phi_j - \sum_{j=1}^{k-1}\phi_{L(j)}}, \quad k = 1,2,...,N. \tag{2.11}$$

As shown in Figure 2-4, the slopes of the various segments that comprise $S_i(0,t)$ are $s_1^i, s_2^i, ...$ From (2.11):

$$s_k^i = \frac{(1-\sum_{j=1}^{k-1}\rho_{L(j)})\phi_i}{\sum_{j=1}^{N}\phi_j - \sum_{j=1}^{k-1}\phi_{L(j)}}, \quad k = 1,2,...,L(i). \tag{2.12}$$

Lemma 2.7 *$\{s_k^i\}$ $k = 1,2,...,L(i)$ forms an increasing sequence for each session i.*

Proof. Without loss of generality, assume that $i = L(i)$ for $i = 1,2,...,N$. Then

$$s_k^i = \frac{(1-\sum_{j=1}^{k-1}\rho_j)\phi_i}{\sum_{j=k}^{N}\phi_j}, \quad k = 1,2,...,i. \tag{2.13}$$

Also,

$$\rho_k < s_k^k = \frac{\phi_k}{\phi_i}s_k^i. \tag{2.14}$$

We have for $k = 1,2,...,i-1$:

$$\frac{s_{k+1}^i}{\phi_i} = \frac{s_k^i}{\phi_i}\left(\frac{\sum_{j=k}^{N}\phi_j}{\sum_{j=k+1}^{N}\phi_j}\right) - \frac{\rho_k}{\sum_{j=k+1}^{N}\phi_j} \tag{2.15}$$

$$= \frac{s_k^i}{\phi_i}(1 + \frac{\phi_k}{\sum_{j=k+1}^N \phi_j}) - \frac{\rho_k}{\sum_{j=k+1}^N \phi_j}. \quad (2.16)$$

Thus

$$\frac{s_{k+1}^i - s_k^i}{\phi_i} = (\frac{s_k^i \phi_k}{\phi_i} - \rho_k)\frac{1}{\sum_{j=k+1}^N \phi_j} \quad (2.17)$$

$$> (\frac{s_k^i \phi_k}{\phi_i} - \frac{s_k^i \phi_k}{\phi_i})\frac{1}{\sum_{j=k+1}^N \phi_j} \quad (2.18)$$

$$> 0. \quad (2.19)$$

□

Note that

- We only require that

$$0 \leq e_1 \leq e_2 \leq ... \leq e_N,$$

  allowing for several e_i to be equal.

- We only care about $t \leq e_{L(i)}$ since the session i buffer is always empty after this time.

- Session i has exactly one busy period–the interval $[0, e_{L(i)}]$.

Any ordering of the sessions that meets (2.11) is known as a **feasible ordering**. Thus, sessions $1, ..., N$ follow a feasible ordering if and only if:

$$\rho_k < \frac{(1 - \sum_{j=1}^{k-1} \rho_j)\phi_k}{\sum_{j=k}^N \phi_j}, \quad k = 1, 2, ..., N. \quad (2.20)$$

Lemma 2.8 *At least one feasible ordering exists if* $\sum_{i=1}^N \rho_i < 1$.

Proof. By contradiction. Suppose there exists an index i, $1 \leq i \leq N$ such that we can label the first $i-1$ sessions of a feasible ordering $\{1, ..., i-1\}$, but (2.20), does not hold for any of the remaining sessions when $k = i$. Then we have for every session

$k \geq i$:

$$\rho_k \geq (1 - \sum_{j<i} \rho_j)(\frac{\phi_k}{\sum_{j \geq i} \phi_j}).$$

Summing over all such k we have:

$$\sum_{k \geq i} \rho_k \geq 1 - \sum_{j<i} \rho_j \Rightarrow \sum_{j=1}^{N} \rho_j \geq 1$$

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. □

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. More generally, 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:

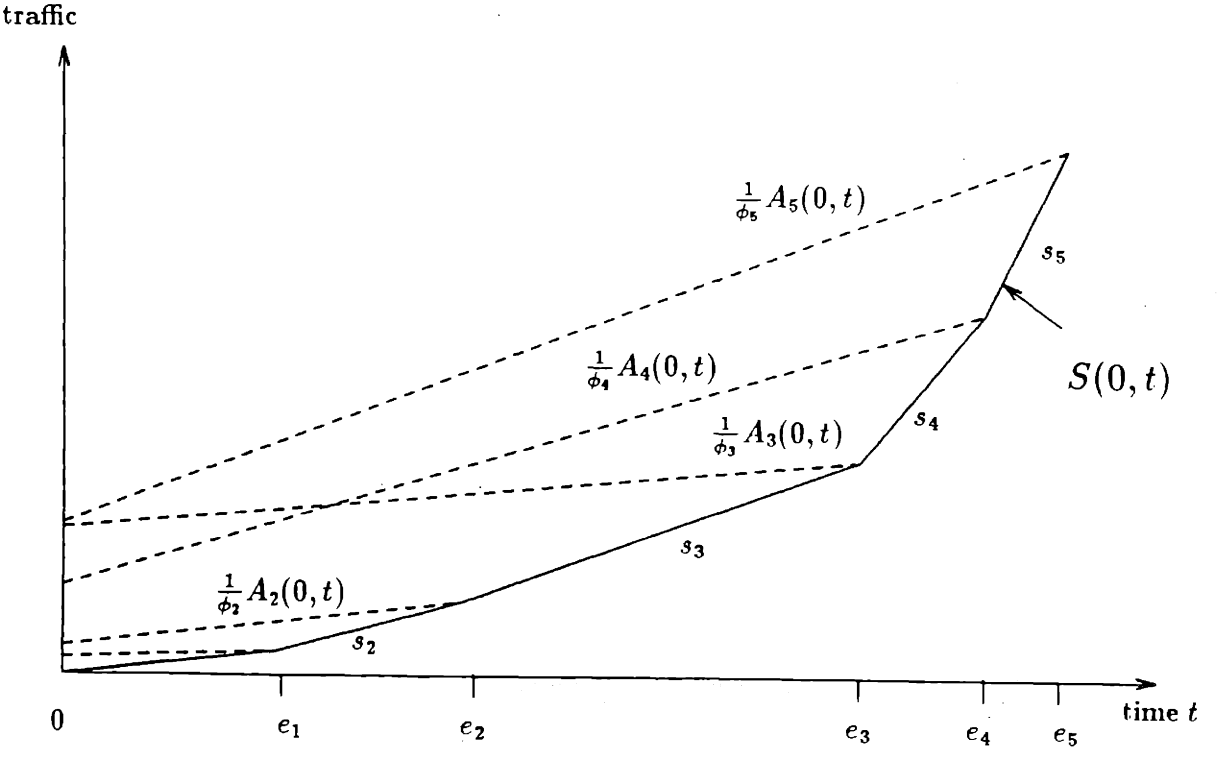
$$s_k = \frac{s_k^i}{\phi_i} = \frac{s_k^j}{\phi_i} = \frac{1 - \sum_{j=1}^{k-1} \rho_j}{\sum_{j=k}^{N} \phi_k}, \quad i, j > k, \; k = 1, 2, ..., N.$$

This allows us to describe the behavior of **all** the sessions in a single figure as is depicted in Figure 2-5. Under the all-greedy regime, the function $V(t)$ (described in Section 1.5.1), *corresponds exactly* to the universal service curve, $S(0,t)$, shown in Figure 2-5. 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 2.9

Assume that session i is in a busy period in the interval $[\tau, t]$. Then (i) For any subset



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 2-4.

Figure 2-5: The dynamics of an all-greedy GPS system.

M of m sessions, $1 \leq m \leq N$, and any time $t \geq \tau$:

$$S_i(\tau,t) \geq \frac{(t-\tau-(\sum_{j\notin M}\sigma_j^\tau+\rho_j(t-\tau)))\phi_i}{\sum_{j\in M}\phi_j}. \tag{2.21}$$

(ii) Under A^τ, there exists a subset of the sessions, M^t, for every $t \geq \tau$ such that equality holds in (2.21).

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

(i) From (2.6)

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

for all j.

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

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

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 (2.21).

(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 2-5, $Q_j(t) = 0$, 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 (2.21). □

2.4.2 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 (2.9).

Theorem 2.2 : *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 weaker result is quite easy to show

Lemma 2.10 *Suppose $B = [0, T]$ is a system busy period. Then*

$$\sum_{j=1}^{N} \sigma_j^t < \sum_{j=1}^{N} \sigma_j, \ 0 < t < T.$$

Proof. From Lemma (2.6) we have for every session j:

$$\sigma_j^t + S_j(0,t) \leq \sigma_j + \rho_j t. \tag{2.22}$$

Summing over $j \leq N$:

$$\sum_{j=1}^{N} \left(\sigma_j^t + S_j(0,t)\right) \leq \sum_{j=1}^{N} (\sigma_j + \rho_j t).$$

Since $[0,t]$ is in a system busy period, $\sum_j S_j(0,t) = t$. Thus

$$\sum_{j=1}^{N} \sigma_j^t + t \leq \sum_{j=1}^{N} (\sigma_j + \rho_j t) < \sum_{j} \sigma_j + t$$

$$\Rightarrow \sum_{j=1}^{N} \sigma_j^t < \sum_{j=1}^{N} \sigma_j.$$

□

The difference between Lemma 2.10 and Theorem 2.2 is that in the Lemma the sum is over *all* sessions, whereas in the theorem the sum is only over p sessions. It is surprising that the stronger result is *so* much harder to prove.

We present the following useful result in Lemma 2.11. It 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 2.11 *Let $1, ..., N$ be a feasible ordering, and suppose that session p is busy in the interval $[\tau, t]$. Further, define x to satisfy*

$$S_p(\tau, t) = \rho_p(t - \tau) - x \tag{2.23}$$

Then

$$\sum_{k=1}^{p-1} S_k(\tau, t) > \sum_{k=1}^{p-1}(t - \tau)\rho_k + x(1 + \sum_{j=p+1}^{N} \frac{\phi_j}{\phi_p}). \tag{2.24}$$

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

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

Thus

$$S_p(\tau, t) < (t - \tau)\left(\frac{1 - \sum_{j=1}^{p-1} \rho_i}{\sum_{i=p}^{N} \phi_{ip}}\right) - x. \tag{2.25}$$

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

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

Using (2.25)

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

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

$$\sum_{k=1}^{p} S_k(\tau, t) > (t - \tau) \sum_{k=1}^{p} \rho_k, \tag{2.26}$$

Proof. Let

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

$x \geq 0$. Then from (2.24) we have are done, since $x \sum_{j=p+1}^{N} \frac{\phi_j}{\phi_p} \geq 0$. □

Lemma 2.13 *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 2.2, for every k,

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

Summing over k, and substituting from (2.26), we have the result. □

If we choose τ to be the beginning of a session p busy period, then Lemma 2.13 says that if $S_p(\tau, t) \leq \rho_p(t - \tau)$ then

$$\sigma_p^t + \sum_{k=1}^{p-1} \sigma_k^t \leq \sigma_p + \sum_{k=1}^{p-1} \sigma_k^\tau. \tag{2.27}$$

Now we will prove Theorem 2.2: We state it again for reference:

Theorem 2.2: *Let* $1, ..., N$ *be a feasible ordering. Then for any time* t *and session* p:

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

Proof. 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 2.2:

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

This shows the basis.

Inductive Step: Assume the hypothesis for $1, 2, ..., p-1$ and show it for p. Let τ be the start of the session p busy period that contains t.

Observe that if $Q_i(t) = 0$ for any session i then $\sigma_i^t \leq \sigma_i$. Now consider two cases:

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

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

Thus

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

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 (2.6): $S_p(\tau, t) < \rho_p(t-\tau)$. Applying (2.27):

$$\sigma_p^t + \sum_{k=1}^{p-1} \sigma_k^t \leq \sigma_p + \sum_{k=1}^{p-1} \sigma_k^\tau \leq \sum_{k=1}^{p} \sigma_k, \tag{2.28}$$

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

2.4.3 Proof of the main result

In this section we will use Lemma 2.9 and Theorem 2.2 to prove Theorem 2.1 for infinite capacity incoming links. First we will show the result for delay and backlog, and then show it for outgoing session burstiness, σ_i^{out}.

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 2.14 *Suppose that time t is contained in a busy period that begins at time τ: Then*

$$\hat{S}_p(0, t-\tau) \leq S_p(\tau, t). \tag{2.29}$$

Proof. Define $\mathcal{B}$ as the the set of sessions that are busy at time $t-\tau$ under $\hat{A}$. From Lemma 2.9:

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

Since the order in which the sessions become inactive is a feasible ordering, Theorem 2.2 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 2.9), and (2.29) is shown. □

Lemma 2.15 *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 (2.9), 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 2.14:

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

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 (2.9), and suppose that for a session i busy period that begins at time τ:

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

From the definition of delay in equation (2.2):

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

Let us denote $d_i^* = t^* - \tau$. From Lemma 2.14:

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

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

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

Thus

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

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

□

From Lemma 2.5 we see that σ_i^{out} must also be achieved in an all-greedy regime. Thus we have shown Theorem 2.1 for infinite capacity incoming links.

2.5 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. We will find that while the system dynamics are complicated slightly, Theorem 2.1 still holds. The next section analyzes the case when all sessions are busy from time zero, which allows us to establish Theorem 2.1.

2.5.1 An All-greedy GPS system

Suppose all sessions are greedy starting at time zero, and for clarity of exposition assume that $\sigma_i > 0$ for all i. Then every session starts a busy period at time zero. Since the system busy period is finite we can label the sessions in the order that their first individual busy periods are terminated. Recall that for the infinite link capacity case this corresponds to labeling the sessions according to a particular feasible ordering.

Session i traffic will arrive at rate C_i for exactly

$$b_i = \frac{\sigma_i}{C_i - \rho_i},$$

after which it arrives at rate ρ_i.

Assume that $C_i \geq 1$ for every session i. Since session i traffic is arriving at least as fast as it is being served from time 0 to b_i, we must have $b_i \leq e_i$, where e_i is the time (≥ 0) when the session i busy period is terminated for the first time. 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 $\frac{\phi_i}{\sum_{k=1}^{N} \phi_k}$. At e_1, session 1 ends its busy period. Now since $e_1 \geq b_1$, session i must have been sending traffic at rate ρ_i at time e_1. Thus $\rho_1 < \frac{\phi_1}{\sum_{k=1}^{N} \phi_k}$. Similarly, we can show that

$$\rho_k < \frac{\phi_k(1 - \sum_{j=1}^{k-1} \rho_j)}{\sum_{j=k}^{N} \phi_j}, \quad k = 1, 2, ..., N. \tag{2.30}$$

Comparing this to (2.11) we see that the universal service curve $S(0,t)$ is identical to what it would have been if every session had infinite C_i! Since any arrival function $A_i \sim (\sigma_i, \rho_i, C_i)$ is also consistent with $(\sigma_i, \rho_i, \infty)$, arguments similar to those in Section 2.4 yield:

Lemma 2.16 *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.*

Since the traffic observed at the output of the system is indistinguishable from a system in which all the incoming links have infinite capacity, we see that maximum burstiness is also maximized when all the sessions are greedy from time zero. Thus we have shown Theorem 2.1.

Now let us consider the case in which the only constraint on C_j is that it is strictly

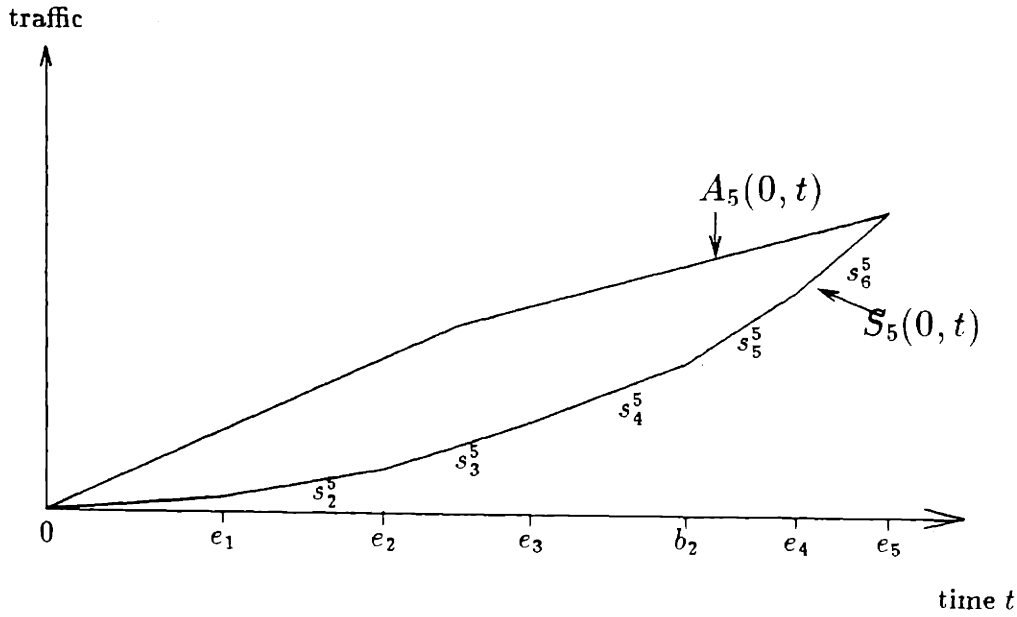


Figure 2-6: The arrivals and departures of session 5 when session 2 is deficient.

greater than ρ_j for each j. Then the system becomes more complicated under the all-greedy regime for two reasons:

1. A session j busy period that begins at zero may end *before* b_j. We call such as session a deficient session.

2. Under the all-greedy regime, the slope of $S_i(0,t)$ may be *greater* than the slope $A_i^0(0,t)$ for some time t.

The service curve under the all-greedy regime retains its convex-$\cup$ form despite the presence of deficient sessions. Consider a five session system in which only session 2 is deficient: Figure 2-6 depicts the session 5 arrivals and departures, and is the analog of Figure 2-4, which we explained in Section 2.4.1. Notice that $S_5(0,t)$ is convex, and is composed of six line segments. In the infinite capacity case we would have had no more than five such segments, i.e., one for every session. Here, the deficient session contributes two line segments: one that begins at e_2 and other that begins at b_2. Note

that in the example, $e_2 < e_3 < b_2$. Thus

$$s_3^5 = \frac{\phi_3(1-\rho_1-C_2)}{\sum_{i=3}^5 \phi_i},$$

$$s_4^5 = \frac{\phi_4(1-\rho_1-C_2-\rho_3)}{\sum_{i=4}^5 \phi_i}.$$

and

$$s_5^5 = \frac{\phi_4(1-\rho_1-\rho_2-\rho_3)}{\sum_{i=4}^5 \phi_i}.$$

The system dynamics can be seen more clearly in Figure 2-7.

The second complication mentioned above leads to more significant difficulties: Consider the two session example in Figure 2-8. It illustrates that the all-greedy regime does not minimize the session 2 service curve, although it does appear to minimize the session's worst-case delay and backlog. Thus the proof technique used in the infinite capacity case will not work for the case in which C_j can be less than the rate of server. While we will not prove that the all-greedy regime maximizes worst-case delay and backlog, some of the machinery required to deal with this case will be developed in Chapter 3. Further comments on this case are deferred to Chapter 5.

Note that Theorem 2.1 can be used to bound the worst-case session delay, backlog and burstiness for any GPS system, since a higher capacity link can always be used to simulate a lower capacity link. I.e.,

$$A_i \sim (\sigma_i, \rho_i, C_i) \Rightarrow A_i \sim (\sigma_i, \rho_i, \hat{C}_i)$$

for any $\hat{C}_i \geq C_i$.

2.6 Calculating D_i^*, Q_i^* and σ_i^{out}

The universal service curve shown in Figures 2-5 and 2-7 can be efficiently computed. Thus, in order to compute worst case performance measures we assume that all the

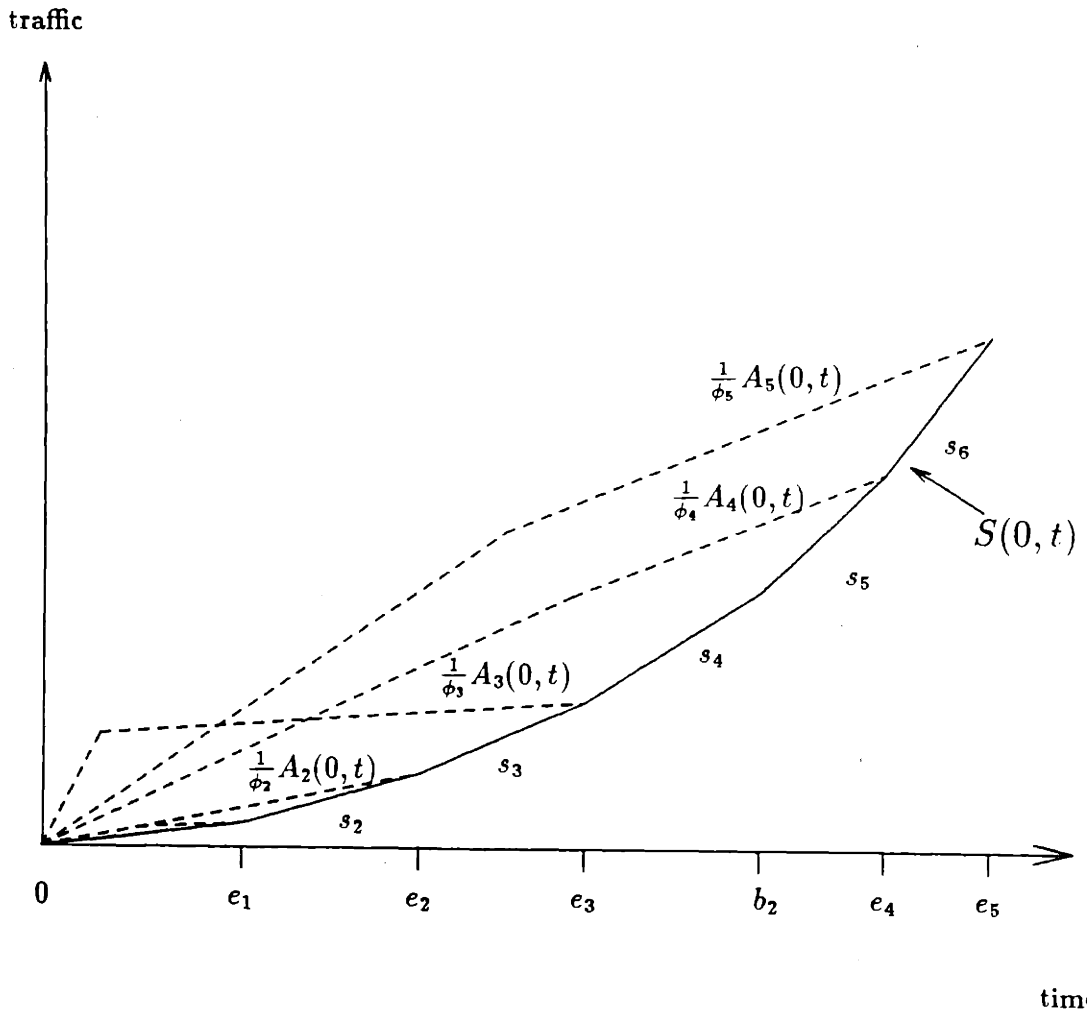
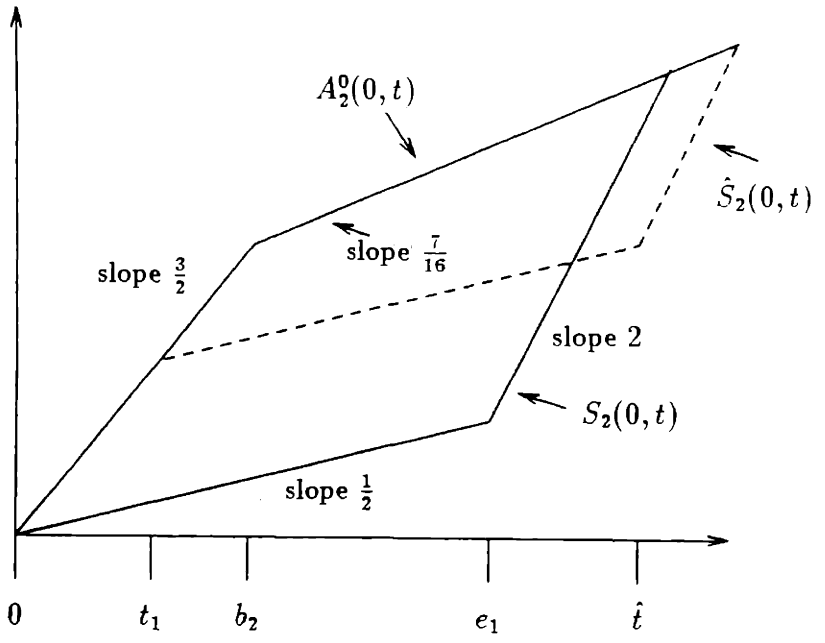


Figure 2-7: The dynamics of an all greedy system with finite link capacities: session 2 is deficient.



The server operates at a rate $r > 2$, and $C_2 = \frac{3}{2}$. The curve $S_2(0,t)$ represents the session 2 service curve under the all greedy regime, and $\hat{S}_2(0,t)$ represents the service curve when session 1 becomes greedy at time t_1.
Note that $S_2(0,\hat{t}) > \hat{S}_2(0,\hat{t})$.

Figure 2-8: A two-session example in which $C_2 < r$

sessions will be greedy starting at some time zero, and that they all have full buckets of tokens at this time. However, the all-greedy regime will be computed for the case $A_j \sim (\sigma_j, \rho_j, \hat{C}_j)$ for each session j, where

$$\hat{C}_j = \max\{r, C_j\}.$$

Since $\hat{C}_j \geq C_j$, the worst-case quantities computed through the all-greedy regime must upper bound their actual values:

Find the largest indexed slope j such that $s_j > \frac{\rho_i}{\phi_i}$ (where s_j is the j^{th} slope of the universal service curve). If there is no such slope set $j = 0$. Now define

$$t_i = max\{b_i, e_j\}.$$

We have :

$$Q_i^* = A_i^0(0, t_i) - \phi_i S(0, t_i)$$

and

$$\sigma_i^{\text{out}} = Q_i^*$$

from Lemma 2.5.

To compute D_i^*, we have to consider two cases:

Case 1: $\sigma_i \leq \phi_i S(0, t_i)$: Let t^* be such that

$$A_i^0(0, t^*) = \phi_i S(0, t_i).$$

Then

$$D_i^* = t_i - t^*.$$

Case 2: $\sigma_i > \phi_i S(0, t_i)$: Let e^* be the smallest time such that

$$S(0, e^*) = \frac{\sigma_i}{\phi_i}$$

and set

$$D_i^* = e^*.$$

In principle it is possible to give closed form expressions for these quantities once the order in which the sessions end their individual busy periods (i.e. the active feasible ordering) has been determined. However such expressions are messy and do not yield much insight.

2.7 Resource Assignment–Picking the ϕ's

In this section we address the following question: Suppose a new sessions wants to use a server that is currently being shared by sessions $\{1, ..., N\}$. Every session i is characterized by $\sigma_i, \rho_i, C_i, d_i$ where d_i is the worst case packet delay that can be tolerated by session i. For the active sessions we have $\phi_1, ..., \phi_N$ and the question is how to assign ϕ_{N+1} so that after including the new session every session i is guaranteed an average rate of ρ_i and worst case delay d_i.

The following is one possible approach that could be used.

1. If $\sum_{i=1}^{N+1} \rho_i \geq 1$ then reject session $N+1$. Otherwise proceed to step 2.

2. Find ϕ_{N+1}^{min}, the smallest value of ϕ_{N+1} that would ensure session $N+1$ a worst case delay of d_{N+1}.

3. For every session $i = 1, 2, ..., N$ find ϕ_{N+1}^i, the largest value of ϕ_{N+1} that would still ensure session i a worst case delay of d_i.

4. Compute $\phi_{N+1}^{max} = \min_{i=1,...,N} \phi_{N+1}^i$. If $\phi_{N+1}^{min} > \phi_{N+1}^{max}$ then reject session $N+1$. Otherwise

$$\phi_{N+1} = \frac{\phi_{N+1}^{max} - \phi_{N+1}^{min}}{2}.$$

Note that any choice of $\phi_{N+1} \in [\phi_{N+1}^{min}, \phi_{N+1}^{max}]$ will meet worst case delay guarantees. However, picking the extreme points is not advisable since otherwise no more sessions

could be accepted after session $N+1$. In step four we pick the midpoint of the interval, but it is not clear that this is always a good choice. More study needs to be done to determine the best rule.

2.8 Stochastic Analysis of GPS when $N = 2$

There are many applications of GPS for which a stochastic analysis would be more appropriate than the one based on leaky-bucket constrained sessions presented in this chapter. The applications we have in mind are for class differentiated customer-based service centers (such as the job-scheduler of a time sharing computer system) that have been analyzed in the Operations Research literature. This analysis is especially important since PGPS approximates GPS so well. We are not aware of any analysis in the literature on generalized processing sharing.

When the arrivals are described by a Poisson process and the customer requirements are distributed exponentially, a queueing analysis is not difficult conceptually, since the Markov Process has a fairly regular structure: The state of the queue at time t is $(c_1, ..., c_N)$ where c_i is the number of waiting session i customers at time t. The instant the server is free (and the system is not empty), it chooses to serve a session i packet with probability $\frac{\phi_i}{\sum_{j:c_j>0}\phi_j}$, and gives this packet up to Δ units of service. As $\Delta \to 0$ the service discipline reduces to session-based GPS. Let session i customers arrive according to a Poisson process with rate λ_i, and let the session i customer service requirement be exponentially distributed with parameter μ_i. Then the state transition diagram is given in Figure 2.8. In principle one can write down balance equations and solve for the steady state probabilities, although this is likely to be a very tedious exercise. It would be interesting to explore other, more elegant approaches to analyzing GPS stochastically.

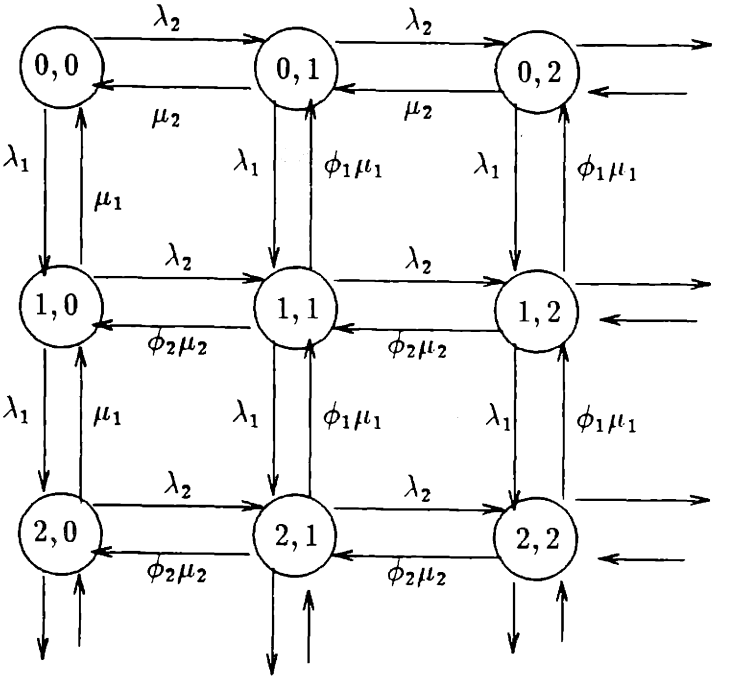


Figure 2-9: Transition state diagram for the GPS with $N = 2$, $\phi_1 + \phi_2 = 1$.

2.9 Other Service Disciplines

In this section we show how concepts introduced in our somewhat involved analysis of the GPS multiplexer can be used to provide simple derivations of delay for other kinds of multiplexers, when the sessions are leaky bucket constrained.

2.9.1 The FCFS Server

Under the first come first serve discipline, packets are served in the order in which they arrive, where a packet is said to have arrived when its last bit arrives. Multiple packets arriving at the same instant of time are served in arbitrary relative order. Assume variable packet lengths with a maximum allowable packet length of $L_{\max}$. The server works at rate 1, is work conserving, and can begin serving a packet before the entire packet has arrived if there are no other packets in queue. We also require for each i that $C_i \geq 1$. Cruz has derived a tight bound for D_i^* for the case $N = 2$ in

[4]. However, we have found that for large N this bound takes on a rather complicated form. In what follows we will derive a simpler, but slightly weaker bound for D_i^*. The bound is off by about $N-1$ packet transmission times for $L_{max} > 0$.

It is convenient to define the quantity b_i for each i:

$$b_i = \frac{\sigma_j}{C_i - \rho_i},$$

and to assume that $b_1 \leq b_2 \leq ... \leq b_N$.

Theorem 2.3 *When $b_N > 0$:*

$$D_i^* \leq \sum_{j=1}^{N-1} (\sigma_j + \rho_j b_N) + (N-1)L_{\max} \tag{2.31}$$

Proof. As an easy first case, let us bound delay for the case $L_{\max} = 0$. By definition of FCFS and since $L_{\max} = 0$, a bit arriving at time t is delayed by $\sum_{i=1}^{N} Q_i(t)$ time units. Since $\sum_{j=1}^{N} Q_j(t)$ is maximized when all sessions are greedy starting at time zero:

$$D_i(t) \leq \sum_{j=1}^{N} \min\{C_j t, \sigma_j + \rho_j t\ \} - t, \tag{2.32}$$

for each session i. The function $A^*(t) = \sum_{j=1}^{N} \min\{C_j t, \sigma_j + \rho_j t\ \}$ is piece-wise linear and convex-$\cap$ as shown in Figure 2-10. Since $C_i \geq 1$ for each j, the slopes of A^* are decreasing up to time b_N after which the slope is less than 1 (see Figure 2-10). Thus $D_i(t)$ is maximized at $t = b_N$, and setting $D_i(b_N) = D^*$:

$$\begin{aligned} D^* &= \sum_{j=1}^{N} \sigma_j + \rho_j b_N \ - b_N \\ &= \sum_{j=1}^{N-1} \sigma_j + \rho_j b_N \ + b_N - b_N \\ &= \sum_{j=1}^{N-1} \sigma_j + \rho_j b_N \ . \end{aligned} \tag{2.33}$$

Now suppose $L_{\max} > 0$, and that the first bit of a session i packet arrives at time

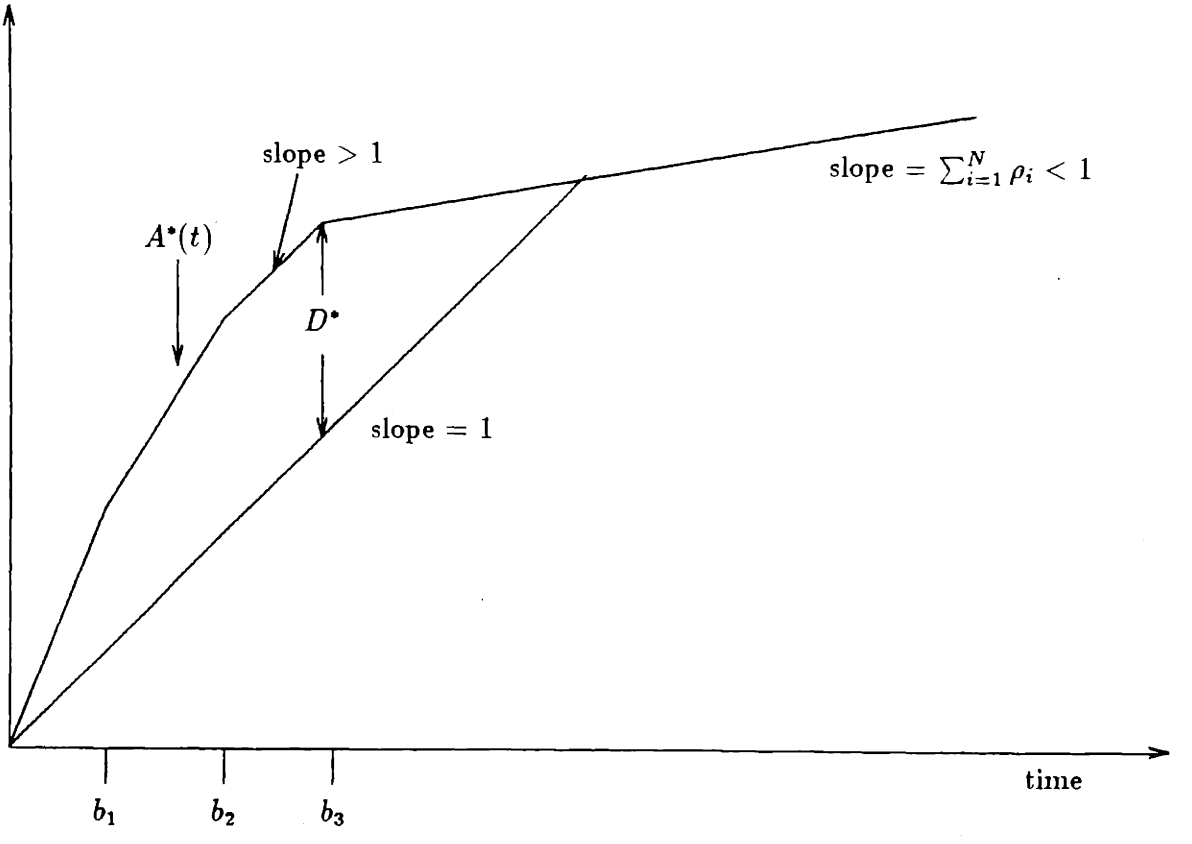


Figure 2-10: Maximum Delay under FCFS.

t. Then if the system backlog does not contain any partially arrived packets at this time, packet delay is bounded by the of Theorem 2.3. However, it is possible for one of the other $N-1$ session backlogs to contain a partially arrived packet at time t, and this packet will be scheduled before the session i packet. Thus, we have for the general case of $L_{\max} > 0$:

$$D_i^* \leq D^* + L_{\max} = \sum_{j=1}^{N-1} (\sigma_j + \rho_j b_N) + L_{\max}$$

for each session i. □

To obtain $S_1(0,t)$ for FCFS, note that within a server busy period:

$$S_1(0, t + D_1(t)) = A_1(0,t)$$

and

$$D_1(t) = \sum_{i=1}^{N} A_i(t) - t.$$

While D_i^* follows rather nicely from an all-greedy regime, the same is not true of σ_i^{out}. We will now show that σ_i^{out} is **not** maximized under an all-greedy regime. Consider the following simple two session example: $A_1 \sim (1, .5, 1)$, $A_2 \sim (6, 0, 1)$, and $L_{\max} = 0$: Figure 2-11 illustrates the all-greedy regime. Notice that the maximum session 1 backlog is 2. This backlog can be increased under the following regime: Session 2 is greedy from time 0, but session 1 is greedy *starting at time* 6 rather than at time zero. $A_1(0,t) = .5t$ for $t \leq 6$. Figure 2-12 shows that this results in a session 1 backlog of 2.5, i.e. .5 units greater than under the all-greedy regime. It turns out (see Theorem 4.3 of [4]) that for $N = 2$:

$$\sigma_1^{\text{out}} = \sigma_1 + \rho_1^2 b_2,$$

which is 2.5 for our example. Thus the regime of Figure 2-12 achieves maximum session 1 burstiness. It is also worth pointing out that if the server were GPS with

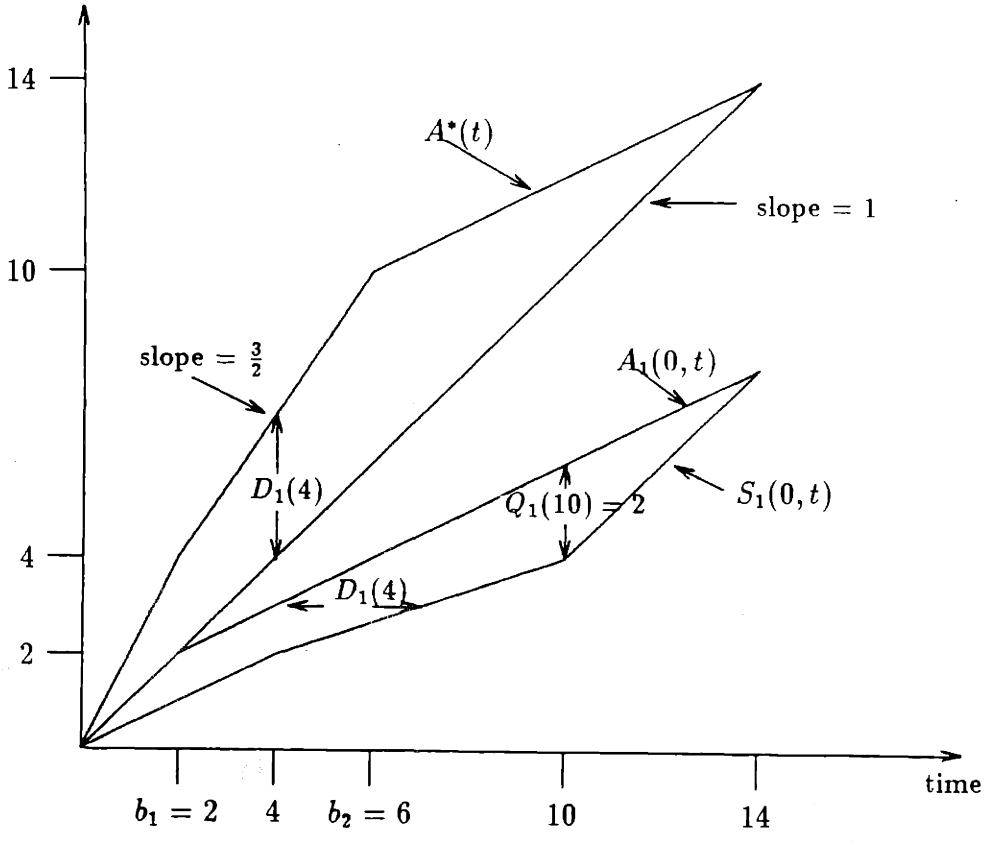


Figure 2-11: The all-greedy regime for FCFS for the two session example.

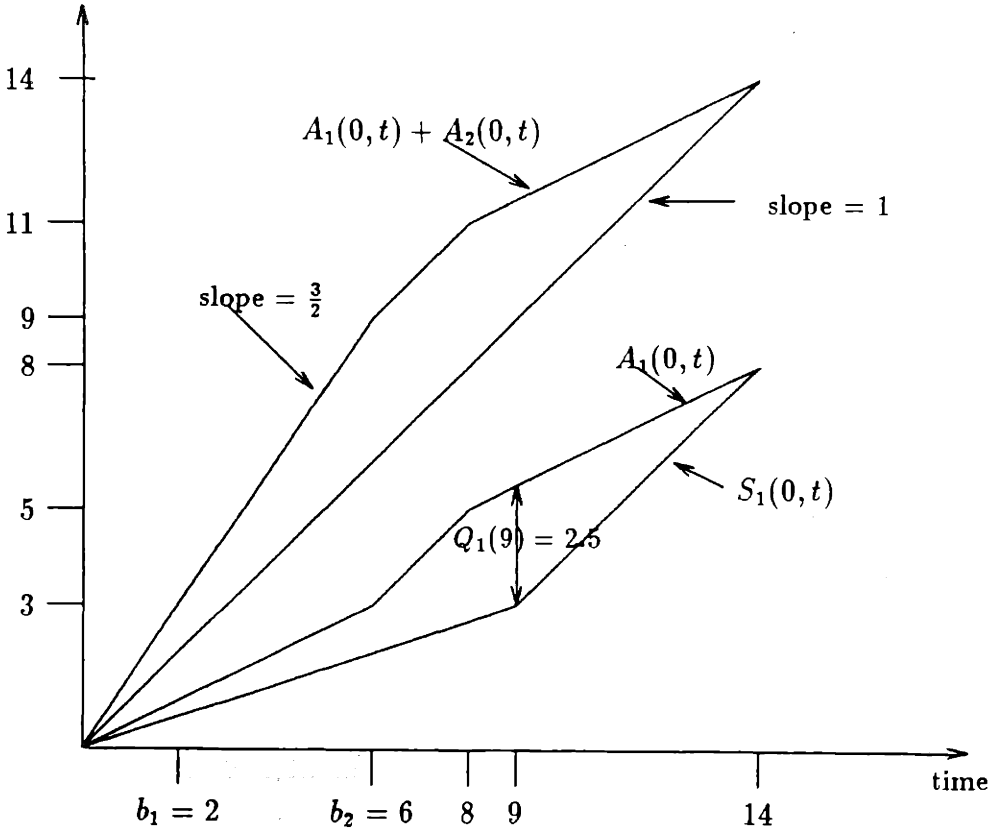


Figure 2-12: The regime maximizing session 1 burstiness for the two session example.

$\phi_1 = \phi_2$ then $\sigma_i^{\text{out}} = \sigma_i$ for each session i.

2.9.2 Strict Priority and Locally FCFS Service Disciplines

A locally FCFS service discipline is any work conserving multiplexer that operates under the constraint that the packets of a given session are served in FCFS order. Thus GPS and Round Robin are locally FCFS. In this section we rederive a special case of Theorem 4.2 in [4] by using the techniques developed in the GPS analysis. The bound in [4], while more general, is stated in terms of the maximum of a complicated set. Our result assumes that the packet lenghts are negligible, i.e., that $L_{\max} = 0$, but is of a somewhat more intuitive form.

First we state the following result that can be proven by contradiction:

Lemma 2.17 *If session i has the lowest priority in a strict priority service discipline, then D_i^* is no less than the maximum session i delay under any other locally FCFS service discipline.*

Proof. By contradiction. □

We now bound the worst-case session i delay under a strict priority scheme.

Lemma 2.18 *If session i has the lowest priority in a strict priority service discipline, then*

$$D_i^* = \frac{\sum_{j \neq i} \sigma_j}{1 - \sum_{j \neq i} \rho_j} + \frac{C_i b_i}{1 - \sum_{j \neq i} \rho_j} - b_i$$

for each session i, where $b_i = \frac{\sigma_i}{C_i - \rho_i}$.

Proof. Assume that all the sessions are greedy from time 0, the beginning of a system busy period. Then the first session i bit will be served when all the other session queues are empty. This will happen at time

$$T_i = \frac{\sum_{j \neq i} \sigma_j}{1 - \sum_{j \neq i} \rho_j}.$$

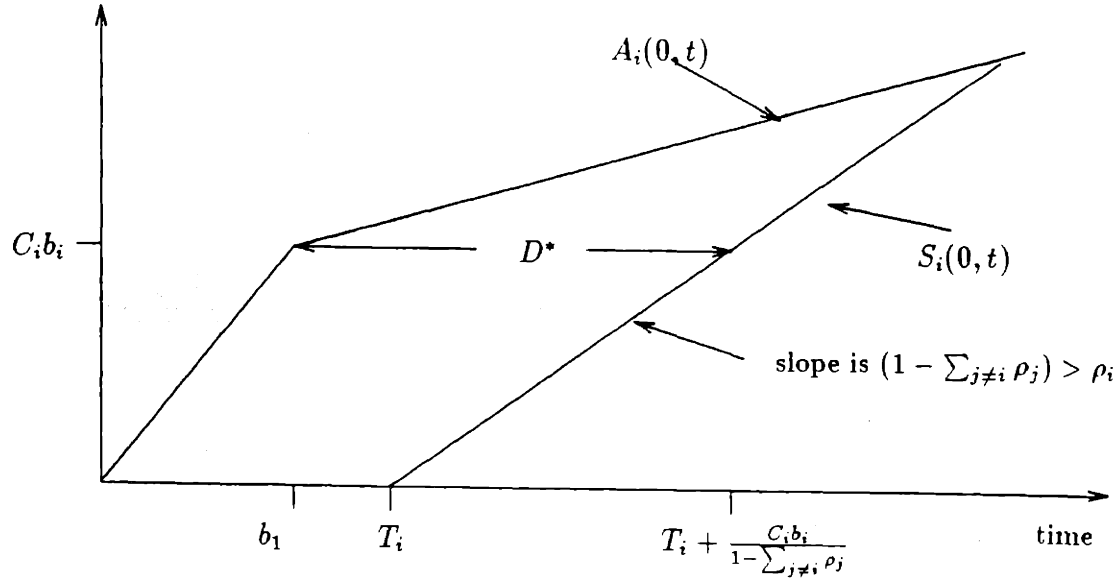


Figure 2-13: Maximum Delay over all Locally FCFS service disciplines

For $t > T_i$, the rate of service received by i is exactly $1 - \sum_{j\neq i}\rho_j > \rho_i$, until the session i queue is depleted. This is shown in Figure 2.9.2, which yields a maximum delay of:

$$D^* = T_i + \frac{C_ib_i}{1-\sum_{j\neq i}\rho_j} - b_i, \tag{2.34}$$

and which is the result we want to prove.

We now show that the all-greedy regime actually achieves *worst-case delay*: Consider an arbitrary regime of arrivals that achieves the worst-case delay for session i. Define τ to be the last time that the session i queue is empty before worst-case delay is achieved. Let $\hat{S}_i$ be the session i service curve for the regime under consideration.

It is easy to argue that D_i^* will be achieved for some bit arriving in the interval $[\tau, t]$ when every session $j \neq i$ is greedy starting at time τ. Thus

$$\hat{S}_i(\tau,t) = \begin{cases} 0, & \text{for} \quad t-\tau \quad \leq \frac{\sum_{j\neq i}\sigma_j^T}{1-\sum_{j\neq i}\rho_j} \\ (1-\sum_{j\neq i}\rho_j)\Delta & \text{for} \quad t-\tau-\frac{\sum_{j\neq i}\sigma_j^T}{1-\sum_{j\neq i}\rho_j} = \Delta > 0. \end{cases} \tag{2.35}$$

Now since every session $j \neq i$ has priority over session i, and since $L_{\max} = 0$ this

subset of sessions is oblivious to session i. Thus Lemma 2.10 applies to the subset of sessions $\{j : j \neq i\}$ and

$$\sum_{j \neq i} \sigma_j^\tau < \sum_{j \neq 1} \sigma_j.$$

Thus $\hat{S}_i(\tau, t)$ is minimized under an all-greedy regime, and D_i^* is achieved under this regime as well. □

From Lemmas 2.18 and 2.17:

Theorem 2.4 *The worst-case session i delay over all locally FCFS disciplines is*

$$D_i^* = \frac{\sum_{j \neq i} \sigma_j}{1 - \sum_{j \neq i} \rho_j} + \frac{C_i b_i}{1 - \sum_{j \neq i} \rho_j} - b_i$$

for each session i, where $b_i = \frac{\sigma_i}{C_i - \rho_i}$. *Further, this delay is achieved for a strict priority service discipline under which session i has the least priority.*

When $C_j = 1$ for all j, we have:

$$\begin{aligned} D^* &= \frac{\sum_{j \neq i} \sigma_j}{1 - \sum_{j \neq i} \rho_j} + \frac{b_i \sum_{j \neq i} \rho_j}{1 - \sum_{j \neq i} \rho_j} \\ &= \frac{\sum_{j \neq i} \sigma_j + b_i \sum_{j \neq i} \rho_j}{1 - \sum_{j \neq i} \rho_j} \end{aligned} \tag{2.36}$$

Also notice that $\lim_{C \to \infty} C_i b_i = \sigma_i$. Thus for the case $C_i = \infty$:

$$D^* = \frac{\sum_{j=1}^{N} \sigma_j}{1 - \sum_{j \neq i} \rho_j}.$$

2.10 Summary and Suggestions for Further Work

The major contribution of this chapter was an analysis of the single GPS server when the sources are constrained by leaky buckets. We presented an efficient algorithm to determine worst-case delays for such a system and discussed methods to add new users to the system. We were also able to use the techniques developed for the GPS analysis to gain insights into other service disciplines such as FCFS.

There are a number of directions in which the work in this chapter can be extended:

- It is disappointing that Theorem 2.1 does not extend easily to the case in which $C_j \in (\rho_j, r)$ for some session j, where r is the rate of the server. We will discuss how one might approach this case in Chapter 5.

- It may be useful to generalize the leaky bucket constraint so that

$$A_i(\tau, t) \leq f_i(t - \tau)$$

  for all τ, t, where f_i is a convex-$\cap$ function. This has been done for FCFS and some other disciplines by Cruz [4]. We suspect that the all-greedy regime will still maximize delay and backlog for GPS. Also, this will resolve the issue of analyzing the case in which the sessions are leaky bucket constrained and the only constraint on C_j is that $C_j > \rho_j$ for each j (i.e. the issue discussed earlier).

- We mentioned that the universal service curve can be computed efficiently but did not give any details on how this can be done. It would be especially useful to exploit properties of the curve so that it can be quickly re-computed when a new session is to be added. This would speed up the the procedure described in Section 2.7.

- Stochastic analysis of GPS: In this model the sessions are not leaky bucket constrained but the arrivals are described by a stochastic process. We briefly discussed the problems and issues related to this area in Section 2.8.

- The application of PGPS to computer time-sharing, where the sessions are jobs, and a packet corresponds the amount of service given to a job in a time-sharing quantum.

- PGPS with switch-over costs: In this model the sessions are leaky bucket constrained, but the server takes time to move between sessions, and so is not work

conserving. A simple extension of GPS that includes switch-over costs does not make sense since the server spends only an infinitesimal amount of time uninterrupted at each session. A service discipline that assures fair rates in the sense of providing service guarantees proportional to quantities such as the ϕ's of GPS might be unddesirable, since the server may spend a significant fraction of its time switching between sessions, and consequently may only be able to assure very low guaranteed rates. On the other hand, providing high overall throughput at the expense of fairness might be equally undesirable. The figure of merit for the service discipline should find a balance between fairness and efficiency.

- We have tried to determine the class of service disciplines for which D_i^* and Q_i^* are achieved by all-greedy regimes when the sessions are leaky bucket constrained. Since this class is likely to be quite broad, such a characterization could be very useful in analyzing and comparing the merits of diverse service disciplines.

Chapter 3

Arbitrary Topology GPS Networks

In this chapter we extend the analysis of Chapter 2 to arbitrary topology networks of GPS servers. As in the single node case, the sources are constrained by leaky buckets so that the amount of traffic entering the network from any session i is

$$A_i(\tau, t) \leq \min\{(t - \tau)C_i, \sigma_i + \rho_i(t - \tau)\}, \quad \forall t \geq \tau \geq 0. \tag{3.1}$$

The main question addressed is the following: Given a network with the values of the server parameters fixed and a set of leaky bucket constrained sessions, what is the worst-case session delay and backlog for each of the sessions? We find that a broad class of assignments, called Consistent Relative Session Treatment (CRST), is flexible enough to accommodate a wide variety of session delay constraints, and is also amenable to analysis in arbitrary topology networks. We make considerable progress on providing good bounds on session delay and backlog for CRST, and for some other classes of GPS networks, and further, these bounds can be efficiently computed. The results of this chapter will be useful in the analysis of PGPS networks, which we defer to Chapter 4.

The outline of this chapter is the following: In Section 3.1 we set up our model of the network and specify notation. Then the concepts of network backlog and delay are discussed and graphically interpreted. The phenomenon of virtual feedback, that complicates the analysis of networks with cycles is discussed in Section 3.3. In Section 3.4, several classes of server parameter assignments are analyzed for which virtual feedback effects are absent. Rate Proportional Processor Sharing networks are also discussed in this section. In Section 3.5, more general assignment schemes are considered, and algorithms for characterizing the traffic within the network are presented. Then networks with given internal traffic characterization are analyzed to yield bounds on worst-case session backlog and delay. The effects of propagation delay are incorporated in Section 3.7. Conclusions are in Section 3.8.

3.1 The Network Model

The network is modeled as a directed graph in which nodes represent switches, and arcs represent links. Let there be N sessions using the network. A route is a path in the graph, and the path taken by session i is defined as $P(i)$. Let $P(i,k)$ be the k^{th} node in $P(i)$, and K_i be the total number of nodes in $P(i)$. The rate of the server at node m is r^m.

The amount of session i traffic that enters the network in the interval $[\tau, t]$ is given by $A_i(\tau, t)$. As in Chapter 2, $A_i \sim (\sigma_i, \rho_i, C_i)$, where $C_i \geq r^{P(i,1)}$. Let $S_i^{(k)}$, $k = 1, \ldots, K_i$, be the function that describes session i traffic after it has been served by k nodes. Thus, $S_i^{(K_i)}$ is the traffic that leaves the network. We characterize $S_i^{(k)}$ by $(\sigma_i^{(k)}, \rho_i, C_i^{(k)})$.

Often, we will analyze what happens at a particular server, m. In this case the notation described above becomes overly cumbersome. Define $I(m)$ to be the set of sessions that are served by server m. For every session $i \in I(m)$, let the arrival function into that node be described by $A_i^m \sim (\sigma_i^m, \rho_i, C_i^m)$ and the departure function

be described by $S_i^m \sim (\sigma_i^{m,out}, \rho_i, r^m)$. For example, at server 0 in Figure 3-1: $A_0^0 = A_0$, $A_2^0 = A_2^{(2)}$, and $A_3^0 = A_3^{(1)}$. It is worth noting that when $k = P(i,j)$ for a particular session, i, the functions $S_i^{(j)}$ and S_i^k are identical. The rate of the link associated with server m is denoted by r^m. The value of ϕ_i at m is given by ϕ_i^m for all $i \in I(m)$.

3.2 Network Delay, Backlog and Stability

In this section we extend the notions of session i delay and backlog introduced in Chapter 2 to the multiserver case. Given a set of arrival functions for every session in the network, we define $Q_i^{(k)}(t)$ be the session i backlog at node $P(i,k)$ at time t. Similarly, let $Q_i^m(t)$ be the session i backlog at node $m \in P(i)$. Thus, if $m = P(i,k)$, then

$$Q_i^{(k)}(t) = Q_i^m(t) = A_i^m(0,t) - S_i^m(0,t).$$

Define the total session i backlog at time t to be

$$Q_i(t) = \sum_{k=1}^{K_i} Q_i^{(k)}(t). \tag{3.2}$$

Thus, $Q_i(t)$ is the amount of session i traffic buffered in the network at time t. Also, let $D_i(t)$ be the time spent in the network, by a session i bit that arrives at time t. Figure 3-2 shows how to represent the notions of backlog and delay graphically. We see that $D_i(\tau)$ is the horizontal distance between the curves $A_i(0,t)$ and $S_i^{(K_i)}(0,t)$ at the ordinate value of $A_i(0,\tau)$.

Clearly, $D_i(\tau)$ depends on the arrival functions $A_1, ..., A_N$ (recall that the total number of sessions in the network is N). We are interested in computing the maximum delay over all time, and over all arrival functions that are consistent with (1.11). Let D_i^* be the maximum delay for session i. Then

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

be described by $S_i^m \sim (\sigma_i^{m,out}, \rho_i, r^m)$. For example, at server 0 in Figure 3-1: $A_0^0 = A_0$, $A_2^0 = A_2^{(2)}$, and $A_3^0 = A_3^{(1)}$. It is worth noting that when $k = P(i,j)$ for a particular session, i, the functions $S_i^{(j)}$ and S_i^k are identical. The rate of the link associated with server m is denoted by r^m. The value of ϕ_i at m is given by ϕ_i^m for all $i \in I(m)$.

3.2 Network Delay, Backlog and Stability

In this section we extend the notions of session i delay and backlog introduced in Chapter 2 to the multiserver case. Given a set of arrival functions for every session in the network, we define $Q_i^{(k)}(t)$ be the session i backlog at node $P(i,k)$ at time t. Similarly, let $Q_i^m(t)$ be the session i backlog at node $m \in P(i)$. Thus, if $m = P(i,k)$, then

$$Q_i^{(k)}(t) = Q_i^m(t) = A_i^m(0,t) - S_i^m(0,t).$$

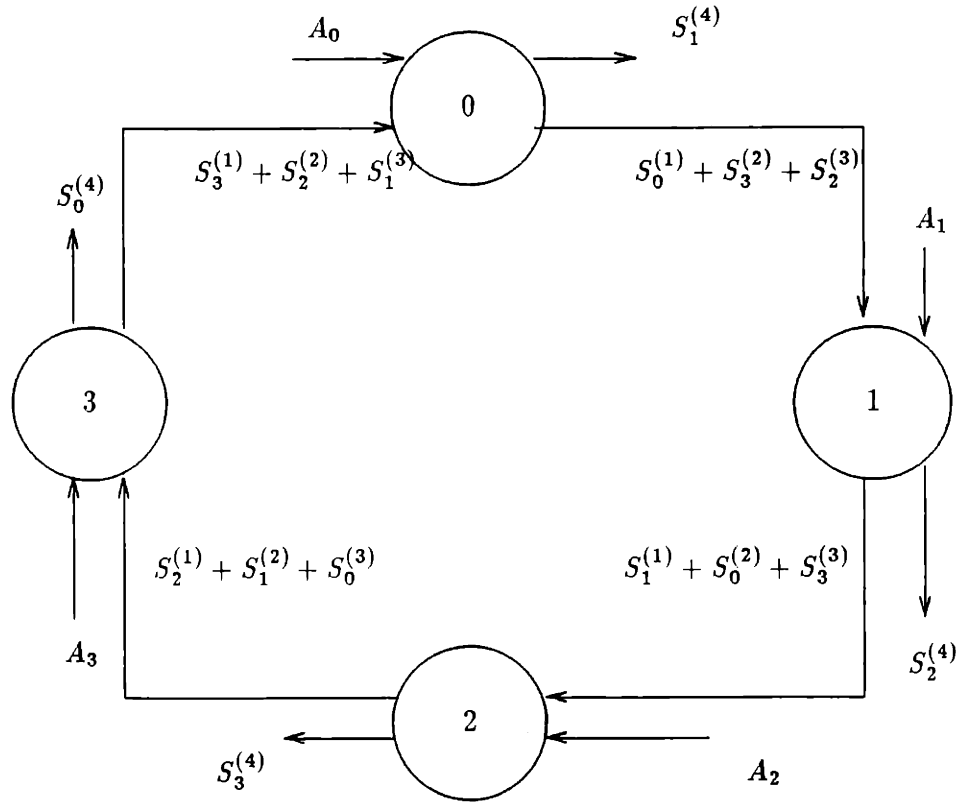
Define the total session i backlog at time t to be

$$Q_i(t) = \sum_{k=1}^{K_i} Q_i^{(k)}(t). \tag{3.2}$$

Thus, $Q_i(t)$ is the amount of session i traffic buffered in the network at time t. Also, let $D_i(t)$ be the time spent in the network, by a session i bit that arrives at time t. Figure 3-2 shows how to represent the notions of backlog and delay graphically. We see that $D_i(\tau)$ is the horizontal distance between the curves $A_i(0,t)$ and $S_i^{(K_i)}(0,t)$ at the ordinate value of $A_i(0,\tau)$.

Clearly, $D_i(\tau)$ depends on the arrival functions $A_1, ..., A_N$ (recall that the total number of sessions in the network is N). We are interested in computing the maximum delay over all time, and over all arrival functions that are consistent with (1.11). Let D_i^* be the maximum delay for session i. Then

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



Server 0

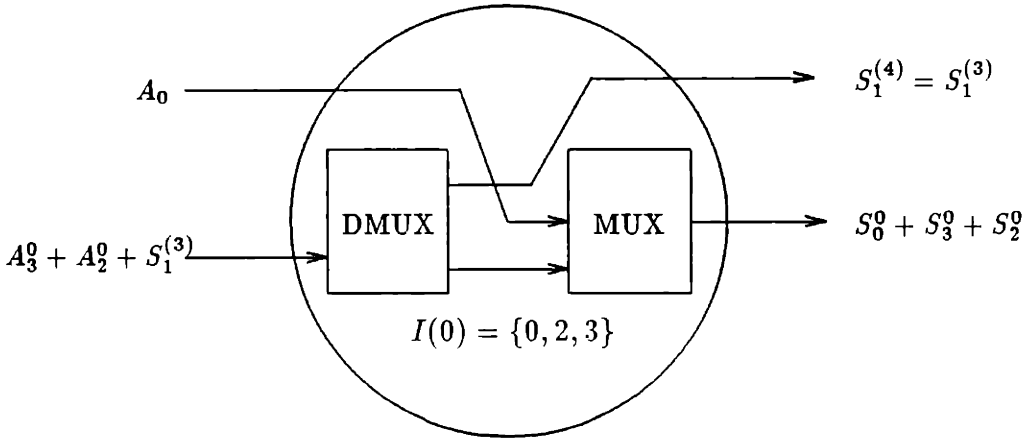
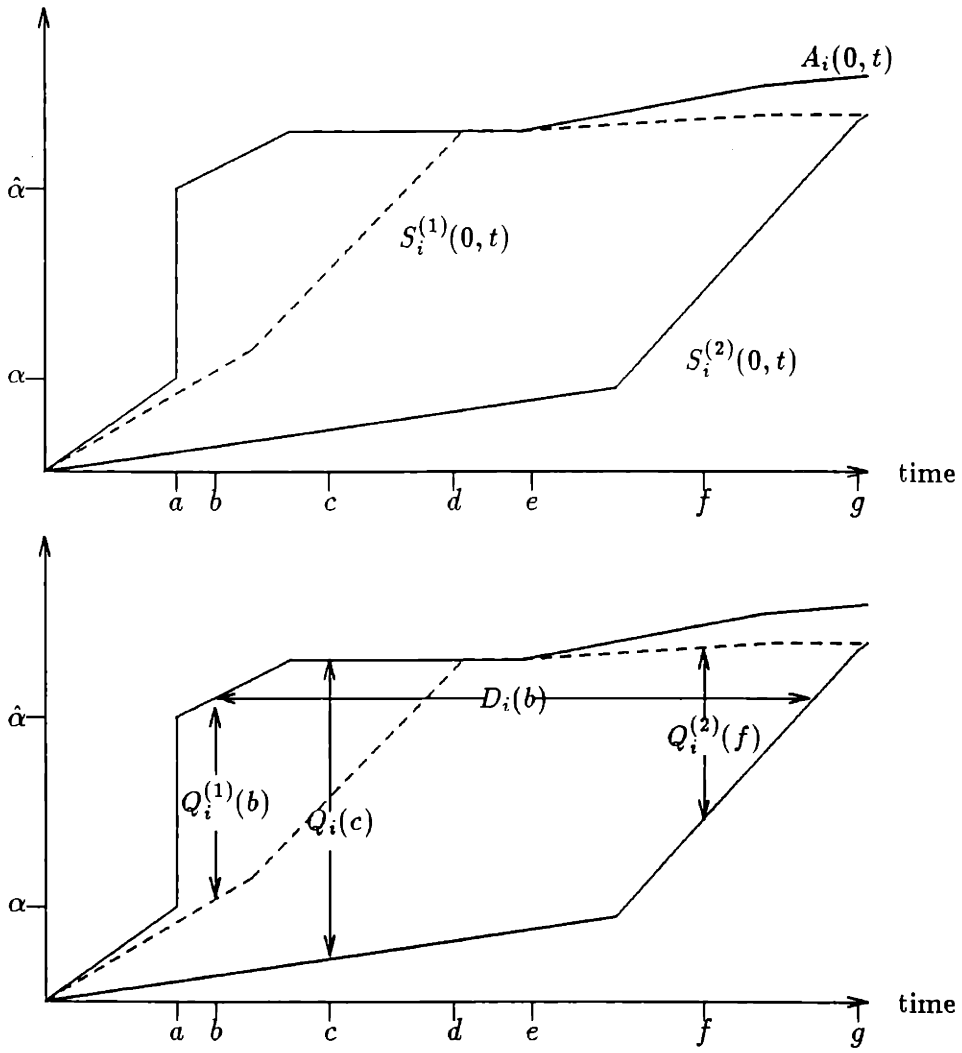


Figure 3-1: A four server network. The demultiplexer works instantaneously.



The first figure shows how session i traffic progresses through the nodes of its route. Notice that the arrival function to node 2 is the session i service function of node 1.
The second figure shows how the backlog and delay can be measured and illustrates the definitions of Section 3.2.

Figure 3-2: An Example of Session i flow when $K_i = 2$.

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

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

The backlogs at every node in $P(i)$ can be determined from Figure 3-2 as shown. Note that A_i contains an impulse at time a; this is possible only if $C_i = \infty$. As in Chapter 2, we adopt the convention that the arrival functions are continuous from the left, so that $A_i(0,a) = \alpha$ and $A_i(0,a^+) = \hat{\alpha}$.

Define the utilization of server m to be

$$u^m = \frac{\sum_{j \in I(m)} \rho_j}{r^m}. \tag{3.3}$$

A network is defined to be *stable* if $D_i^* < \infty$ for all sessions i. In most of our analysis we will show stability under the assumption that $u^m < 1$ at every server m. Allowing utilizations of greater than 1 would permit infinite backlogs and delays to build up unboundedly, and as we shall see in Section 3.3, permitting $u^m = 1$ at each server m can result in problems as well.

Finally, we extend the definitions of system and session busy periods given in Chapter 2 for a single node, to the multiple node case. A network system (session i) busy period is defined to be the maximal interval B (B_i) such that for every $\tau \in B$ ($\tau \in B_i$), there is at least one server in the network that is in a system (session i) busy period at time τ.

3.3 Virtual Feedback

While every route in a data network is acyclic, the union of several routes may result in cycles being induced in the network topology. The presence of these cycles can complicate the analysis of delay considerably, but more importantly, it can lead to feedback effects that drive the system towards instability. This phenomenon has

been noticed by researchers from fields as diverse as manufacturing systems [20, 18], communication systems [5] and VLSI circuit simulation [17].

It is important for us to understand the circumstances under which this tendency towards instability is manifested, so that we can avoid assignments of the ϕ_i's for which reasonable delay guarantees cannot be made. In this section we will discuss some of the analytical difficulties that emerge in non-acyclic networks and will also provide some insight into why such networks may not be stable for particular service disciplines. In subsequent sections we will demonstrate that GPS networks are stable for a wide variety of assignments, and we will also describe how to make delay guarantees for these networks.

We begin by analyzing simple ring networks such as the four node example in Figure 3-1 for which $\rho_i = \rho$ and $C_i = \infty$: $i = 0, 1, 2, ..., K-1$, $K \geq 4$. To make things even simpler, assume that each server works at rate 1. Notice that each server, k, has *only two* incoming links–one has traffic coming in consistent with (σ_k, ρ, ∞) and the other has the combined traffic from sessions $k+2, k+3, ..., k+K-1$ (mod K). Notice that this topology is identical to Example 2 of Cruz [5].

Following Cruz [5], let the service discipline work as follows: Server k gives session k the lowest priority for $k = 0, 1, 2, 3, ..., K-1$. In terms of GPS, we can think of $\phi_k^k \approx 0$ and $\phi_i^k = 1$ when $i \neq k$, for $k = 0, 1, 2, ..., K-1$. To understand the analytical difficulties involved consider the following argument for analyzing any session i: First notice that since each link has capacity 1 it follows that session k can only experience delay at server k. Thus $\sigma_k^m = \sigma_k^n$ for any two nodes $m, n \in P(k) - \{k\}$. Now consider a single server k in isolation. Assuming stability, the burstiness of the session k traffic leaving the server (only session k experiences positive delay at this server) can be given by by Lemma 2.18 of Section 2.9.2:

$$\sigma_k^{k+1} \leq \sigma + \rho \left(\frac{\sum_{m=2}^{K-1} \sigma_{k+m}^{k+1}}{1 - (K-2)\rho} \right) \tag{3.4}$$

since every server is a locally first-come-first-serve multiplexer that yields worst case

delay. (This is also shown for the two session case in Remark 3 of the proof for Theorem 4.2(b) of [5].) The bound in (3.4) is tight for a single server system. Now observe from the symmetry of the problem that $\sigma_i^1 = \sigma_1^1$ for all i. Thus we have

$$\sigma_k^1 \leq \sigma + \rho\left(\frac{(K-2)\sigma_1^1}{1-(K-2)\rho}\right) \tag{3.5}$$

$$\Rightarrow \sigma_k^1 \leq \left(\frac{1-(K-2)\rho}{1-2(K-2)\rho}\right)\sigma. \tag{3.6}$$

Notice that if we allow the values of σ and ρ to vary over the sessions, the solution of the system of equations in (3.4) will be considerably more involved, and perhaps even impractical to evaluate for large networks. (See [4] for more on this.)

For large K this bound in (3.6) blows up for $\rho \approx \frac{1}{2K}$, implying that the maximum possible utilization of a link in a stable system is about $\frac{1}{2}$ for large K. However, we strongly suspect that this implication is misleading. For example, for $K = 4$, Gallager [8] has shown that the ring is stable for **any** work conserving discipline as long as $\rho < \frac{1}{3}$. The bound of 3.6 suggests that it should be less than $\frac{1}{4}$. The proof of this result is not included since it does not seem to be conveniently generalizable to larger rings. We do, however, conjecture the following:

Conjecture 3.1 *A ring network with $A_i \sim (\sigma_i, \rho, C)$, $\forall i$, and $\rho < \frac{1}{K-1}$, is stable for any work-conserving discipline.*

We believe the bound of (3.6) to be weak because of the difficulty in dealing with virtual feedback. Notice that the expression for σ_k^{k+1} in (3.4) is in terms of $K-2$ other unknowns. In order to eliminate these unknowns, we have to solve a system of inequalities, which we do in going from (3.4) to (3.5). The single server approximation made in (3.4) is extremely weak when it is made for all the servers simultaneously, as it is in the above analysis. It is not clear what other approximation we could use—perhaps we could characterize the through traffic at node k in a more accurate manner than with σ, ρ constraints, but this leads to further analytical complications.

Thus, the approach taken in the subsequent sections of this chapter is to restrict the analysis of GPS networks to allocations in which the virtual feedback effects are easier to characterize. Fortunately, the range of classes admitted under this restriction is extremely broad.

So far we have explained the analytical difficulties of dealing with virtual feedback. Another serious problem is that networks may be actually driven to instability by this phenomenon. Our stipulation in Conjecture 3.1 that ρ be strictly less than $\frac{1}{K-1}$ is not merely a technical restriction, since the ring network **can** be unstable when $\rho = \frac{1}{K-1}$. Assume the service discipline considered earlier, i.e., through traffic gets priority. We now exhibit an arrival pattern which leads to instability for this choice of ρ when we allow every session to have a different value of σ—i.e. $A_i \sim (\sigma_i, \rho, \infty)$.

Suppose that session 0 traffic begins arriving in a greedy fashion starting at time 0, and traffic from sessions $1, 2, ..., K-2$ begins arriving in a greedy fashion starting at time 0^+. Further, suppose that session $K-1$ does not send any traffic in the interval $[0, t_1]$, where

$$t_1 = \frac{\sigma_0}{1-\rho}.$$

Then session 0 gets exclusive use of the servers $0, 1, 2, ..., K-2$ during the interval $[0, t_1]$ and its backlog is zero at time t_1. Also, there are exactly $\sigma_i + \rho t_1$ units of traffic backlogged at each node $i = 1, 2, ..., K-2$ at time t_1. Now let session 0 stop sending traffic at time t_1, which allows the backlogged session 1 traffic to enter the network, and to instantaneously get exclusive use of nodes $1, 2, ..., K-1$. Let session $K-1$ submit a burst of size σ_{K-1} at time t_1^+. This burst will not get through as long the session 1 backlog is being cleared. This takes time

$$t_2 = \frac{\sigma_1 + \rho t_1}{1-\rho}.$$

Note that session 0 *does not send any traffic in the interval* $[t_1, t_2]$. At time $t_1 + t_2$, session 1 stops sending traffic, allowing the session 2 traffic to get through. Now

session 0 drops a burst of size σ_0 at time $(t_1 + t_2)^+$. Since the session 0 bucket is empty of tokens at time t_1 it follows that we must choose σ_0 such that

$$\sigma_0 \leq \rho t_2 \tag{3.7}$$

Similarly, we may choose $\sigma_2, ...\sigma_{K-1}$ to extend the pattern of arrivals just described. We now show that the total amount of backlogged traffic grows unboundedly with time when $\rho = \frac{1}{K-1}$. Suppose that a session i gets exclusive use of the servers in its route starting at time T, and let its backlog at that time be q. This backlog will clear after $\frac{q}{1-\rho}$ time units, during which $K - 2$ of the sessions will have increased their backlogs by $\frac{\rho q}{1-\rho}$. Session $i - 1 \pmod K$ sends no traffic during the interval $[T, T + \frac{q}{1-\rho}]$, and drops a burst of size σ_{i-1} at time $(T + \frac{q}{1-\rho})^+$. Thus the net change in the total backlog at time $(T + \frac{q}{1-\rho})^+$ is

$$\sigma_{i-1} + (K-2)\frac{\rho q}{1-\rho} - q = \sigma_{i-1} - q(1 - \frac{(K-2)\rho}{1-\rho})$$

$$= \sigma_{i-1} - q(\frac{1-(K-1)\rho}{1-\rho}).$$

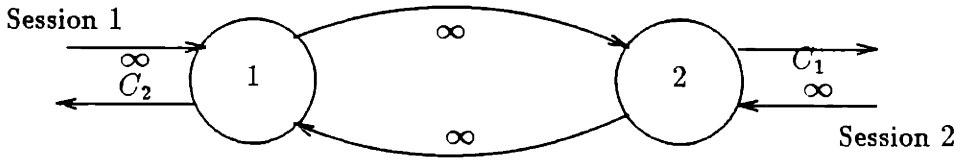
Setting $\rho = \frac{1}{K-1}$ we see that the net change in total backlog is exactly σ_{i-1}. Thus the total backlog grows unboundedly with time, for this choice of ρ. Table 3.1 shows the behavior of the system when $K = 4$ and $\rho = \frac{1}{3}$.

Finally, consider the following two node example (adapted from [18]) in which the nodes act as switches rather than multiplexers. Assume that starting at time 0, traffic enters the network from both sessions at a constant rate of 1. Each switch serves its incoming sessions exhaustively—i.e. once it has begun serving the traffic of a given session i, it does not serve traffic from any other session until it has cleared the session i backlog. Corresponding to every incoming session at a node, is an outgoing link. The rate at which the node serves traffic from a session is always equal to the rate of that session's outgoing link at the node. Then, we would expect that as long as

time/backlog	0	1	2	3	Total Backlog
0	2/3	1	1	0	8/3
1	0	4/3	4/3	0	8/3
1^+	0	4/3	4/3	1	11/3
3	0	0	2	5/3	11/3
3^+	2/3	0	2	5/3	13/3
6	5/3	0	0	8/3	13/3
6^+	5/3	1	0	8/3	16/3
10	3	7/3	0	0	16/3
10^+	3	7/3	1	0	19/3
14.5	0	23/6	5/2	0	19/3
14.5^+	0	23/6	5/2	2/3	7

$\sigma_0 = 2/3, \sigma_1 = \sigma_2 = \sigma_3 = 1$ and $\rho = \frac{1}{3}$

Table 3.1: Instability for a four node ring network. The total backlog grows unboundedly with time.



All the links are of infinite capacity except the links through which the sessions exit the network.

Figure 3-3: A potentially unstable two node switching network.

$C_1 > 1$ and $C_2 > 1$, the backlogs remain bounded over time.

As before, let $Q_i^m(t)$ be the session i backlog at time t at node m. Now suppose that $Q_1^2(0) = K$, and all other buffers are empty at $t = 0$: Server 2 begins to serve session 1 starting at $t = 0$ and clears it at time

$$t_1 = \frac{K}{C_1 - 1}.$$

Now $Q_2^2(t_1) = t_1$, since no session 2 packets can be served while server 2 is clearing its session 1 backlog. At t_1, server 2 clears its session 2 backlog instantaneously, and so we have $Q_2^1(t_1) = t_1$. The session 2 buffer at node 1 is cleared at time

$$t_1 + t_2 = t_1 + \frac{t_1}{C_2 - 1} = t_1 + \frac{K}{(C_1 - 1)(C_2 - 1)}.$$

Again, no session 1 packets can be served in the interval $[t_1, t_1+t_2]$, and so $Q_1^1(t_1+t_2) = t_2$. Since server 1 will clear its session 1 backlog instantaneously, we have

$$Q_1^2(t_1 + t_2) = \frac{K}{(C_1 - 1)(C_2 - 1)}.$$

Now observe that all the other buffers are empty at time $t_1 + t_2$, and so we are back to the situation at time 0. The system will be unstable if

$$\frac{1}{(C_1 - 1)(C_2 - 1)} > 1 \Rightarrow \frac{1}{C_1} + \frac{1}{C_2} > 1.$$

Thus the system could be unstable even when both C_1 and C_2 are greater than 1. The problem is that if $Q_i^m(0) > 0$ for any i and m, then the system is forever constrained to work only on *one session at a time*. Thus one of the servers is always underutilized, or as Kumar puts in [18], *starved* for packets.

3.4 Rate-Proportional Processor Sharing

In this section we will analyze several special cases of Generalized Processor Sharing for which the feedback effects discussed in Section 3.3 are completely absent. This property will allow us to give succinct bounds on delay and backlog that are independent of the topology of the network.

We have shown in Lemma 2.6 of Chapter 2 that each session i at node m is guaranteed a service rate of

$$g_i^m = \frac{\phi_i^m}{\sum_{j \in I(m)} \phi_j^m} \, r^m \tag{3.8}$$

where r^m is the rate of server m.

Then session i is guaranteed a rate of

$$g_i = \min_{m \in P(i)} g_i^m \tag{3.9}$$

at every node in its route. Let us now examine how to characterize D_i^* and Q_i^* for a session i such that

$$g_i \geq \rho_i.$$

Lemma 3.1 *Suppose that $g_i \geq \rho_i$ and $K_i = 1$. Then*

1. $Q_i^* \leq \sigma_i$
2. $D_i^* \leq \frac{\sigma_i}{g_i}$
3. $\sigma_i^{out} = \sigma_i$.

Proof. In any single server GPS system Q_i^* and D_i^* are both achieved when all the sessions are greedy starting at time zero, the beginning of a system busy period: Thus

$$Q_i^* = \max_{t \geq 0}(\min\{C_i t, \sigma_i + \rho_i t\} - S_i(0,t)) \leq \max_{t \geq 0}(\sigma_i + \rho_i t - S_i(0,t)).$$

Since $g_i \geq \rho_i$, the service rate during this interval is greater than ρ_i. Thus

$$Q_i^* \leq \max_{t \geq 0}(\sigma_i + \rho_i t - \rho_i t) \leq \sigma_i.$$

Now note that an arriving session i bit can see a backlog of size at most Q_i^* and is guaranteed a rate of at least g_i. Thus

$$D_i^* \leq \frac{Q_i^*}{g_i}.$$

Finally, from Proposition 2.5 of Chapter 2:

$$\sigma_i^{out} = \sigma_i.$$

□

The bounds for Q_i^* and D_i^* are not the tightest we can find, but they have the appeal of being independent of the behavior of sessions other than i.

Now consider the case $K_i \geq 1$: If $g_i \geq \rho_i$ at each node in the session i route, then Lemma 3.1 ensures that no session i traffic is delayed by more than $\frac{\sigma_i}{g_i}$ at any node. Thus $D_i^* \leq K_i \frac{\sigma_i}{g_i}$, implying that Q_i^* is bounded.

We now show how to get a much better bound on D_i^*. The following straightforward, yet useful Lemma is stated without proof—to see that it is true, recall that we are ignoring propagation delays:

Lemma 3.2 *For every interval $[\tau, t]$ that is contained in a single session i network busy period:*

$$S_i^{(K_i)}(\tau, t) \geq g_i\ (t - \tau).$$

We can now state and prove the major result of this section:

Theorem 3.1 *If $g_i \geq \rho_i$ for session i:*

$$Q_i^* \leq \sigma_i,$$

$$D_i^* \leq \frac{\sigma_i}{g_i}.$$

Note that the delay bound in Theorem 3.1 is independent of the topology of the network and number of links in the route taken by the session. Also, it is independent of the σ_j, $j \neq i$.

Proof. Suppose Q_i^* is achieved at time t, and let τ be the first time before t when there are no session i bits backlogged in the network. Then by Lemma 3.2, $S_i^{(K_i)}(\tau, t) \geq \rho_i(t - \tau)$. Consequently,

$$Q_i^* \leq (\sigma_i + \rho_i(t - \tau)) - \rho_i(t - \tau) = \sigma_i.$$

An arriving session i bit will be served after at most Q_i^* session i bits have been served. Using Lemma 3.2 again, these backlogged bits are served at a rate of at least g_i. Therefore:

$$D_i^* \leq \frac{Q_i^*}{g_i} \leq \frac{\sigma_i}{g_i}.$$

□

This result is quite broadly applicable since it is valid for any GPS assignment for the other sessions. The other sessions need not be leaky bucket constrained, nor need the system be stable.

An important special case of the allocations we have been analyzing is the following:

Define a *Rate-Proportional Processor Sharing* network to be a GPS network for which:

$$\phi_i^m = \rho_i \tag{3.10}$$

at every node, m, and every session $i \in I(m)$. Thus,

$$\frac{\phi_i^m}{\phi_j^m} = \frac{\rho_i}{\rho_j}$$

for all $i, j \in I(m)$. Substituting (3.10) into (3.8) we have:

$$g_i^m = \rho_i \frac{r_m}{\sum_{j \in I(m)} \rho_j} > \rho_i$$

for every session i and $m \in P(i)$. From Theorem 3.1:

$$Q_i^* \leq \sigma_i, \quad D_i^* \leq \frac{\sigma_i}{\rho_i}.$$

If we define the "relative burstiness" of session i to be $\frac{\sigma_i}{\rho_i}$ then the delay guarantees under RPPS are in proportion to the session relative burstiness. This is in keeping with policies prescribed by Yates [24] and others. However, the drawback of RPPS allocation is that the delay requirements of individual sessions are not taken into account in determining the ϕ_i's.

3.5 Characterizing internal session traffic

Our analysis of RPPS involved two steps: we first characterized the session traffic inside the network, (in Lemma 3.1), and we then found bounds on D_i^* and Q_i^* given this characterization. In the rest of the chapter we will use these two steps to find bounds on the worst-case session delay and backlog for various kinds of GPS networks. An important point of departure from our analysis of RPPS is that for a given node m, and session $i \in I(m)$, we will only be able to calculate *upper bounds* on the least quantity $\sigma_i^{m,out}$ such that $S_i^m \sim (\sigma_i^{m,out}, \rho_i, r^m)$. In this section we concentrate on obtaining these bounds for two broad classes of networks. However, it is first necessary to introduce the important concept of the All-Greedy bound:

3.5.1 The All-Greedy Bound for a single node

Consider a particular node m. Suppose that for every $j \in I(m)$, we are given that $A_j^m \sim (\sigma_j^m, \rho_j, C_j^m)$, $C_j^m \geq r^m$. We know that worst-case delay and backlog for session

i (at node m) is achieved when all the sessions $j \in I(m)$ are simultaneously greedy from time zero, the beginning of a system busy period. However, if two sessions j and p are both served by the same node, n, just before they contend for node m, then it may not be possible for both of them to be simultaneously greedy, as is required in the All-Greedy regime. Thus, the achievable worst-case delay and backlog at node m may be less (but never more) than that calculated under the All-Greedy regime.

In the rest of this chapter we will make frequent use of the all-greedy bound, in order to simplify procedures for estimating D_i^* and Q_i^*. The following notation is useful in this regard:

We are given $\sigma_j^m, \rho_j, C_j^m$ for each $j \in I(m)$, such that $\sum_{j \in I(m)} \rho_j < r^m$. Consider a fictitious system in which no traffic enters node m before time zero, and all the sessions at m are greedy starting at time zero. Denote $\hat{A}_i^m$ as the resulting session i arrival function for all $i \in I(m)$. Also denote $\hat{S}_i^m$ as the service curve at node m. Recall from Chapter 2, that for $t > 0$, as long as $Q_i^m(t) > 0$, the function $\hat{S}_i^m(0,t)$ is piecewise linear and convex-$\cup$. By using the techniques of Chapter 2 we can find $\hat{\sigma}_i^{m,out}$ such that $\hat{S}_i^m \sim (\hat{\sigma}_i^{m,out}, \rho_i, r^m)$. Note that $\hat{\sigma}_i^{m,out}$ is the least quantity characterizing the burstiness of $\hat{S}_i^m$. From the discussion above,

$$\hat{\sigma}_i^{m,out} \geq \sigma_i^{m,out}. \tag{3.11}$$

Thus, we may bound the burstiness of S_i^m by $\hat{\sigma}_i^{m,out}$.

3.5.2 Acyclic Networks

We make use of the following elementary property of acyclic graphs:

Lemma 3.3 *For any acyclic network there exists a labeling of the nodes so that every link terminates at a higher labeled node than it originates from.*

Proof. Let the directed graph associated with the acyclic network be denoted G, and define a node that has no incoming arcs to be a start node. There must be at

least one start node in G. (Otherwise G will contain a cycle.) Pick any start node, and label it 1. Now the graph $G - \{1\}$ is also acyclic, and must contain a start node as well. Pick such a start node, and label it 2. Continue this process until all nodes are labeled. Now it is easy to see that the Lemma holds for this labeling. □

Assume the labeling outlined in the proof of the Lemma. Notice that for any session, i, such that $i \in I(1)$, we must have $P(i,1) = 1$, i.e. node 1 must be the first node encountered by session i. Then we can use the results for the single node case in Chapter 2 to calculate $\hat{\sigma}_i^{1,out}$ for every $i \in I(1)$. Now it is easy to see that by considering the nodes in the order that they are labeled, and by using the single node results, we can compute $\hat{\sigma}_i^m$, for every node m and session $i \in I(m)$. Now notice that:

Lemma 3.4 *Suppose session i is in $I(k)$ for some node k, and that for every session $j \in I(k)$, σ_j^k is bounded. Then $\sigma_i^{k,out}$ must be bounded as well.*

As a consequence of the labeling procedure, and from Lemma 3.4:

Lemma 3.5 *An acyclic network is stable if and only if $u^m \leq 1$ at every node m.*

3.5.3 Non-Acyclic GPS networks under Consistent Relative Session Treatment

In this section we address the problem of characterizing internal session traffic when cycles may be induced in the network through the union of session routes. One of the major causes of the large delays in the example of Section 3.3, is that a given session is treated poorly relative to a set of sessions at a particular node, but is treated at least as well relative to the same set of sessions at other nodes. In this section we will consider a broad class of assignments under which this cannot happen. We begin by giving two useful definitions:

Definition. *Session j is said to <u>impede</u> a session i, at a node m if*

$$\frac{\phi_i^m}{\phi_j^m} < \frac{\rho_i}{\rho_j}.$$

Note that for any two sessions, i and j, that contend for a node m, either session i impedes session j or vice-verse, unless $\frac{\phi_i^m}{\phi_j^m} = \frac{\rho_i}{\rho_j}$, in which case neither session impedes the other.

Definition. *A Consistent Relative Session Treatment GPS assignment (CRST) is one for which there exists a strict ordering of the sessions such that for any two sessions i, j, if session i is less than session j in the ordering, then session i does not impede session j at any node of the network.*

The class of assignments that are CRST is quite broad: For example, consider the special case of a CRST system for which

$$\phi_{ij} = \frac{\phi_i^m}{\phi_j^m}, \quad \forall m \text{ s.t. } i, j \in I(m). \tag{3.12}$$

Thus, whenever sessions i and j contend for service at a link, they are given the same relative treatment. Note that $\phi_{ij} = \frac{\phi_{ip}}{\phi_{jp}}$, where session p is in $I(i) \cap I(j)$. Such CRST systems are called Uniform Relative Session Treatment (URST) systems. Note that

- By normalizing the values of the ϕ_i^m's at each node m, we may equivalently define a URST system to be one in which for every session i, and node m that is on the session i route: $\phi_i = \phi_i^m$.
- The Rate Proportional Processor Sharing system that we studied in Section 3.4 is a special case of a URST system.

We will show that every CRST system for which $u^m < 1$ at each node, is stable, and will also provide an algorithm for characterizing the internal traffic for every session in a CRST system. The following Lemma will be crucial to us:

Lemma 3.6 *Suppose sessions i and j contend for a link m, and that session j does not impede session i. Then the value of $\hat{\sigma}_i^{m,out}$ is independent of the value of σ_j^m.*

Proof. We will first consider the case $\sigma_j^m = 0$; the case $\sigma_j^m > 0$ will follow from this easily: As we explained in Chapter 2, under an all-greedy regime, the order in

which the sessions terminate their busy periods at node m corresponds to a particular feasible ordering of the sessions. Let this feasible ordering be $\mathcal{F}$ when $\sigma_j^m = 0$. Also, suppose that session j terminates its busy period at time e_j^0. (It is clear that if $\sigma_j^m > 0$ then the time at which this backlog would be cleared must be $\geq e_j^0$.) If session i is less than session j in the feasible ordering, $\mathcal{F}$, the Lemma clearly applies. Now suppose that session j is less than session i under $\mathcal{F}$: Let q_i be the (least) time at which maximum backlog is achieved for session i under the all greedy regime. Then

$$\hat{S}_j^m(0, e_j^0) = \sigma_j^m + \rho_j e_j^0 \geq \sigma_j^m + \frac{\rho_i \phi_j^m}{\phi_i^m} e_j^0.$$

Thus,

$$\hat{S}_i^m(0, e_j^0) = \frac{\phi_i^m \hat{S}_j^m(0, e_j^0)}{\phi_j^m} \geq \frac{\phi_i^m \sigma_j^m}{\phi_j^m} + \rho_i e_j^0 \geq \rho_i e_j^0.$$

This implies that $q_i \leq e_j^0$. Since the time at which the session j busy period terminates can only be greater than e_j^0 for arbitrary values of σ_j^m, we see that the maximum session i backlog is always achieved in the all-greedy regime before the session j busy period terminates. Since session j is in a busy period in the interval $[0, q_i]$, the value of q_i is independent of σ_j^m. Now recall from Chapter 2 that

$$\hat{\sigma}_i^{m,out} = Q_i^m(q_i)\}.$$

Thus, the value of σ_j^m does not influence the value of $\hat{\sigma}_i^{m,out}$. □

Lemma 3.7 *Suppose session i is in $I(m)$ for some node k, and that for every session $j \in I(m)$ that can impede i, σ_j^m is bounded. Then $\sigma_i^{m,out}$ must be bounded as well.*

Proof. From Lemma 3.6 it follows that $\hat{\sigma}_i^{m,out}$ is bounded. Now from (3.11) we are done. □

Partition the sessions into non-empty classes $H_1, ..., H_L$, such that the sessions in H_k are impeded only by those in H_l, $l < k$, and if two sessions i, j, are in the same

class, their routes are either edge disjoint or

$$\frac{\phi_i^m}{\phi_j^m} = \frac{\rho_i}{\rho_j}$$

at every node, m, that is common to the routes of sessions i and j. Then the sessions in H_1 are not impeded by *any* other session.

Lemma 3.8 *For any session $j \in H_1$:*

$$\rho_j < \frac{\phi_j^m}{\sum_{p \in I(m)} \phi_p^m} \tag{3.13}$$

for all nodes $m \in P(j)$.

Proof. Consider a session $j \in H_1$, and suppose that its route includes the node m. There must exist at least one session i, such that

$$\rho_i < \frac{\phi_i^m}{\sum_{p \in I(m)} \phi_p^m}.$$

By definition, i cannot impede session j. Therefore:

$$\frac{\phi_j^m}{\phi_i^m} \geq \frac{\rho_j}{\rho_i} > \frac{\rho_j \sum_{p \in I(m)} \phi_p^m}{\phi_i^m}$$

$$\Rightarrow \phi_j^m > \rho_j \sum_{p \in I(m)} \phi_p^m.$$

Now the claim is proven by rearranging the terms. □

For $j \in H_1$, (3.13) shows that j's guaranteed rate of service exceeds ρ_j so that

$$\hat{\sigma}_j^{m,out} = \sigma_j$$

. Thus from (3.11):

$$\sigma_j^{m,out} \leq \sigma_j \tag{3.14}$$

Lemma 3.8 enables us to characterize the internal traffic of all the sessions in H_1. Now using Lemmas 3.6 and 3.7 we can sequentially characterize the internal traffic of the sessions in classes $H_2, H_3, ..., H_L$. The following procedure specifies the method.

- Compute $H_1, ..., H_L$.
- $k = 1$

  While $k \leq L$, for each session $i \in H_k$

  For $p = 1$ to K_i

  $m = P(i,p)$

  Compute $\hat{\sigma}_i^{m,out}$ given:

  $\sigma_j^m = \hat{\sigma}_j^m$ for all sessions j that impede i at m (computed in earlier steps).

  $\sigma_j^m = 0$ for all sessions j that do not impede i at m.

  Set $\hat{\sigma}_j^{(p)} = \hat{\sigma}_i^{m,out}$.

  $k := k + 1$

Now use (3.11) to evaluate upper bounds to σ_i^m for every session i and node $m \in P(i)$.

This procedure enables us to show that

Theorem 3.2 *A CRST GPS network is stable if $u^m < 1$ at each node m.*

To conclude this section we point out an alternative characterization of CRST assignments:

Lemma 3.9 *An assignment is CRST if and only if the sessions can be numbered so that*

$$i \text{ impedes } j \Rightarrow i < j.$$

3.6 Computing Delay and Backlog for Stable Systems with Known Internal Burstiness

Suppose that we are given a stable GPS system in which the sessions are leaky bucket constrained as in (1.11). For every session j and node m such that $j \in I(m)$, we are given a value σ_j^m, such that $A_j^m \sim (\sigma_j^m, \rho_j, C_j^m)$, $C_j^m \geq r^m$. This section deals with the question of how to compute good bounds on D_i^* and Q_i^* for a particular session, i, given such a characterization of all the internal traffic of the network. In Section 3.6.1 we describe a method that is applicable to non-GPS networks as well as to GPS networks, but it yields bounds that are likely to be weak. When we restrict ourselves to GPS networks, the performance of the general procedure of Section 3.6.1 can be improved considerably.[1] In Section 3.6.2, we describe the notion of a Staggered Greedy regime, which will help us determine much better bounds on D_i^* and Q_i^*. A particular class of staggered greedy regimes, which we call (K,t)-staggered greedy regimes, plays a role analogous to that of the all-greedy regime in the single node system. In contrast to the all-greedy regime, which yields worst-case delay and backlog for *all* sessions, a staggered greedy regime is defined with respect to a *particular* session, i. We prove in Section 3.6.2 that Q_i^* and D_i^* are each achieved in (possibly different) session i (K,t)-staggered greedy regimes.

3.6.1 The Additive Method

As we discussed in Section 3.5.1, worst case delay (backlog) at a single node of the network can be upper bounded by applying the techniques of Chapter 2 when the traffic characterization of sessions sharing that node is known. Under the Additive Method due to [5], we add the worst case bounds on delay (backlog) for session i at each of the nodes $m \in P(i)$ considered in isolation. While this approach works for any

[1] This may be true for other networks as well, such as FCFS networks.

server discipline for which the single node can be analyzed, it may yield very loose bounds. For example, when applied to an RPPS system we get $D_i^* \leq K_i \frac{\sigma_i}{\rho_i}$, rather than $D_i^* \leq \frac{\sigma_i}{\rho_i}$. The problem, of course, is that we are ignoring strong dependencies among the queueing systems at the nodes in $P(i)$. Figure 3-4 illustrates the Additive Method.

3.6.2 Staggered Greedy Sessions for GPS Systems

In this section, we refine the bounds of Section 3.6.1 by analyzing the session route as a whole. For notational simplicity we focus on a particular session, i, that follows the route $1, 2, ..., K$. Figure 3-5 illustrates the system to be analyzed. We will assume that:

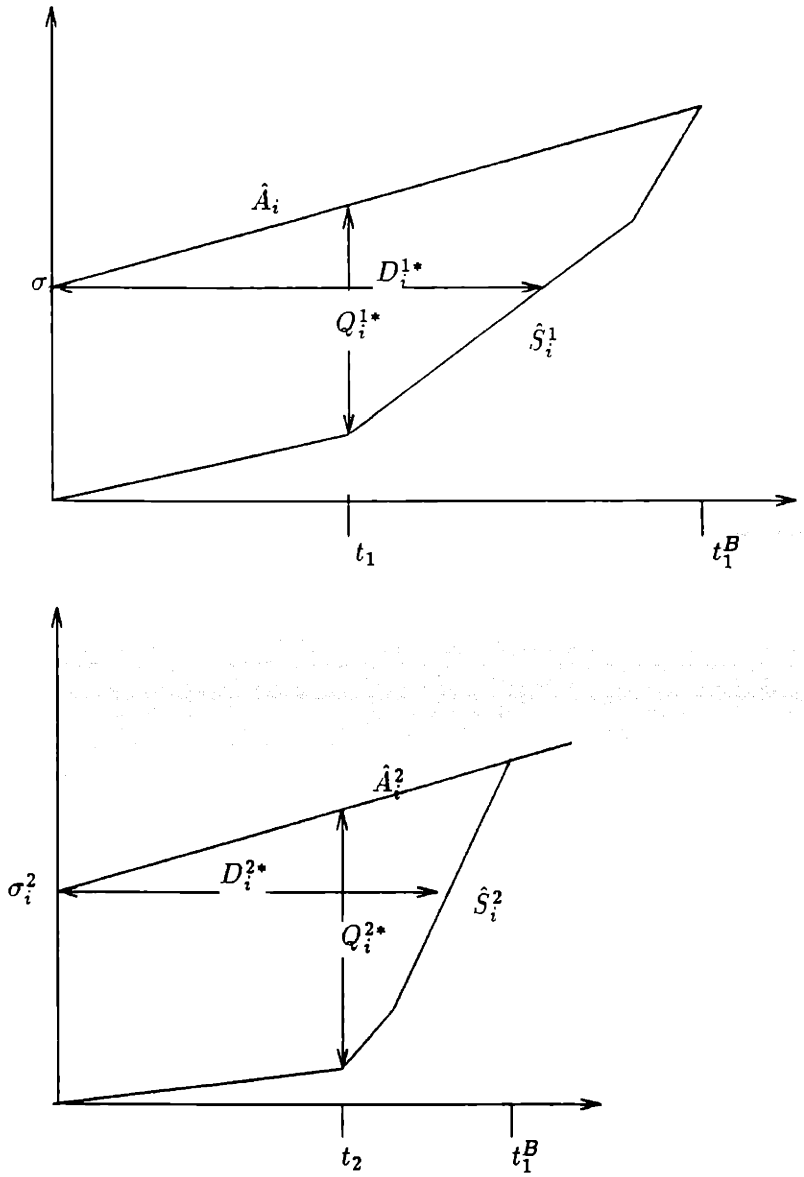
1. The sessions $j \in I(m) - \{i\}$ (for $m = 1, 2, ..., K$) are free to send traffic in any manner as long as $A_j^m \sim (\sigma_j^m, \rho_j, C_j^m)$, $C_j \geq r^m$. Thus it is appropriate to call the sessions in $I(m) - \{i\}$, the *independent sessions* at node m ($m = 1, 2, .., K$).

2. Session i traffic is constrained to flow along its route so that

$$A_i^m = S_i^{m-1} \quad m = 2, 3, ..., K.$$

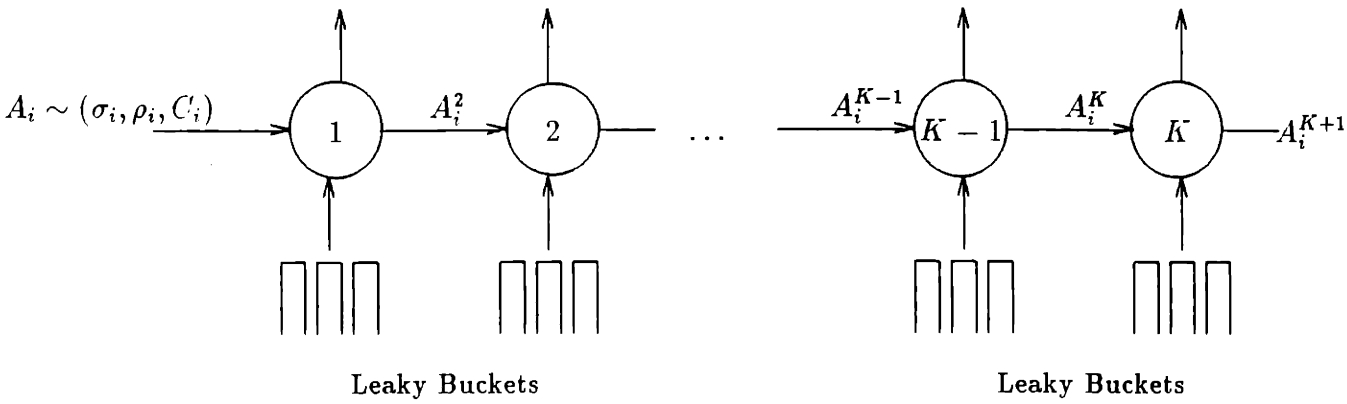
Assumptions 1 and 2 are collectively known as the independent sessions assumption. Note that while the network topology may preclude certain arrival functions of A_j^k that are consistent with $(\sigma_j^k, \rho_j, C_j^k)$, these functions are included under the independent sessions assumption. On the other hand, every arrival function allowable in the network, is allowed under the independent sessions assumption. Thus, the values of D_i^* and Q_i^* that hold under the independent sessions assumption, must be upper bounds on the true values of these quantities.

In this section, we provide an exact analysis of D_i^* and Q_i^* under the independence assumption.



The figures (a) and (b) can be determined independently. The Additive Method yields bounds $Q_i^* \leq Q_i^{1*} + Q_i^{2*}$ and $D_i^* \leq D_i^{1*} + D_i^{2*}$.

Figure 3-4: The Additive Method for session i when $P(i) = \{1, 2\}$.



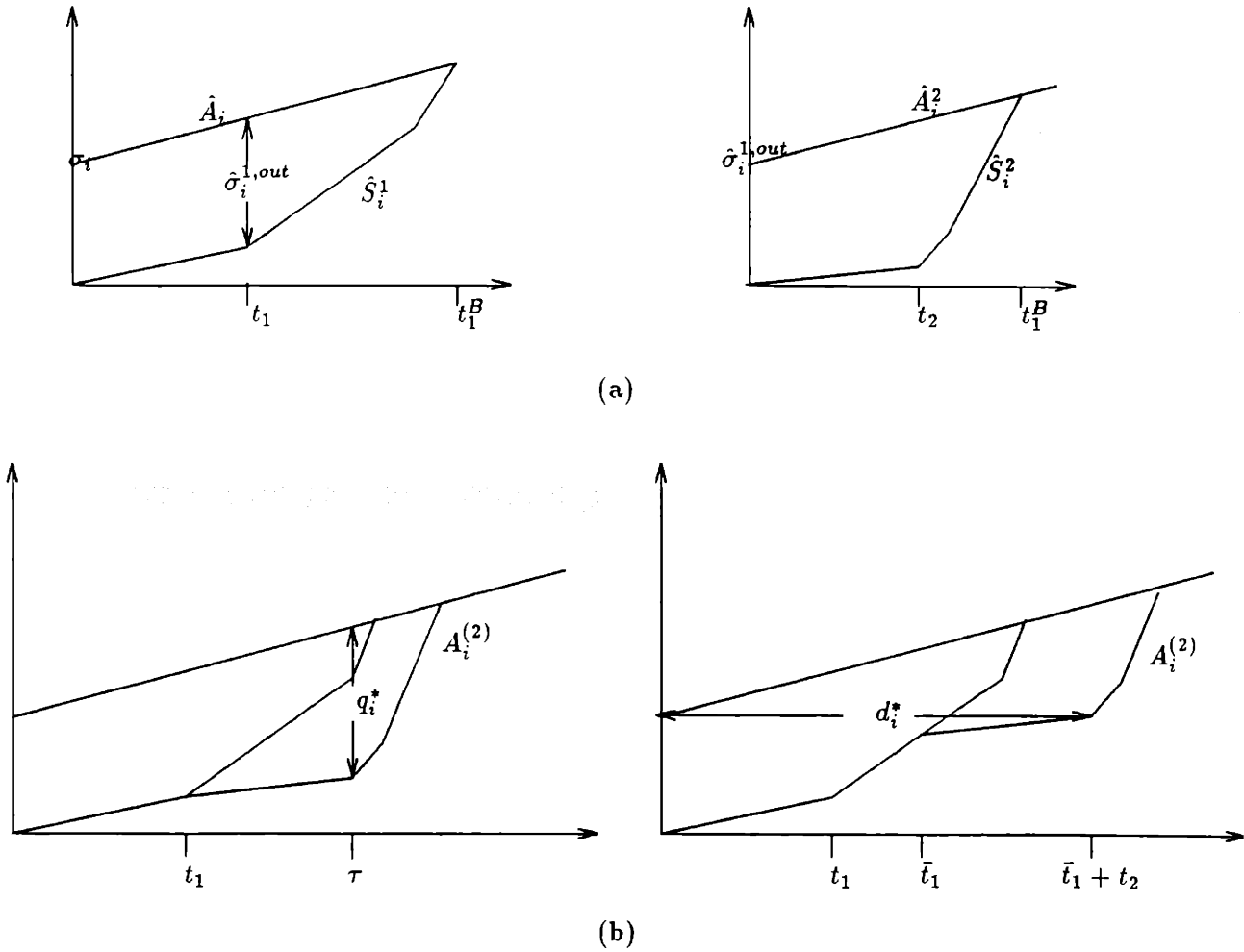
Session i traffic enters the network so that it is consistent with (σ_i, ρ_i, C_i), and $A_i^k = S_i^{k-1}$ for $k = 2, 3, ..., K$. The independent sessions at node k are free to send traffic in any manner as long as $A_j^k \sim (\sigma_j^k, \rho_j, C_j^k)$ for every session $j \in I(k) - \{i\}$, $k = 1, 2, ..., K$.

Figure 3-5: Analyzing the Session i route as a whole, under the Independent Sessions Assumption.

In view of our results for the single node case, it would be satisfying if maximum delay (and backlog) were achieved when all the sessions of the network are greedy starting at time zero (the beginning of a system busy period). However, this is not necessarily true. What is required is that the sessions at a particular node j become greedy simultaneously, but only after the sessions at node $j-1$ become greedy. We call this pattern of arrivals a staggered greedy regime. The instants of time at which the sessions become greedy depend on the session for which maximum delay and backlog is being estimated. We will also find that Q_i^* and D_i^* may not both be achieved for the *same* staggered greedy regime. This is illustrated in Figure 3-6. One of the goals of this section is to find (under the independent sessions assumption) a single curve from which both D_i^* and Q_i^* may be computed. In the single-node case this curve is just $\hat{S}_i$, i.e. recall Lemma 2.14 of of Chapter 2:

Lemma 2.14 *Suppose session i is busy in the interval $[\tau, t]$ in a single node system. Then*

$$S_i(\tau, t) \geq \hat{S}_i(0, t - \tau) \tag{3.15}$$



The curves $\hat{S}_i^1$ and $\hat{S}_i^2$ are shown in (a). Note that $\sigma_i^2 = \hat{\sigma}_i^{1,out}$, and so $\hat{S}_i^1$ and $\hat{S}_i^2$ cannot be determined independently.

Figure (b) shows two staggered greedy regimes. In the first, the sessions in $I(2) - \{i\}$ become greedy at time t_1, which yields a maximum backlog of q_i^* at time τ. In the second staggered greedy regime, the sessions at $I(2) = \{i\}$ wait until time $\bar{t}_1$ to become greedy—this results in a maximum delay of d_i^* for session i at time zero.

Figure 3-6: Two Staggered Greedy Regimes when $P(i) = \{1, 2\}$

The network analog of this Lemma will be established in Lemma 3.3.

The Session i Universal Service Curve

The possibility of D_i^* and Q_i^* being achieved under different staggered greedy regimes is discouraging from a practical standpoint, especially if computing either one of these quantities involves solving a complicated optimization problem. It would be much more desirable to have a single function from which both delay and backlog can be bounded. In this section we describe such a function, which we call the session i universal curve, $U_i(t)$. This curve is constructed without computing any staggered greedy regimes, and both D_i^* and Q_i^* can be determined exactly from it (under the independent sessions assumption). In addition, the staggered greedy regimes that achieve these worst-case values can also be efficiently determined from $U_i(t)$.

For notational simplicity, we will focus on a session i such that $P(i) = (1, 2, ..., K)$. The functions $\hat{S}_i^1, ..., \hat{S}_i^K$ can be computed using the internal traffic characterization of Section 3.5.3. We first describe how to construct U_i from $\hat{S}_i^1, ..., \hat{S}_i^K$, and then define the curve analytically. Finally, we establish the relationship between U_i and the session i departures from the network, $S_i^{(K)}$.

Recall that for each node $m = 1, 2, ..., K$, $\hat{S}_i^m$ is continuous, piece-wise linear and is convex-$\cup$ in the range $[0, t_m^B]$, where t_m^B is the duration of the session i busy period at m under the all-greedy regime. . Also $\hat{S}_i(0) = 0$. Thus it can be described (in the range $[0, t_m^B]$) by a list of pairs:

$$(s_1^m, d_1^m), (s_2^m, d_2^m), \ldots, (s_{n_m}^m, d_{n_m}^m),$$

where s_j^m is the slope of the j^{th} line segment and d_j^m is its duration. Clearly,

$$s_1^m < s_2^m < \ldots < s_{n_m}^m, \tag{3.16}$$

and

$$\sum_{j=1}^{n_m} d_j^m = t_m^B \tag{3.17}$$

Now let E_i^k be the collection of all the pairs (s_j^m, d_j^m) for $m = 1, 2, ..., k$—i.e.

$$E_i^k = \bigcup_{m=1}^{K} \bigcup_{j=1}^{n_m} \{(s_j^m, d_j^m)\}.$$

The session i universal service curve, U_i is defined as:

$$U_i(t) = \min\{G_i^K(t), \hat{A}_i(0,t)\},$$

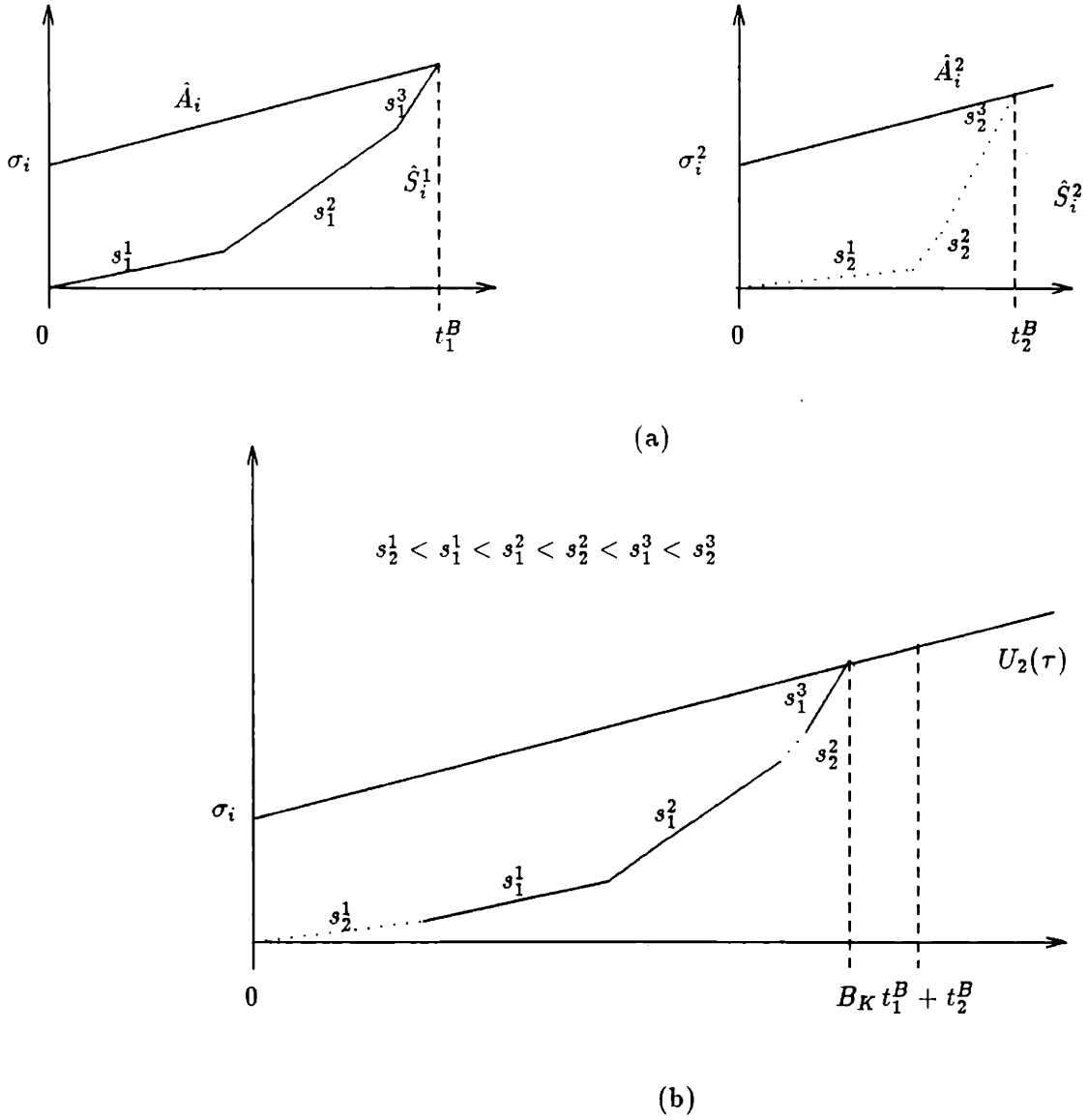
where the curve G_i^k (for $k = 1, 2, ..., K$) is a continuous curve constructed from the elements of E_i^k as follows:

1. Set $G_i^k(0) = 0$, Remaining-in-E $= E_i^k$; $Glist = \phi$; $u = 0$; $t = 0$.
2. Remove from Remaining-in-E an element of smallest slope: $e^{new} = (s^{new}, d^{new})$. Append Glist with e^{new}. If Remaining-in-E is not empty then repeat step 2.
3. G_i^k is defined in the range $[0, \sum_{m=1}^k t_m^B]$ by the convex-$\cup$ curve formed by $Glist$; for $t \geq \sum_{m=1}^k t_m^B$ set

$$G_i^k(t) = G_i^k(\sum_{m=1}^{k} t_m^B) + \hat{A}_i(\sum_{m=1}^{k} t_m^B, t). \tag{3.18}$$

Figure 3.6.2 illustrates the construction of U_i for a simple two node example. Note that:

- G_i^k is defined for $k = 1, 2, ..., K$, but U_i is defined in terms of G_i^K.
- For each m, the relative of order of the elements from $\hat{S}_i^m$ is preserved in $Glist$.
- We still have to show that for any network, the curve G_i^K always meets $\hat{A}_i$—this is established in Lemma 3.10.



The two service curves in (a) show $\hat{S}_i^1$ and $\hat{S}_i^2$. In (b) the line segments that make up these curves are concatenated to make a piece-wise linear convex curve that meets $\hat{A}_i$ at time B_K. Thus

$$U_i(t) = \begin{cases} G_i^K(t) & t \leq B_K \\ \hat{A}_i(0,t) & t > B_K. \end{cases}$$

Note that the line segment with slope s_2^3 is never used in the construction of U_2, i.e. $T_2 < t_1^B + t_2^B$.

Figure 3-7: An example of how U_i is constructed for $K = 2$.

Describing the construction of G_i^{IK} is useful in understanding its form, but we need an analytical definition of the curve in order to prove things about it. The following is a useful, notationally compact definition: For all $t \in [0, \sum_{m=1}^{k} t_m^B]$:

$$G_i^{Ik}(t) = \begin{cases} \hat{S}_i^1(0,t), & \text{for } k = 1 \\ \min_{\tau \in [0,t]}\{G_i^{k-1}(\tau) + \hat{S}_i^k(0, t-\tau)\}, \quad t - \tau \leq t_k^B, & \text{for } k \geq 2. \end{cases} \tag{3.19}$$

Suppose we are given $G_i^1 = \hat{S}_i^1(0,t)$, and wish to compute $G_i^2(t)$ for some $t \in [0, \sum_{m=1}^{2} t_m^B]$. Applying the algorithm construction of G_i^2, we determine $\hat{\tau}$, the duration of the elements picked from the list describing $\hat{S}_i^1$. Then $\hat{\tau}$ corresponds to the minimizing value of τ in (3.19). Generalizing this to larger values of k, we can rewrite (3.19) in terms of $\tau_1, ..., \tau_{k+1}$ such that $\tau_1 = 0$, $\tau_{k+1} = t$ and $\tau_{m+1} - \tau_m \leq t_m^B$ for each $m = 1, 2, .., k$ to yield:

$$\begin{aligned} G_i^k(t) &= \min_{\tau_k \in [0,t]} \min_{\tau_{k-1} \in [0,\tau_k]} \cdots \min_{\tau_2 \in [0,\tau_3]} \left\{ \sum_{m=1}^{k} \hat{S}_i^m(0, \tau_{m+1} - \tau_m) \right\} \\ &= \min_{0 \leq \tau_2 \leq \tau_3 \leq \cdots \tau_K \leq t} \sum_{m=1}^{k} \hat{S}_i^k(0, \tau_{m+1} - \tau_m). \end{aligned} \tag{3.20}$$

For each m, the quantity $\tau_{m+1} - \tau_m$ corresponds to the total duration of the elements picked for *Glist* from the list describing $\hat{S}_i^k$. Thus $G_i^k(t)$ is the curve described by *Glist*. Note that the minimizing values of $\tau_2, ..., \tau_k$ are functions of t.

In the next Lemma we show that $G_i^k(t)$ must meet $\hat{A}_i(0,t)$ at some time before $\sum_{m=1}^{k} t_m^B$:

Lemma 3.10

$$G_i^k(\sum_{m=1}^{k} t_m^B) \geq \hat{A}_i(0, \sum_{m=1}^{k} t_m^B).$$

Proof. By definition:

$$\tau_{m+1} - \tau_m \leq t_m^B.$$

for each $m = 1, 2, ..., k$. Since $t = \sum_{m=1}^{k} t_m^B$ we must have equality in each of these K

inequalities. Thus

$$\hat{S}_i^m(0, \tau_{m+1} - \tau_m) = \hat{S}_i^m(0, t_m^B) = \hat{A}_i(0, t_m^B)$$

(where the second equality follows from the definition of t_m^B), and

$$G_i^k(\sum_{m=1}^{k} t_m^B) = \sum_{m=1}^{k} \hat{S}_i^m(0, t_m^B) = \sum_{m=1}^{k} \hat{A}_i(0, t_m^B) \geq \hat{A}_i(0, \sum_{m=1}^{k} t_m^B).$$

□

Now observe from (3.18) that for any $t \geq \sum_{m=1}^{k} t_m^B$ we must have:

$$G_i^k(t) \geq \hat{A}_i(0, t). \tag{3.21}$$

Then there exists $B_k \leq \sum_{m=1}^{K} t_m^B$ such that

$$\begin{aligned} G_i^k(t) &< \hat{A}_i(0,t), \quad t < B_k \\ &= \hat{A}_i(0,t), \quad t = B_k \\ &\geq \hat{A}_i(0,t), \quad t \geq B_k. \end{aligned} \tag{3.22}$$

Thus,

$$U_i(t) = \begin{cases} G_i^K(t) & t \leq B_K \\ \hat{A}_i(0,t) & t > B_K. \end{cases} \tag{3.23}$$

Having defined U_i, we now relate it to the session i departures from the network. First, we state two important results that are crucial to the analysis that follows. The first Lemma establishes that if the independent sessions at a node m are greedy from time zero, then as long as session i remains busy in an interval $[0, \tau]$, the function S_i^m will be identical to $\hat{S}_i^m$ in this interval. Thus session i does not have to be greedy, just busy during the interval. The second Lemma states that if the independent sessions at a node m are quiet during the interval $[0, \tau]$ and then are greedy starting at τ,

then this behavior minimizes $S_i(\tau,t)$, the amount of service received by session i at node m from time τ on. The proofs of these Lemmas follow almost directly from our work in Chapter 2.

Lemma 3.11 *Suppose that t is contained in a session i busy period at node m that begins at time 0. Also, suppose that none of the independent sessions has sent any traffic before time 0, and that each is greedy starting at time zero. Then S_i^m is identical to $\hat{S}_i^m$ in the range $[0,t]$.*

Lemma 3.12 *Suppose the independent sessions assumption holds, and that time t is contained in a session i busy period at server m that starts at time $\tau \leq t$. Then $S_i^m(\tau,t)$ is minimized, for all $t \geq \tau$, when for every independent session p at node m:*

1. $A_p^m(0,\tau) = 0$.
2. *Session p is greedy from time τ.*

Proof. When the independent sessions behave according to conditions 1 and 2 of the Lemma:

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

from Lemma 3.11. Now using Lemma 2.14 we are done. □

In the next Lemma we establish the relationship between S_i^m and G_i^m:

Lemma 3.13 *Consider a given arrival function, A_i, and a given time τ such that $Q_i(\tau) = 0$. Then for each m, $1 \leq m \leq K$, each $t > \tau$:*

$$S_i^m(\tau,t) \geq \min_{V \in [\tau,t]} \{A_i(\tau,V) + G_i^m(t-V)\}. \tag{3.24}$$

In the next section we are going to show that $G_i^m(t)$ is the amount of service given to session i under a specific staggered greedy regime called the (m,t)-staggered greedy regime. Thus Lemma 3.13 shows that the service to session i is minimized when a

such a staggered greedy regime is delayed by an appropriate amount, which is the minimizing value of V.

Proof. For $m = 1$, (3.24) states that

$$S_i^1(\tau, t) \geq \min_{V \in [\tau, t]} \{A_i(\tau, V) + \hat{S}_i^1(0, t - V)\}.$$

Choosing V to be last time in the interval $[\tau, t]$ that session i begins a busy period at node 1:

$$\begin{aligned} S_i^1(\tau, t) &\geq A_i(\tau, V) + \hat{S}_i(0, t - V) \\ &\geq \min_{V \in [\tau, t]} \{A_i(\tau, V) + \hat{S}_i^1(0, t - V)\}. \end{aligned} \tag{3.25}$$

Now assume the result for nodes $1, 2, ..., m - 1$. Then, letting t_m be the last time in the interval $[\tau, t]$ that session i is in a busy period at node m:

$$S_i^m(\tau, t) = S_i^{m-1}(\tau, t_m) + S_i^m(t_m, t) \tag{3.26}$$

By the induction hypothesis:

$$S_i^{m-1}(\tau, t_m) \geq \min_{V \in [\tau, t_m]} \{A_i(\tau, V) + G_i^{m-1}(t_m - V)\}. \tag{3.27}$$

Also, from Lemma 3.12:

$$S_i^m(t_m, t) \geq \hat{S}_i^m(0, t - t_m). \tag{3.28}$$

Substituting (3.27) and (3.28) into (3.26):

$$S_i^m(\tau, t) \geq \min_{V \in [\tau, t_m]} \{A_i(\tau, V) + G_i^{m-1}(t_m - V)\} + \hat{S}_i^m(0, t - t_m) \tag{3.29}$$

$$\geq \min_{V \in [\tau, t_m]} \{A_i(\tau, V) + G_i^{m-1}(t_m - V) + \hat{S}_i^m(0, t - t_m)\} \tag{3.30}$$

$$\geq \min_{V \in [\tau, t_m]} \{A_i(\tau, V) + G_i^m(t - V)\} \tag{3.31}$$

$$\geq \min_{V\in[\tau,t]}\{A_i(\tau,V)+G_i^m(t-V),\} \tag{3.32}$$

where the inequality in (3.31) follows from the definition of G_i^m in (3.19). □

Now we may state the major result of this section:

Theorem 3.3 *For every session i:*

$$Q_i^* \leq \max_{\tau\geq 0}\{\hat{A}_i(0,\tau)-G_i^K(\tau)\}, \tag{3.33}$$

and

$$D_i^* \leq \max_{\tau\geq 0}\left\{\min\{t: G_i^k(t)=\hat{A}_i(0,\tau)\}-\tau\right\}. \tag{3.34}$$

Proof. We first show (3.33): For some given set of arrival functions $A_1,...,A_N$:

$$Q_i(t) = A_i(0,t) - S_i^K(0,t).$$

From Lemma 3.13,

$$\begin{aligned} Q_i(t) &\leq A_i(0,t) - \min_{V\in[0,t]}\{A_i(0,V)+G_i^K(t-V)\} && (3.35)\\ &\leq A_i(0,t) - A_i(0,V_{\min}) + G_i^K(t-V_{\min}) && (3.36)\end{aligned}$$

where $V_{\min}$ is the minimizing value of V. Thus

$$\begin{aligned} Q_i(t) &\leq A_i(V_{\min},t) - G_i^K(t-V_{\min}) && (3.37)\\ &\leq \hat{A}_i(0,t-V_{\min}) - G_i^K(t-V_{\min}) && (3.38)\\ &\leq \max_{\tau\geq 0}\{\hat{A}_i(0,\tau)-G_i^K(\tau)\}, && (3.39)\end{aligned}$$

and (3.33) follows.

Next we show (3.34): For a given set of arrival functions, $A_1,...,A_N$ and $t\geq 0$, we

have from Lemma 3.13:

$$S_i^K(0,t) \geq \min_{V\in[0,t]}\{A_i(0,V) + G_i^K(t-V)\}.$$

Thus, for all $\hat{t} \geq 0$:

$$\begin{aligned}
D_i(\hat{t}) &= \min\left\{t : S_i^K(0,t) = A_i(0,\hat{t})\right\} - \hat{t} && (3.40)\\
&\leq \min\left\{t : \min_{V\in[0,t]}\{A_i(0,V) + G_i^K(t-V)\} = A_i(0,\hat{t})\right\} - \hat{t} && (3.41)\\
&\leq \min\left\{t : A_i(0,V_{\min}) + G_i^K(t-V_{\min}) = A_i(0,\hat{t})\right\} - \hat{t} && (3.42)\\
&\leq \min\left\{t : G_i^K(t-V_{\min}) = A_i(V_{\min},\hat{t})\right\} - \hat{t} && (3.43)\\
&\leq \min\left\{t : G_i^K(t) = A_i(V_{\min},\hat{t})\right\} + V_{\min} - \hat{t} && (3.44)\\
&\leq \min\left\{t : G_i^K(t) = \hat{A}_i(0,\hat{t}-V_{\min})\right\} + V_{\min} - \hat{t} && (3.45)\\
&\leq \min\left\{t : G_i^K(t) = \hat{A}_i(0,\hat{t}-V_{\min})\right\} - (\hat{t}-V_{\min}) && (3.46)\\
&\leq \max_{\tau\geq 0}\left\{\min\{t : G_i^K(t) = \hat{A}_i(0,\tau)\} - \tau\right\}. && (3.47)
\end{aligned}$$

In (3.42) we choose the *smallest* minimizing value of V. Then $V_{\min} \leq \hat{t}$, since $G_i^K(t - V_{\min}) \geq 0$. □

The inequalities (3.33) and (3.34) illustrate the importance of the universal curve. We will show in the next section, that under the independent sessions assumption, these inequalities are satisfied with equality for (K,t)-staggered greedy regimes.

The (K,t)-Staggered Greedy Regime

In this section we make clear the relationship between staggered greedy regimes and the session i universal curve U_i. As in the previous sections, we will focus on staggered greedy regimes with respect to a session i, and assume that $P(i) = \{1, 2, ..., K\}$.

Any staggered greedy regime can be characterized by a vector

$$(T_1, ..., T_K), \quad T_1 \leq T_2 \leq \ldots \leq T_K$$

such that all the sessions at node 1 are simultaneously greedy starting at time T_1, and the independent sessions at node j do not send any traffic in the interval $[T_1, T_j)$, but are simultaneously greedy starting at time T_j. Observe that the first staggered greedy regime in Figure 3-6(b) can be characterized by $(0, t_1)$ and the second by $(0, \bar{t}_1)$.

A (K,t)-staggered greedy regime, $t \leq B_K$, is the staggered greedy regime characterized by $(0, T_2, ..., T_K)$ such that

$$\sum_{k=1}^{K} \hat{S}_i^k(0, T_{k+1} - T_k) = G_i^K(t) \tag{3.48}$$

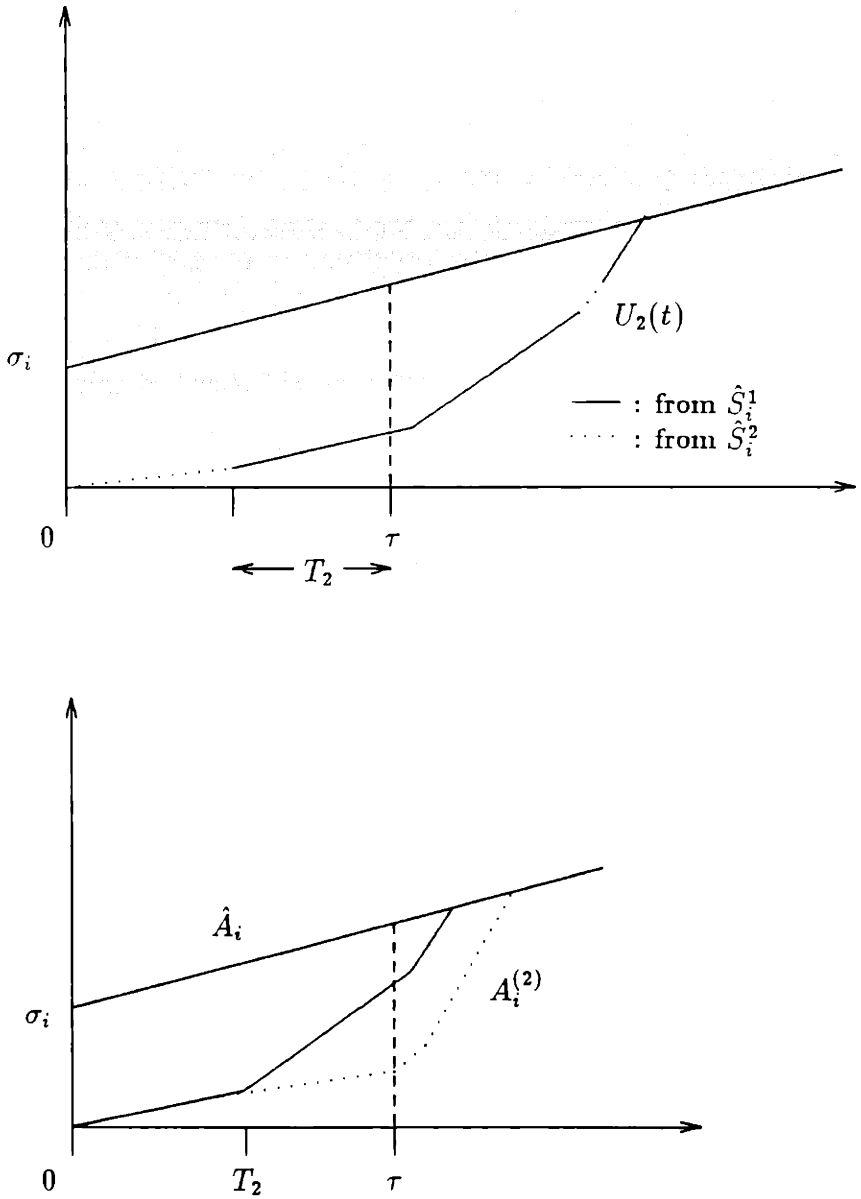
where $T_1 = 0$, $T_{K+1} = t$ and $T_{k+1} - T_k \leq t_k^B$ for $k = 1, 2, ..., K$.
Note that

- Since $t \leq B_K$, $G_i^K(t) = U_i(t)$.
- Comparing (3.48) with (3.20) it is clear that $(T_2, ..., T_K)$ is a minimizing vector in (3.20).
- For each $k = 1, 2, ..., K-1$ the staggered greedy regime defined by $(0, T_2, ..., T_k)$ describes a (k, T_{k+1})-staggered greedy regime.

Figure 3-8 illustrates the construction of a (K,t)-staggered greedy regime for the simple case of $K = 2$. Notice from the figure that in the range $[0, T_2]$, $S_i^{(2)}$ is comprised of the line segments belonging to $\hat{S}_i^1, \ldots, \hat{S}_i^{K-1}$ that make up the universal curve in the range $[0, t]$. Also,

$$S_i^{(2)}(0, \tau) = U_i(\tau).$$

The next Lemma establishes this relationship for any (K,t)-staggered greedy regime.



The top figure the curve U_2 that was constructed from $\hat{S}_i^1$ and $\hat{S}_i^2$. In order to find the $(2,\tau)$-staggered greedy regime, add the durations of the line segments taken from $\hat{S}_1$ that are in $U_2(t)$, $t \leq \tau$. This sum is T_2, the time that the independent sessions at node 2 become greedy. This characterizes the staggered greedy regime which is shown in the bottom figure.

Figure 3-8: Computing a (k,t)-Staggered Greedy Regime when $P(i) = \{1,2\}$

Lemma 3.14 *Given any (K,t)-staggered greedy regime characterized by $(0,T_2,...,T_K)$, $t \leq B_K$: For each node $k = 1,2,...,K$:*
For each $j = 1,2,...,k-1$, and $\tau \in [T_j, T_{j+1}]$:

$$A_i^k(0,\tau) = (\sum_{m=1}^{j-1} \hat{S}_i^m(0, T_{m+1} - T_m)) + \hat{S}_i^j(0, \tau - T_j), \tag{3.49}$$

and for $\tau > T_k$:

$$S_i^{(k)}(0,\tau) = \min\{\hat{A}_i(0,\tau), \sum_{m=1}^{k-1} \hat{S}_i^m(0, T_{m+1} - T_m) + \hat{S}_i^k(0, \tau - T_k)\}. \tag{3.50}$$

Proof. We proceed by induction on k: For $k = 1$ only (3.50) applies. Since $S_i^1 = \hat{S}_i^1$ the basis step is shown. Now assume the result for nodes $1,2,...,k-1$. Then by induction hypothesis and the since $(0,T_2,...,T_k)$ is a (k,T_{k+1})-staggered greedy regime, we have for $\tau \geq T_k$:

$$S_i^{(k)}(0,\tau) = \min\{\hat{A}_i(0,\tau), \sum_{m=1}^{k-1} \hat{S}_i^m(0, T_{m+1} - T_m) + \hat{S}_i^k(0, \tau - T_k)\} - Q_i^k(T_k). \tag{3.51}$$

Now if $Q_i^k(\tau) = 0$ for all $\tau \leq T_k$ we are done by (3.51) and the induction hypothesis. So let us assume that $Q_i^k(\tau) > 0$ for some $\tau \leq T_k$. Since the independent sessions at k are quiet during the interval $[0,T_k]$ it follows that there is at least one interval before T_k during which $S_i^{(k-1)}$ has slope greater than r_k (where r_k is the rate of k). Since the slope of $\hat{S}_i^k(0,t)$ is strictly less than r_k for $t \in [0, t_k^B]$, we have from minimization of (3.20) that $T_{k+1} - T_k = t_k^B$. Since no node k busy period can be longer than t_k^B time units, it follows that $Q_i^k(T_{k+1}) = 0$. Thus

$$S_i^{(k)}(0,T_{k+1}) = \hat{A}_i(0,T_{k+1}) = G_i^k(T_{k+1}) - Q_i^k(T_k),$$

where the first equality is from the induction hypothesis, and the second from (3.51).

Then $G_i^k(T_{k+1}) \geq \hat{A}_i(0, T_{k+1})$, and

$$T_{k+1} = B_k.$$

Now, let $[a, a+\Delta]$, such that $\Delta > 0$ and $a+\Delta \leq T_k - a$, be an interval during which $S_i^{(k-1)}$ has largest slope, and such that this slope belong to a single node, $j < k$. Note that the slope of $S_i^{(k-1)}$ during this interval must be greater than r_k, since $Q_i^k(\tau) > 0$ for some $\tau < T_k$. Then the staggered greedy regime characterized by

$$\hat{T} = (0, T_2, ..., T_j - \Delta, T_{j+1} - \Delta, ..., T_k - \Delta)$$

is a $(k, T_{k+1} - \Delta)$-staggered greedy regime. I.e.,

$$\sum_{m=1}^{k} \hat{S}_i^m(0, \hat{T}_{m+1} - \hat{T}_m) = G_i^k(t - \Delta), \tag{3.52}$$

where $\hat{T}_{k+1} = T_{k+1} - \Delta$. Now since $\hat{T}_{k+1} - \hat{T}_k = t_k^B$, it follows from similar reasoning as above that under $\hat{T}$, session i is not backlogged at k at time $\hat{T}_{k+1}$, and that therefore

$$\hat{T}_{k+1} = B_k.$$

Thus

$$\hat{T}_{k+1} = T_{k+1} - \Delta = B_k.$$

But this implies that $\Delta = 0$, which is a contradiction. □

The following Theorem follows easily from Lemma 3.14

Theorem 3.4 *For any (K, t)-staggered greedy regime:*

$$S_i^{(K)}(0, t) = G_i^K(t).$$

Proof. Pick $\tau = t > T_k$ in Lemma 3.14. Then (3.50) applies, and since $t \leq B_k$ the

result follows. □

It is easy to construct any (K,t)-staggered regime given U_i. Figure 3-9 shows how to construct the staggered greedy regimes that maximize backlog and delay. From Theorems 3.3 and 3.4 we have the main theorem of this section:

Theorem 3.5 *Under the independent sessions assumption, both D_i^* and Q_i^* are achieved under a (K,t)-staggered greedy regime.*

Now since the values of D_i^* and Q_i^* achieved under the independent sessions assumption are upper bounds to the actual values of these quantities, we are have shown how to find upper bounds on session backlog and delay.

3.6.3 Calculating bounds on D_i^* and Q_i^*

Let t_q be the time at which the backlog under the universal session i service curve is maximized:

$$\hat{A}_i(0,t_q) - U_i(t_q) = \max_{\tau}(\hat{A}_i(0,\tau) - U_i(\tau)).$$

We first compute the bounds on delay and backlog when $C_i = \infty$

Corollary 3.1 *If $C_i = \infty$ then under the independent sessions assumption:*

$$Q_i^* = \sigma_i + \rho_i t_q - U_i(t_q).$$

Corollary 3.2 *If $C_i = \infty$, then under the independent sessions assumption:*

$$D_i^* = \begin{cases} \frac{Q_i^*}{\rho_i} = \frac{\sigma_i + \rho_i t_q - U_i(t_q)}{\rho_i} & \text{if } U_i(t_q) \geq \sigma_i \\ \hat{t}, \text{ where } U_K(\hat{t}) = \sigma_i & \text{if } U_i(t_q) < \sigma_i. \end{cases} \tag{3.53}$$

Since an infinite capacity link can always simulate a finite capacity link, worst case session i backlog and delay calculated under this assumption must upper bound the actual values of these quantities. As we have explained earlier, the bounds obtained

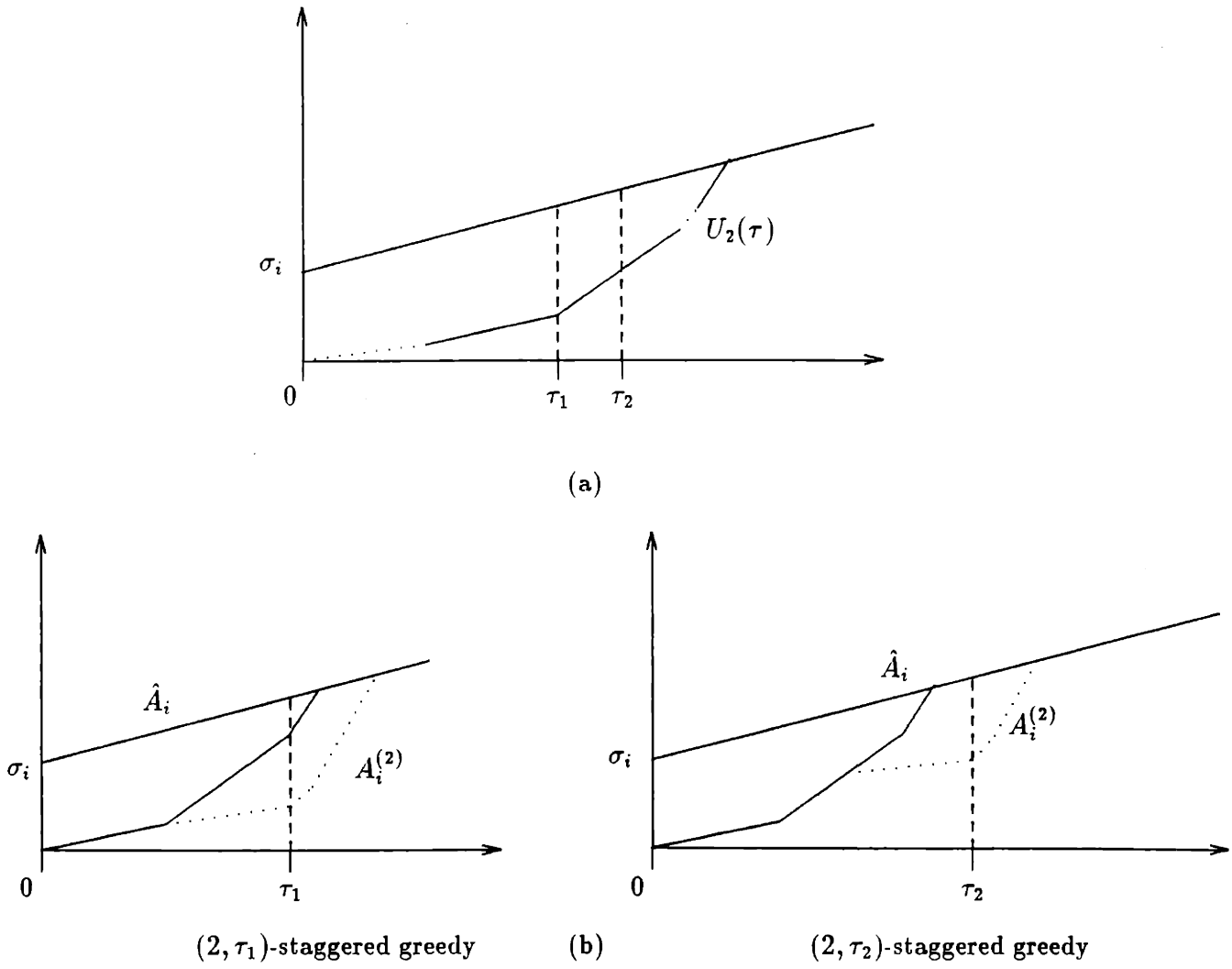


Figure (a) shows the session i universal curve. Notice that for this curve "backlog" is maximized at time τ_1 and "delay" is maximized at time τ_2. Figure (b) shows the two staggered greedy regimes corresponding to these times. Notice that the backlog at time τ_1 in the first regime is exactly equal to the "backlog" at time τ_1 in (a), and similarly the delay at time τ_2 in the second regime is exactly equal to the "delay" at that time in (b).

Figure 3-9: The staggered greedy regimes that maximize backlog and delay under the independent sessions assumption.

in these Corollaries hold even when the independent sessions assumption does not hold, and when $C_i < \infty$.

Note that for a RPPS system, $t_q = 0$ so that Corollary 3.1 yields:

$$Q_i^* \leq \sigma_i - U_i(0) = \sigma_i, \tag{3.54}$$

which is exactly what we derived in Theorem 3.1 in our study of RPPS networks. Also, the bound

$$D_i^* \leq \frac{\sigma_i}{\rho_i} u_i^{max}$$

which we obtained Theorem 3.1 follows in exactly the same manner from (3.54).

We now present a procedure for obtaining bounds on D_i^* and Q_i^* that are in general, more tight than those obtained by the Corollaries 3.1 and 3.2: Now let

$$b_i = \frac{\sigma_i}{C_i - \rho_i}.$$

This is the maximum amount of time that session i traffic can arrive at node 1 at a rate of C_i (without violating (1.11)).

1. Compute $\sigma_i^{(k)}$ and $\hat{S}_i^k$ for $k = 1, 2, ..., K$.

2. Compute the curve U_i and t_q.

3. If $U_i(t_q) > \hat{A}_i(0, b_i)$ then

   $Q_i^* \leq \hat{A}_i(0, t_q) - U_i(t_q)$

   $D_i^* \leq t_q - t^*$, where $\hat{A}_i(0, t^*) = U_i(t_q)$

   else

   $\tau^* = \max\{b_i, t_q\}$.

   $Q_i^* \leq \hat{A}_i(0, \tau^*) - U_i(0, \tau^*)$

   $D_i^* \leq \hat{t} - \tau^*$, where $U_i(\hat{t}) = \hat{A}_i(0, \tau^*)$.

3.7 Propagation Delay

It is easy to incorporate deterministic propagation delays into our network framework: Suppose that every bit transmitted on link (i,j), incurs a delay of $d_{l,m}$ time units. Then each link acts as a constant delay element, and the characterization of internal traffic (using the method of Section 3.5.3) remains the same. A natural modification of the independent sessions assumption allows us to bound end-to-end delay as well: Consider a session i such that $P(i) = 1, 2, ..., K$: Also, let $d_{0,1}$ be the propagation delay on the access link. Then

1. The independent sessions at node m, $j \in I(m) - \{i\}$ (for $m = 1, 2, ..., K$) are free to send traffic in any manner as long as $A_j^m \sim (\sigma_j^m, \rho_j, C_j^m)$, $C_j^m \geq r^m$.

2. Session i traffic is constrained to flow along its route so that

$$A_i^m(\tau, t) = S_i^{m-1}(\tau - d_{m-1,m}, t - d_{m-1,m}) \qquad m = 2, 3, ..., K.$$

In view of the analysis of Section 3.6, it can be seen that

$$D_i^* \leq \sum_{m=1}^{K} d_{m-1,m} + D_i^{*,\text{noprop}}$$

where $D_i^{*,\text{noprop}}$ is the the worst-case session i delay computed for the same characterization of internal traffic when propagation delays are zero. The number of bits in "flight" on a link (l,m) is at most

$$q_{l,m} = C_l d_{l,m}. \tag{3.55}$$

Thus

$$Q_i^* \leq \sum_{m=1}^{K} q_{m-1,m} + Q_i^{*,\text{noprop}}.$$

3.8 Conclusions

In this chapter we described techniques for deriving bounds on worst-case session delay and backlog, for broad classes of GPS networks. For Rate Proportional Processor Sharing and its variants, we were able to give succinct close-form expressions for these bounds. A two step procedure enabled us to deal with more general assignments: In the first step the internal traffic was characterized in terms of leaky bucket parameters, and in the second step the session route was analyzed for worst-case delay and backlog. We found that efficient algorithms exist to perform both of these steps.

There are two major areas for future work. First, a better understanding of stability issues in work conserving networks (so that issues such as Conjecture 3.1 are resolved) would help in designing assignment schemes that are more general than CRST. We will discuss this issue further in Chapter 5. Second, the algorithms presented in this chapter can probably be made more efficient, and may also be conducive to parallelization.

Chapter 4

Arbitrary Topology PGPS Networks

In this chapter we extend the analysis of arbitrary topology GPS networks to PGPS networks, thereby completing the final phase of this thesis. An important intermediate step is to incorporate packet lengths into the GPS network analysis of Chapter 3—we do this Section 4.1. In Section 4.2, we give our results for PGPS networks, and in Section 4.3 these results are interpreted for the important case of Rate-Proportional Processor Sharing networks. Conclusions are in Section 4.4.

4.1 GPS Networks with Non-negligible Packet Sizes

The analysis in Chapter 3 dealt with GPS networks in which the packet lengths are negligible—i.e., we assumed that traffic is perfectly pipelined throughout the network. However, for most networks with heterogeneous link speeds, it is desirable for the nodes to postpone the transmission of packets until they have *completely arrived.* Thus, if $m-1$ and m are successive nodes on a session i's route, we cannot assume,

as we did in Chapter 3, that $S_i^{m-1} = A_i^m$. In fact, for $P(i) = \{1, 2, ..., K_i\}$:

$$S_i^{m-1}(0,t) \geq A_i^m(0,t) \geq S_i^{m-1}(0,t) - L_i, \quad m = 2, ..., K_i.$$

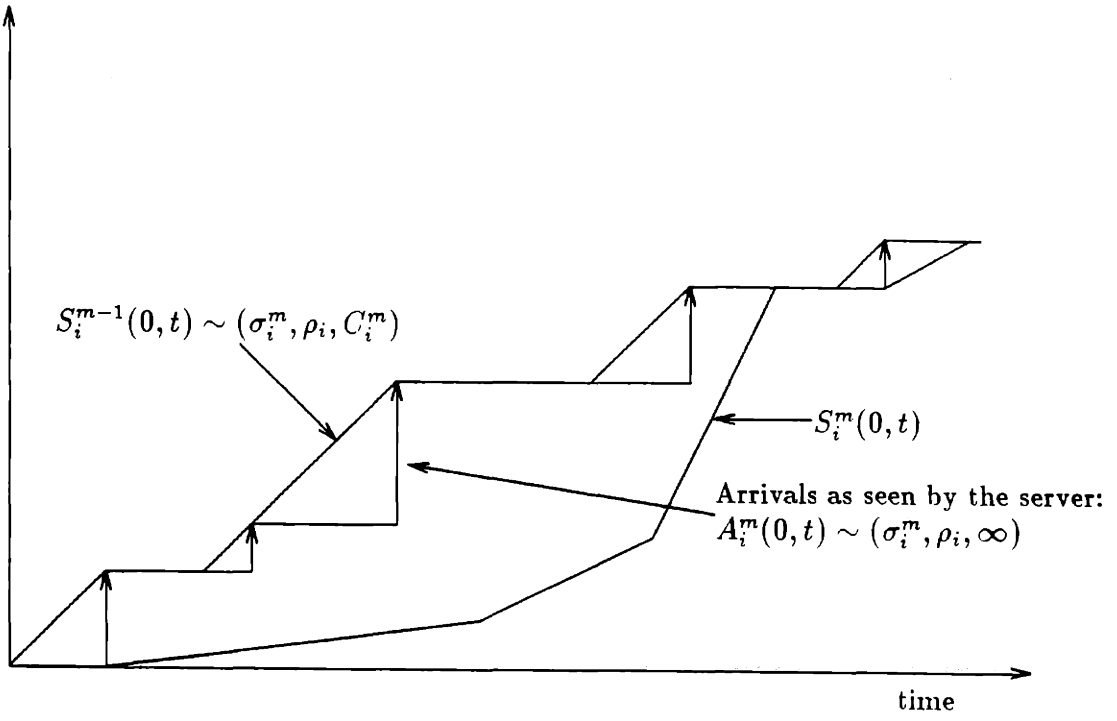
This important difference notwithstanding, we will still find the results for $L_i = 0$, to be very useful in the more general case of non-negligible packet sizes. It may seem strange to impose this constraint on the packet length while maintaining the abstraction of processor sharing, but we shall see that this intermediate step leads us to a result for PGPS.

Assume that the maximum packet size for each session i is $L_i \leq \sigma_i$. Since the GPS server does not begin serving a packet until its last bit has arrived, it "sees" the arrivals as a series of impulses, such that the height of each impulse is at most L_i. Thus A_i^m, is consistent with $(\sigma_i^m, \rho_i, \infty)$ rather than with $(\sigma_i^m, \rho_i, C_i^m)$ (see Figure 4.1). Similarly, the arrivals seen by the server from every other session, j, at node m, are consistent with $(\sigma_j^m, \rho_j, \infty)$. Thus, as long as we assume that $C_j^m = \infty$ for every session $j \in I(m)$, we can use the results of Chapter 2 to bound worst-case delay and backlog.

To analyze networks of such GPS servers, we follow the same steps as we did in Chapter 3—we first characterize the internal traffic in terms of leaky bucket parameters, and then bound the worst-case delay and backlog for each session by analyzing its route as a whole. There are two points of departure from the analysis of Chapter 3:

(A) When using the *all-greedy bound* (Section 3.5.1) at any node m, assume that $C_j^m = \infty$, for all $j \in I(m)$.

Similarly, when using the *independent sessions assumption*, assume that $C_j = \infty$ for each independent session, j, at node m (Section 3.6.2).



$A_i^m(0,t)$ represents the cumulative arrivals seen by server, m. The length of each impulse of $\mathcal{A}_i(0,t)$ is bounded by L_i, the maximum packet size for session i. Since $L_i \leq \sigma_i^m$, it can be seen from the figure that $A_i^m \sim (\sigma_i^m, \rho_i, \infty)$.

Figure 4-1: A GPS Server when the packet sizes are non-negligible.

(B) Given that $P(i) = \{1, 2, ..., K_i\}$,

$$S_i^{m-1}(\tau, t) \geq A_i^m(\tau, t) \geq S_i^{m-1}(\tau, t) - L_i, \quad m = 2, ..., K_i, \quad \tau < t. \tag{4.1}$$

Consider a GPS network with CRST assignments. The internal traffic can be characterized using the same procedure as in 3.5.3, applying condition (A) to compute the all-greedy bounds.

To analyze the session i route given internal characterization of the traffic, we proceed as follows: Define $\hat{S}_i^m$ to be the session i output at node m under the all-greedy regime (again applying condition (A)). Then the session i universal service curve is computed as it was in Chapter 3. Note that Lemma 3.11 also holds:

Lemma 4.1 *Suppose that t is contained in a session i busy period at node m that begins at time 0. Also, suppose that none of the independent sessions has sent any traffic before time 0, and that each is greedy starting at time zero. Then S_i^m is identical to $\hat{S}_i^m$ in the range $[0, t]$.*

However, in order to bound session i delay and backlog, we need to modify Lemma 3.13 and Theorem 3.3, in order to incorporate condition (B):

In what follows we assume (for notational simplicity) that $P(i) = \{1, 2, ..., K\}$:

Lemma 4.2 *Consider some time τ such that $Q_i(\tau) = 0$. Then for each m, $1 \leq m \leq K$, each $t > \tau$:*

$$S_i^m(\tau, t) \geq \min_{V \in [\tau, t]} \{A_i(\tau, V) + G_i^m(0, t - V)\} - mL_i. \tag{4.2}$$

Proof. For $m = 1$, (4.2) states that

$$S_i^1(\tau, t) \geq \min_{V \in [\tau, t]} \{A_i(\tau, V) + \hat{S}_i^1(0, t - V)\} - L_i.$$

Choosing V to be last time in the interval $[\tau, t]$ that session i begins a busy period at

node 1:

$$\begin{aligned} S_i^1(\tau,t) &\geq A_i^1(\tau,V) + \hat{S}_i^1(0,t-V) && (4.3)\\ &\geq A_i(\tau,V) - L_i + \hat{S}_i^1(0,t-V) \\ &\geq \min_{V\in[\tau,t]}\{A_i(\tau,V) + \hat{S}_i^1(0,t-V) - L_i\}. && (4.4)\end{aligned}$$

Now assume the result for nodes $1,2,...,m-1$. Then, letting t_m be the last time in the interval $[\tau,t]$ that session i begins a busy period at node m:

$$S_i^m(\tau,t) = A_i^m(\tau,t_m) + S_i^m(t_m,t).$$

From (4.1):

$$S_i^m(\tau,t) \geq S_i^{m-1}(\tau,t_m) - L_i + S_i^m(t_m,t). \quad (4.5)$$

By the induction hypothesis:

$$S_i^{m-1}(\tau,t_m) \geq \min_{V\in[\tau,t_m]}\{A_i(\tau,V) + G_i^{m-1}(t_m-V) - (m-1)L_i\}. \quad (4.6)$$

Also, from Lemma 4.1:

$$S_i^m(t_m,t) \geq \hat{S}_i^m(0,t-t_m). \quad (4.7)$$

Substituting (4.6) and (4.7) into (4.5):

$$\begin{aligned} S_i^m(\tau,t) + mL_i &\geq \min_{V\in[\tau,t_m]}\{A_i(\tau,V) + G_i^{m-1}(t_m-V)\} + \hat{S}_i^m(0,t-t_m) && (4.8)\\ &\geq \min_{V\in[\tau,t_m]}\{A_i(\tau,V) + G_i^{m-1}(t_m-V) + \hat{S}_i^m(0,t-t_m)\} && (4.9)\\ &\geq \min_{V\in[\tau,t_m]}\{A_i(\tau,V) + G_i^m(t-V)\} && (4.10)\\ &\geq \min_{V\in[\tau,t]}\{A_i(\tau,V) + G_i^m(t-V),\} && (4.11)\end{aligned}$$

where the inequality in (4.10) follows from the definition of G_i^m in (3.19). □

Theorem 4.1 *For every session i:*

$$Q_i^* \leq \max_{\tau \geq 0}\{\hat{A}_i(0,\tau) - G_i^K(\tau)\} + KL_i. \tag{4.12}$$

Proof. For some given set of arrival functions $A_1, ..., A_N$:

$$Q_i(t) = A_i(0,t) - S_i^{(K)}(0,t).$$

From Lemma 4.2,

$$\begin{aligned} Q_i(t) - KL_i &\leq A_i(0,t) - \min_{V \in [0,t]}\{A_i(0,V) + G_i^K(t-V)\} && (4.13)\\ &\leq A_i(0,t) - A_i(0,V_{\min}) + G_i^K(t - V_{\min}) && (4.14) \end{aligned}$$

where $V_{\min}$ is the minimizing value of V. Thus

$$\begin{aligned} Q_i(t) - KL_i &\leq A_i(V_{\min},t) - G_i^K(t - V_{\min}) && (4.15)\\ &\leq \hat{A}_i(0, t - V_{\min}) - G_i^K(t - V_{\min}) && (4.16)\\ &\leq \max_{\tau \geq 0}\{\hat{A}_i(0,\tau) - G_i^K(\tau)\}, && (4.17) \end{aligned}$$

and (4.12) follows. □

Having bounded the worst-case backlog, we turn to delay. Here we need a refinement of Lemma 4.2, where we restrict our values of t to be such that a session i packet, p_m, departs node m at time t. Given such a time t, let the corresponding packet arrive at time a_m, i.e.,

$$A_i(0,a_m) = S_i^m(0,t). \tag{4.18}$$

Note that a_m is also the time that packet arrives at node 1, i.e.,

$$A_i^1(0,a_m) = A_i(0,a_m) \tag{4.19}$$

We will now show that for each node m, $1 \leq m \leq K$:

$$S_i^m(\tau,t) \geq \min_{V\in[\tau,a_m]}\{A_i(\tau,V) + G_i^m(0,t-V)\} - (m-1)L_i. \tag{4.20}$$

This can be proven along same lines as Lemma 4.2. Note that the last session i busy period at node m before t cannot start at time strictly greater than the time that p_m arrives at node m.

Now using induction on m, we note that for $m = 1$, we are done by (4.19) and (4.3). In the inductive step, proceed as in the proof of Lemma 4.2 until (4.5). Inequality (4.7) holds as well. Now by the induction hypothesis:

$$S_i^{m-1}(\tau,t_m) \geq \min_{V\in[\tau,a_{m-1}]}\{A_i(\tau,V) + G_i^{m-1}(0,t-V)\} - (m-2)L_i, \tag{4.21}$$

where t_m is the start of the last session i busy period at node m in the interval $[\tau,t]$, and a_{m-1} is the arrival time of the packet (call it p_{m-1}) that departs node $m-1$ at time t_m. Note that such a packet must exist for a busy period to begin at node m at time t_m. Now since p_{m-1} cannot arrive at node m *after* p_m does, and since the relative order of the packets is preserved over all links, it follows that

$$a_{m-1} \leq a_m. \tag{4.22}$$

Now proceeding analogously to the proof of Lemma 4.2:

$$\begin{aligned}
S_i^m(\tau,t) + (m-1)L_i &\geq \min_{V\in[\tau,a_{m-1}]}\{A_i(\tau,V) + G_i^{m-1}(t_m - V)\} + \hat{S}_i^m(0,t-t_m) \\
&\geq \min_{V\in[\tau,a_{m-1}]}\{A_i(\tau,V) + G_i^{m-1}(t_m - V) + \hat{S}_i^m(0,t-t_m)\} \\
&\geq \min_{V\in[\tau,a_{m-1}]}\{A_i(\tau,V) + G_i^m(t-V)\} \qquad (4.23) \\
&\geq \min_{V\in[\tau,a_m]}\{A_i(\tau,V) + G_i^m(t-V),\} \qquad (4.24)
\end{aligned}$$

where the inequality in (4.23) follows from the definition of G_i^m in (3.19), and the

inequality in (4.24) follows from (4.22). This establishes (4.20). Now we can prove the following:

Theorem 4.2 *For every session i, define D_i^* to be the maximum session i packet delay. Then*

$$D_i^* \leq \max_{\tau \geq 0} \left\{ \min\{t : G_i(t) = \hat{A}_i(0,\tau) + (K-1)L_i\} - \tau \right\}. \tag{4.25}$$

Proof. For a given set of arrival functions, $A_1, ..., A_N$ and t such that a packet departs node K at time $t \geq 0$, we have from (4.20):

$$S_i^{(K)}(0,t) \geq \min_{V \in [0,\hat{t}]} \{A_i(0,V) + G_i^K(t-V)\} - (K-1)L_i,$$

where the packet departing at time t arrived at time $\hat{t}$. Thus, for all packet arrival times $\hat{t} \geq 0$:

$$\begin{aligned}
D_i(\hat{t}) &= \min\left\{t : S_i^{(K)}(0,t) = A_i(0,\hat{t})\right\} - \hat{t} \\
&\leq \min\left\{t : \min_{V \in [0,\hat{t}]} \{A_i(0,V) + G_i^K(t-V) - (K-1)L_i\} = A_i(0,\hat{t})\right\} - \hat{t} \\
&\leq \min\left\{t : A_i(0,V_{\min}) + G_i^K(t-V_{\min}) = A_i(0,\hat{t}) + (K-1)L_i\right\} - \hat{t} \\
&\leq \min\left\{t : G_i^K(t-V_{\min}) = A_i(V_{\min},\hat{t}) + (K-1)L_i\right\} - \hat{t} \\
&\leq \min\left\{t : G_i^K(t) = A_i(V_{\min},\hat{t}) + (K-1)L_i\right\} + V_{\min} - \hat{t} \\
&\leq \min\left\{t : G_i^K(t) = \hat{A}_i(0,\hat{t}-V_{\min}) + (K-1)L_i\right\} + V_{\min} - \hat{t} \\
&\leq \min\left\{t : G_i^K(t) = \hat{A}_i(0,\hat{t}-V_{\min}) + (K-1)L_i\right\} - (\hat{t}-V_{\min}) \\
&\leq \max_{\tau \geq 0} \left\{ \min\{t : G_i^K(t) = \hat{A}_i(0,\tau) + (K-1)L_i\} - \tau \right\}.
\end{aligned} \tag{4.26}$$

□

Theorems 4.1 and 4.2 allow us to bound D_i^* and Q_i^* in terms of the universal service curve.

4.2 PGPS networks

When packet sizes are small so that maximum packet transmission time at any link of the network is negligible, we may conclude from Theorem 1.2 of Chapter 1, that the behavior of GPS and PGPS are (essentially) identical. Thus in this case, all of the bounds for GPS networks in Chapter 3 apply to PGPS networks as well.

We now consider the more general case in which packet sizes are not negligible, and outline how the results of Section 4.1 can be extended to this case:

4.2.1 Characterizing the Internal Traffic

Suppose we are given a network in which the nodes are PGPS servers. Further, the assignments of the ϕ_i's meet the CRST requirements of Section 3.5.3. Consider a session $j \in H(1)$ and let $P(j) = \{1, 2, ..., K_j\}$. We know from Corollary 1.1 that

$$\hat{Q}_j^1(\tau) - Q_j^1(\tau) \leq L_{\max}$$

for all τ where $\hat{Q}_j^m$, and Q_j^m represent the session i backlogs at node m, under PGPS and GPS respectively. Thus

$$\hat{Q}_j^{1,*} \leq Q_j^{1,*} + L_{\max}.$$

Also, from Lemma 2.5G:

$$\sigma_j^{\text{out}} = \max\{Q_j^*, \sigma_j\}.$$

Since $S_j^1 \sim (\sigma_j, \rho_j, C_1)$ under GPS, it follows that $\sigma_j^{out,1} \leq \sigma_j + L_{\max}$ under PGPS. Similarly, we can apply the procedure of Section 3.5.3 (modified so that condition (A)) applies, to characterize the internal traffic at each node in $P(j)$, first under GPS and then under PGPS. The algorithm for characterizing the internal traffic for all the sessions is presented below:

- Compute $H_1, ..., H_L$.
- $k = 1$

  While $k \leq L$, for every session $i \in H_k$

  For $p = 1$ to K_i

  $m = P(i,p)$

  Compute $\hat{\sigma}_i^{m,out}$ using the all-greedy bound given:

  $\sigma_j^m = \hat{\sigma}_j^m$ for all sessions j that impede i at m (computed in earlier steps).

  $\sigma_j^m = 0$ for all sessions j that do not impede i at m.

  Set $\hat{\sigma}_i^{(p)} = \hat{\sigma}_i^{m,out} + L_{\max}$.

  $k := k + 1$

4.2.2 Analyzing Delay along the Session i Route

The next two Lemmas are useful to us in the route-wide analysis:

Lemma 4.3 *Given the same set of arrival functions at a node, m. Defining $S_i^{\mathrm{GPS}}(0,t) = 0$ for $t \leq 0$:*

$$S_i^{\mathrm{PGPS}}(0,t) \geq S_i^{\mathrm{GPS}}(0, t - \frac{L_{\max}}{r^m}) \tag{4.27}$$

for each session i and time t.

Proof. From Theorem 1.2 the completion of a packet arrival time under PGPS is delayed by at most $\frac{L_{\max}}{r^m}$ more under PGPS than under GPS. Then the service curve is translated in time at most that amount. The result follows. □

Lemma 4.4 *Suppose we are given arrival functions $A_1, ..., A_N$ at a single GPS server, such that for a particular session i, the k^{th} session i packet arrives at time a_k, and has length $l_k < L_i$. Replace A_i, with $\bar{A}_i$ such that for all k:*

$$\bar{l}_k = l_k$$

and

$$\bar{a}_k \geq a_k,$$

where $\bar{l}_k$ and $\bar{a}_k$ represent the length and arrival time respectively, of the k^{th} session packet under the new regime. Then if f_k is the time that the k^{th} session i packet is served under the old regime, and $\bar{f}_k$ is the time it is served under the new regime, we have:

$$\bar{f}_k \geq f_k$$

for all k.

Proof. Since the server is GPS, the relative order in which packets are served from the set of sessions $\{1, 2, ..., N\} - \{i\}$ is independent of $\bar{A}_i$. Now let T_k be the set of packets from $j \neq i$ transmitted under A_i, at time a_k. Then T_k must be contained in the set $\bar{T}_k$, which is the the set of packets from $j \neq i$ transmitted under $\bar{A}_i$, at time $\bar{a}_k$. The Lemma follows from this fact. □

Now suppose we are given a PGPS network with arrival functions $A_1, ..., A_N$, and we would like to bound delay for a particular session, i. Without loss of generality, assume that $P(i) = \{1, 2, ..., K\}$. The arrival functions A_j^m, for each m, $1 \leq m \leq K$, and $j \in I(m)$ are completely determined and are assumed to be known.

Now construct a GPS network consisting of nodes $1, 2, ..., K$ connected in a line, i.e. the links are given by $\{(e, e+1) : e = 1, 2, ..., K-1\}$. The rates of the links are the same as the corresponding links in the PGPS network, but the link leaving node m has a fixed propagation delay of $\frac{L_{\max}}{r^m}$. The GPS network supports a session i, with route $1, 2, ..., K$ and arrival function given by A_i, i.e. the route and arrival functions are identical to those in in the PGPS network. The other sessions on the GPS network have a route of exactly one hop and are defined as follows: At each node m, for every session j in the PGPS network define a session j' such that

$$A_{j'}^m(0, t) = A_j^m(0, t).$$

Now for each node, m, let $S_i^{m,\text{PGPS}}$ describe the session i departures from node m in the PGPS network, and let $S_i^{m,\text{GPS}}$ describe the session i departures from node m in the corresponding GPS network. Then

Lemma 4.5 *For* $k = 1, 2, ..., K$

$$S_i^{k,\text{GPS}}(0, t - \frac{L_{\max}}{r^k}) \leq S_i^{k,\text{PGPS}}(0, t)$$

for all t.

Proof. By induction on k: For $k = 1$ we are done from Lemma 4.3. Assume the result at nodes $1, 2, ..., k-1$ and show it at node $k \leq K$:
First, consider the service function $\bar{S}_i^{k,\text{GPS}}$ that results at node m if the session i arrivals at node k in the GPS network are identical to the session i arrivals at node k in the PGPS network. Then from Lemma 4.3:

$$\bar{S}_i^{k,\text{GPS}}(0, t - \frac{L_{\max}}{r^k}) \leq S_i^{k,\text{PGPS}}(0, t). \tag{4.28}$$

By the induction hypothesis:

$$S_i^{k-1,\text{GPS}}(0, t - \frac{L_{\max}}{r^{k-1}}) \leq S_i^{k-1,\text{PGPS}}(0, t) \tag{4.29}$$

for all t. The LHS of (4.29) describes the traffic that has traversed the link $(k-1, k)$ in the GPS network in the interval $[0, t]$. Thus, every session i packet arrives at node k *earlier* in the PGPS network, than it does in the GPS network. From Lemma 4.4:

$$S_i^{k,\text{GPS}}(0, t - \frac{L_{\max}}{r^k}) \leq \bar{S}_i^{k,\text{GPS}}(0, t - \frac{L_{\max}}{r^k}) \tag{4.30}$$

for all t. From (4.30) and (4.28):

$$S_i^{k,\text{GPS}}(0, t - \frac{L_{\max}}{r^k}) \leq S_i^{k,\text{PGPS}}(0, t). \tag{4.31}$$

□

Now assume a fixed network topology with no propagation delay. Also assume a fixed internal characterization for all the sessions. Let $D_i^{*,GPS}$ be the worst-case session i delay when the nodes have GPS servers. and let $D_i^{*,PGPS}$ be the worst-case session i delay when the nodes have PGPS servers. Then a direct consequence of Lemma 4.5 is that

$$D_i^{*,\text{PGPS}} \leq D_i^{*,\text{GPS}} + \sum_{m=1}^{K} \frac{L_{\max}}{r^m}. \tag{4.32}$$

Note that the GPS network being considered here has internal characterization identical to the PGPS network—-thus the traffic is burstier than it would be if the procedure of Section 3.5.3 had been used.

Now using the bounds in Theorems 4.1 and 4.2:

$$D_i^{*,\text{PGPS}} \leq \max_{\tau \geq 0} \left\{ \min\{t : G_i^K(t) = \hat{A}_i(0,\tau) + (K-1)L_{\max}\} - \tau \right\} + \sum_{m=1}^{K} \frac{L_{\max}}{r^m}. \tag{4.33}$$

Also note that as the link speeds become faster, i.e., as $r^m \to \infty$,

$$D_i^{*,\text{PGPS}} = D_i^{*,\text{GPS}}.$$

4.3 Rate Proportional Processor Sharing Networks

In this section we will interpret the results of Section 4.2 for a special CRST assignment. Under RPPS Networks $\phi_i^m = \rho_i$ for every session i and $m \in I(m)$. In Section 3.4 we analyzed RPPS networks when the packet sizes are negligible. We concluded that for each session i:

$$Q_i^* \leq \sigma_i, \tag{4.34}$$

and

$$D_i^* \leq \frac{\sigma_i}{\rho_i}. \tag{4.35}$$

Now applying to (4.33), the fact that the slope of G_i^K is never less than ρ_i for each session i, we have:

$$D_i^{*,\text{PGPS}} \leq \frac{\sigma_i + (K-1)L_{\max}}{\rho_i} + \sum_{m=1}^{K} \frac{L_{\max}}{r^m}. \tag{4.36}$$

The first term on the RHS is likely to dominate in most instances. In particular, in high speed networks we assume that $r^m \to \infty$, and we have

$$D_i^{*,\text{PGPS}} \leq \frac{\sigma_i + (K-1)L_{\max}}{\rho_i}, \tag{4.37}$$

and that as $L_{\max} \to 0$, we get (4.35).

The extra delay of $\frac{(K-1)L_{\max}}{\rho_i}$ in (4.36) does not diminish with increasing link speed. However, as the following example shows, this term is not superfluous, but is a consequence of the PGPS service discipline:

Consider a PGPS network with a large number of identically characterized sessions—i.e. $A_j \sim (\sigma, \rho, r)$, $\phi_j = 1$ for each session j, and all the packets have the same length, L. Every link operates at rate r, is shared by N sessions, and

$$N\rho = r - \epsilon, \quad \epsilon > 0, \ \epsilon \approx 0. \tag{4.38}$$

We focus on a session i route that consists of nodes $1, 2, ..., K$, and follow the progress of a session i packet, p, along this route. If p arrives at a node 1 at time t_1, then assume that every other session contending for service at that node sends a packet at time t_1^-. Under PGPS, all $N-1$ packets will be served before p at node 1. Similarly, letting t_m be the time at which p arrives at node m, $2 \leq m \leq K$, we stipulate that every other session contending for service at that node sends a packet at time t_m^-. The delay incurred by p from these packets at node m is $\frac{(N-1)L}{r}$, which is $\approx \frac{L}{\rho}$ for large N. Thus, over all nodes in the route, this delay is $\approx \frac{KL}{\rho}$ for large N. Now letting $r \to \infty$, we observe that the delay term is unchanged as long as (4.38) continues to

hold. If $L = \sigma$, the worst-case packet delay for session i will be at least $\frac{KL}{\rho}$ for large N, which corresponds to (4.37).

This example and (4.37) strongly indicate that small packet lengths should be chosen in RPPS networks so that the term $\frac{L_i}{\rho_i}$ is small. For ATM networks, in which the packets are about 400 bits long, this holds for most kinds of applications.

Next, we compare RPPS to Golestani's Stop-and-Go based strategy [10, 11, 12] in a *high speed network.* (This strategy was described in Section 1.6.5, and we will use notation introduced there.) In [12] the end-to-end packet queuing delay for a type g session, i, is shown to be

$$KT_g \leq D_i^g \leq 2KT_g, \tag{4.39}$$

where K is the number of hops in the session i route. Thus the queueing delay goes up linearly with the number of hops in the route, and also with the frame size.

In order to compare (4.39) to our bounds for RPPS networks, we need to relate T_g to the leaky bucket parameters σ and ρ: Recall that under the Stop-and-Go based strategy at most $r_i T_g$ bits of session k traffic can be contained in a type g frame. In the leaky bucket scheme at most $\sigma_i + \rho_i T_g$ can be sent in any interval of size T_g. Thus given a session that generates traffic according to the leaky bucket constraint we would have to pick r_i and T_g such that

$$r_i T_g \geq \sigma_i + \rho_i T_g,$$

i.e.,

$$T_g \geq \frac{\sigma_i}{r_i - \rho_i}. \tag{4.40}$$

Thus (4.40) gives the lower bound (in terms of r_i) on the smallest frame size that could accommodate traffic that is consistent with $(\sigma_i, \rho_i, \infty)$.

Now assume negligible packet lengths in the RPPS network. Then we have from

(4.37) and (4.39):

$$D_i^{\text{RPPS}} \leq D_i^{\text{Stop-and-Go}} \Rightarrow \frac{\sigma_i}{\rho_i} \leq 2K \frac{\sigma_i}{r_i - \rho_i}$$

i.e.

$$r_i \leq \rho_i(1 + 2K).$$

Under Stop-and-Go for every node, m, $\sum_{k \in I(m)} r_k$ is always less than the capacity of the link represented by node m. Thus, if small frame sizes are used, worst-case session delay under Stop-and-Go is comparable to $D_i^{*,\text{RPPS}}$ only when the network utilization is low relative to RPPS. If large frame sizes are used, then any increase in utilization is accompanied by increase in delay. Thus, when compared in terms of worst-case session delay, RPPS appears to be superior to Stop-and-Go for large networks in which there is a high demand for link bandwidth. However, as pointed out in Section 1.6.5, Stop-and-Go performs better in terms of jitter control.

Finally, we note that the bounds (4.36) hold for more general assignments than RPPS. As we showed in Section 3.4, as long as $g_i \geq \rho_i$, where

$$g_i = g_i = \min_{m \in P(i)} \frac{\phi_i^m}{\sum_{j \in I(m)} \phi_j^m} r^m,$$

we have

$$D_i^* \frac{\sigma_i}{g_i},$$

even if the other sessions are not leaky bucket constrained. Thus we have:

$$D_i^{*,\text{PGPS}} \leq \frac{\sigma_i + (K-1)L_{\max}}{g_i} + \sum_{m=1}^{K} \frac{L_{\max}}{r^m}. \tag{4.41}$$

in this case.

4.4 Conclusions

In this short chapter we brought together a number of important results from earlier parts of the thesis, to provide bounds on worst-case session delay for PGPS networks. These bounds can be evaluated from the universal service curve, which can be efficiently computed. We also observed that when the packet lengths are negligible, the results of Chapter 3 for GPS networks can be applyed without change.

We found that our bound for PGPS networks with non-negligible packet sizes reduces to a simple, closed form solution for Rate Proportional Processing Sharing networks. Further, this bound is independent of the behavior of the other sessions, and scales well with link speed and packet size. These facts bode well for the performance of other, more elaborate CRST assignment schemes.

Chapter 5

Summary and Extensions

The goal of this thesis was to explore the trade-offs between two apparently conflicting requirements of integrated services networks: flexibility and performance guarantees. We sought to achieve this goal through the analysis of a system that combines a work-conserving service discipline called Packet-based Generalized Processor Sharing at the nodes of the network, with leaky bucket admission control at the network periphery. The service discipline is highly flexible, in that it can serve different sessions at different guaranteed rates, and also seems amenable to practical implementation in high speed networks.

We have made considerable progress towards understanding the behavior of the PGPS/Leaky bucket system and have analyzed session delay and backlog for arbitrary topology PGPS networks in the worst-case. Since our techniques apply to very broad classes of server assignments, a major contribution of this thesis is that it provides a flexible framework for flow control in integrated services networks, within which a wide range of hard performance guarantees can be given to real-time sessions.

We have also demonstrated that the worst-case delay bounds provided through our techniques can form the basis of realistic performance guarantees. For example, under Rate Proportional Sharing Networks, we showed that the worst-case delay for any session is bounded (essentially) by the ratio of its bucket size to its token arrival

rate. Thus the bound is independent of the behavior of the other sessions. For more elaborate assignment schemes, we provided efficient algorithms to evaluate the bounds on worst-case delay and backlog that may also be amenable to distributed implementation.

Whether or not the Generalized Processor Sharing approach to flow control is implemented in practice, we believe that our work has shed some light on how to deal with congestion in integrated services networks. Many of the assumptions and techniques employed in our analysis are simple, yet extendible to other service disciplines. Moreover, they can be used to visualize the flow of traffic through the network, which allows for instructive and insightful proofs. This is especially important given the complexity of queueing networks.

Finally, when viewed together, the GPS approach to flow control and the methods used to analyze it, provide a framework that can help one reason about the effects of congestion in integrated services networks, in the worst-case. This gives rise to a number of seemingly analyzable yet unresolved issues, some of which we will attempt to outline in this Chapter.

In the next section we specify the contributions of this thesis in more detail, and discuss some open problems in Section 5.2.

5.1 Contributions

Chapter 1:

- A fair, flexible and efficient multiplexing scheme called Generalized Processor Sharing (GPS).
- A Packet-based Generalized Processor Sharing discipline, PGPS, that is provably close to GPS.
- A practical implementation of PGPS based on a concept of Virtual Time.

Chapter 2:

- A worst-case analysis of a single GPS server that is exact when each session can send packets at the rate of the server.
- An simple, diagrammatic representation of the worst-case behavior of a single GPS server.
- Extensions of the analytical approach used in GPS to understand other service disciplines, such as FCFS and strict priority schemes.

Chapter 3:

- A broad class of GPS assignments called CRST assignments, for which virtual feedback effects can be characterized in the form of internal traffic constraints.
- An analysis of CRST networks with negligible packet sizes for which the entire session route is considered as a whole. This analysis is facilitated by the independent sessions assumption that decouples a given session's route from the rest of the network.

Chapter 4:

- An analysis of CRST GPS networks with general packet sizes.
- An analysis of CRST PGPS networks with general packet sizes.
- An interpretation of the resulting worst-case delay bound for the special case of Rate Proportional Processor Sharing.

5.2 Suggestions For Further Work

Throughout the thesis, we have attempted to point out directions in which our work can be extended. In this section we focus on a few that we consider to be most interesting.

More general arrival constraints should be incorporated into the single node GPS analysis of Chapter 2. We strongly suspect that as long as the greedy arrival curves are convex-$\cap$, the all-greedy regime will still maximize worst-case session delay and backlog. Such a general result would also extend Theorem 2.1 to systems in which $C_j \in (\rho_j, r)$ for some sessions. One approach is to pose the problem as a convex optimization problem, as has been suggested by Humblet [15], but the actual formulation appears to be difficult. Cruz [4] has analyzed other service disciplines using such general arrival constraints, but it is not clear how to extend them to GPS.

Another possibly fertile area of research is in understanding issues of stability in work-conserving networks. Kumar [18] has studied this problem in the context of manufacturing networks, and has made considerable progress in characterizing the conditions that lead to instability. In manufacturing networks, however, the nodes do not represent links and can serve the parts (i.e. packets) of different part-types (i.e. sessions) at different rates. This gives rise to bizarre manifestations of these networks that cause instability. After having done some preliminary work, we strongly conjecture that as long each link of a data network has a utilization of less than 1, it must be stable. However, we cannot even prove this for the case of ring networks when the sessions are leaky bucket constrained, as we saw in Chapter 3.

In this thesis we chose to focus on analytical rather than implementation oriented issues, which does not mean that we do not consider the latter to be important. RPPS networks in particular, may be quite easy to implement, and more elaborate assignments may be feasible as well. However, in order to be sure of their suitability in practice, it is important to estimate how good our bounds are when used to give performance guarantees in a "typical" network. Simulation-based studies may help achieve this goal. It may also be possible to speed up the algorithms outlined in Chapters 2 and 3, since our emphasis was on clarity of exposition rather than on speed of execution.

5.3 A Final Word

As communication and computation speeds continue to increase dramatically, it seems only a matter of time before networks are built to support multimedia applications over wide areas. This thesis focused on a specific flow control strategy that combines the benefits of flexibility, performance guarantees and efficiency. However, there remain many difficult and poorly understood system-level issues in the design of integrated services networks, that go to the very heart of controlling large scale distributed systems. We hope that this thesis and the ideas that shaped it, will be of some help to those who have taken the exciting challenge of resolving these issues head-on.

Bibliography

[1] D. BERTSEKAS AND R. GALLAGER, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[2] D. CLARK. Private Communication, October 1991.

[3] D. CLARK, M. LAMBERT, AND L. ZHANG, *NETBLT: A high throughput transport protocol*, ACM Computer Communication Review, 17 (1988).

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

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

[6] A. DEMERS, S. KESHAV, AND S. SHENKAR, *Analysis and simulation of a fair queueing algorithm*, Proceedings of SIGCOMM '89, (1989), pp. 1–12.

[7] D. FERRARI AND D. C. VERMA, *A scheme for real-time channel establishment in wide-area networks*, IEEE Journal on Selected Areas in Communications, 8 (1990).

[8] R. G. GALLAGER. Private Communication, December 1991.

[9] M. GERLA AND L. KLEINROCK, *Flow control: A comparative survey*, IEEE Transactions on Communications, COM-28 (1980), pp. 553–574.

[10] 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.

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

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

[13] A. C. GREENBERG AND N. MADRAS, *Comparison of a fair queueing discipline to processor sharing*, in Performance '90, Edinborough, Scotland, 1990, pp. 239–254.

[14] J. Y. HUI, *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, Norwell, MA, 1990.

[15] P. HUMBLET. Private Communication, December 1991.

[16] F. P. KELLY, *Reversibility and Stochastic Networks*, John Wiley and Sons, New York, 1987.

[17] R. KOLLA AND B. SERF, *The virtual feedback problem in hierarchical representations of combinatorial circuits*, Acta Informatica, 28 (1991), pp. 463–476.

[18] C. LU AND P. R. KUMAR, *Distributed scheduling based on due dates and buffer prioritization*, tech. rep., University of Illinois Technical Report, 1990.

[19] J. PEHA AND F. TOBAGI, *A cost-based scheduling algorithm to support integrated services*, in Proceedings of IEEE Infocom '91, Miami, FL, 1991.

[20] 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.

[21] H. TAKAGI, *Queueing analysis of polling models: an update*, Stochastic Analysis of Computer and Communication Systems, (1990), pp. 267–318.

[22] E. TELATAR, *Multiaccess Communications with Errors and Erasures Decoding*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, June 1992.

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

[24] R. YATES, *Round Robin Scheduling in High Speed Networks*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, April 1990.

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