Optimal Resource Allocation in Packet Networks

A Thesis

Submitted for the Degree of

Master of Science

in the Faculty of Engineering

by

Nandita D

Center for Electronics Design and Technology

Indian Institute of Science, Bangalore

Bangalore – 560 012 (INDIA)

September 2000

Abstract

Provision of Quality-of-Service (QoS) guarantees is an important issue in the design of Integrated Services packet networks. Call Admission Control is an integral part of the problem. Clearly without Call Admission Control, providing QoS guarantees will be impossible. The task of Call Admission Control is put forth in the following question: Given a new call/session that arrives to a network, can it be accepted by the network at its requested QoS, without violating existing guarantees made to the on-going calls?

In this context, a key issue is how to provide the nodal resources in order to meet the end-to-end QoS requirements of each connection. We investigate the problem of *optimal* nodal bandwidth allocation to satisfy the end-to-end delay requirements of Leaky-bucket shaped connections.

The optimal nodal resource allocation depends greatly on the schedulers used at the network nodes. We first consider a network of Rate-Based schedulers in which the allotted rate for a session at every node is atleast equal to the session's average rate. We obtain the characterization of the departure process of Fluid Rate Proportional Servers. We show that a better resource allocation (less conservative) can be obtained by taking advantage of the properties of the schedulers and the inter-hop dependencies of traffic streams than by considering upper bounds to these traffic streams. We further obtain the optimal bandwidth allocation at each node that satisfies an end-to-end delay requirement in a network of Rate Proportional Servers.

Rate Proportional resource allocation (i.e. rate allotted to a session is atleast equal to its long term average rate) could result in a gross over-allocation of network resources.

This limitation can be overcome by considering a *Non-Rate Proportional* resource allocation. We present an Optimal Call Admission Control algorithm for Generalized Processor Sharing schedulers that does a *General Resource* allocation or a *Non-Rate Proportional Resource* allocation in the single node case. Its fundamental merit arises from the fact that it takes into account the bandwidth released by the sessions which have completed their backlogs, while calculating the sessions rates. The algorithm allots minimum rates to the sessions in order that their delay bounds are not violated. Numerical Results show that the Optimal Call Admission Control can accept significantly more number of connections than the CAC based on the *Rate Proportional* allocation, and hence results in a better network utilization. It is further shown that a lesser resource allocation results in case of Shaped Generalized Processor Sharing schedulers (i.e. an appropriate shaper is introduced before the GPS scheduler).

Finally, we address the problem of reserving bandwidth *optimally* in a *network* of GPS schedulers to provide deterministic end-to-end delay guarantees. The CAC problem in a *network* setting is practically more important and has not been addressed in literature. This problem is non-trivial and has several dimensions apart from those involved in a single-node case, such as the varying connection characteristics as it passes through the nodes in the network and its complex interaction with the other sessions in the network.

Bandwidth allocation in the network case starts by allocating enough bandwidth at each node such that the end-to-end delays can be guaranteed. Then, one seeks to reduce the nodal rate allocations such that the end-to-end delay guarantees are satisfied with equality i.e. the end-to-end delays are not strictly smaller than the required values. It is observed that the "no slack" end-to-end delays can be provided by a number of allocations of the nodal bandwidths. This raises the issue: What is the optimal allocation? It is shown that this problem can be viewed in the framework of a Linear Program, which allows the optimal allocation to be identified easily.

Acknowledgements

I am grateful to my advisors Dr. Joy Kuri and Prof. H. S. Jamdagni for the supervision of this Thesis with care, consideration and interest. I thank Dr. Joy Kuri for going through the Thesis scrupulously. His insights contributed a lot to the results of this Thesis. I will always be grateful to him for his insistence on writing more clearly and for helping make my first research experience an extremely enjoyable one.

I would like to thank Prof. Anurag Kumar and Prof. Utpal Mukherjee for introducing me to Networking. I have benefited greatly by attending the courses taught by them. I gratefully acknowledge the help rendered by the faculty and the staff of CEDT.

I would like to thank all my friends Satya, Sandeep and Laxmi for their wonderful company and constant support. I would like to thank my lab mate Prasanna for making my stay at IISc a lot of fun. I have thoroughly enjoyed my interactions and discussions with him.

Finally, and most importantly, I would like to acknowledge the immeasurable support and encouragement of my parents.

Contents

List of Figures

Chapter 1

Introduction

This thesis focuses on optimal resource allocation in packet networks that support connections with Quality-of-Service requirements. In this Chapter, we present the motivation for our work and a brief overview of the results of the thesis.

1.1 Motivation

The largest packet-switched computer network, the Internet, provides a *Best-Effort* service. In Best-Effort service, a network does not guarantee the packet delivery and if delivered, the delivery time. Such a service is not adequate for several new applications which have diverse Quality-of-Service requirements. These applications which motivate fundamental changes in the services provided by computer networks are:

• Multimedia Applications

For these applications which include video conferencing, video-on-demand, distance-learning, digital libraries, multimedia news services the network must guarantee a minimum bandwidth, the end-to-end delay bound of packets and the end-to-end delay jitter bound.

• Real-time Applications

These applications which include control of manufacturing plants, stock-trading and global electronic commerce, require the network to guarantee bounds on packet delay for their correct functioning.

The design of a network that provides such a flexible QoS is challenging. A fundamental problem at the network layer is to meet the QoS requirements of packet sources and, in order to do so *congestion* must be avoided. A network has to manage two resources to control packet delay and loss, and by avoiding congestion. They are the link bandwidth and buffers at each packet-switch. In a packet-switched network, contention for access to the link and buffer may occur. Thus, a network has to employ algorithms for arbitrating access and limiting contention for these resources.

- Algorithms for arbitrating access

Packet scheduling and buffer scheduling algorithms, arbitrate access to link bandwidth and buffer respectively. A packet scheduling algorithm determines the order of packet transmissions at a link and consequently controls the bandwidth, packet delays and loss for a source. A buffer scheduling algorithm determines the packet to be dropped when buffer overflow occurs and hence controls the packet loss for a source.

- Limiting Contention

The technique for limiting contention depends on the QoS requirements of the sources. We consider sources that require the network to guarantee QoS parameters such as minimum bandwidth and upper bound on packet delay, delay jitter, and loss. A source specifies its QoS requirements and traffic characteristics such as average rate, peak rate, and burst size to the network. The network, in turn, employs an admission control algorithm to determine if the source can be admitted. The admission control algorithm uses the characteristics of the packet scheduling algorithm at the switches and the set of sources that have already been admitted to determine if the QoS requirement of the new source can be met while respecting the QoS requirements of the sources already admitted. If the requirements can be met, the source is admitted and resources are reserved for it.

In this context, a key issue is how to provide the nodal resources in order to meet the end-to-end QoS requirements of each connection. Establishing QoS requirements in a way that minimizes the network resources used, i.e. allotting nodal resources optimally, is an important network optimization problem. Addressing this problem impacts both the routing process and the overall network utilization.

The design and analysis of algorithms to allocate nodal resources such that the connections' QoS guarantees are met, poses several challenges. The question of nodal resource allocations to meet the end-to-end QoS requirements is closely intertwined with the other aspects of a network, such as service models, scheduling disciplines, traffic characterization and QoS specification. Further, the nodal resource allocation must be done in a way that can support a large number of sessions with different performance requirements, while minimizing cost as measured by the network resources. Finally, since the future networks will be high speed in nature, the resource allocation procedures should be computationally efficient. The objective of this thesis is to meet these challenges.

The rest of this Chapter is structured as follows: We present the related work in Section 1.2 and an overview of the results of this Thesis in Section 1.3.

1.2 Related Work

The sequence of packets transmitted by a source to a sink is referred to as a flow. The terms "flow", "session" and "connection" are used synonymously. A flow is serviced by a fixed sequence of packet switches (packet switch is used synonymously with a router). Links are assumed to have a bounded delay. Switches are assumed to be "non-blocking", i.e., when packets arrive at an input link, they can be routed directly to an appropriate output link without switching conflicts. Packets destined for different output links do not interfere with one other, and queueing occurs only at the output port of the switch. The output link of a switch is referred to as a *server*. We assume that *per flow* state is available at each packet switch. This assumption holds for networks with a connection oriented network layer like the Asynchronous Transfer Mode (ATM). It is also applicable in networks with connectionless network layer such as the Internet enhanced with resource reservation protocols such as RSVP. The network supports variable-size packets.

To meet the end-to-end QoS requirements of connections, there are some nodal resource allocation schemes in the literature. We review these schemes briefly.

In [16], Nagarajan *et al* investigated the problem of mapping end-to-end QoS requirements into nodal requirements. They evaluate strategies for such local allocation of end-to-end QoS. A QoS allocation policy is said to perform better than another when the maximum network load that it can support is greater. A major contribution of this work is the development of a *nodal metric* that predicts the relative performance of QoS allocation policies in a network setting. They give insight into the choice of allocation policies by computation of the nodal metric and direct evaluation of allocation policy performance for two simple network models. Their focus was on packet loss probability as the QoS metric and they evaluated the strategies for two stochastic traffic models: Poisson Process and On-Off traffic.

In [15] Lorenze and Orda investigated the problem of optimal resource allocation for end-to-end QoS requirements on unicast paths and multicast trees. They consider a framework in which resource allocation is based on local QoS requirements on each network link, and associated with each link is a convex cost function that increases with the severity of the QoS requirement. They established polynomial solutions to partition an end-to-end QoS requirement into local requirements, such that the overall cost is minimized for both unicast and multicast connections. No specific arrival process is assumed.

Both the above studies do not consider any particular scheduling discipline at the nodes in the network. Though this is good on the scale of generality, this could result in conservative resource allocation. Further they assume unmodified connection characteristics in the network i.e. the traffic characterization of the session at the input of any of the internal nodes in the network is the same as that at the network ingress, and do not incorporate the inter-hop dependencies involved along a session's route. For instance, the internal traffic characterization in a network of *Rate Based Schedulers* like the Virtual Clock, Generalized Processor Sharing etc. will be quite different from that in a network of delay-based schedulers like the Earliest Deadline First etc. Thus the optimal allocation in case of a Generalized Processor Sharing is different from that in Earliest Deadline First.

Thus, the Connection Admission Control at a switch depends greatly on the packet scheduling discipline employed and hence it is imperative to take it into consideration while allotting the nodal resources optimally. Among the scheduling algorithms that have been proposed in the literature, the class of schemes based on Generalized Processor Sharing is most popular.

GPS [23], [24] is an idealized fluid discipline with a number of very desirable properties, including the provision of minimum bandwidth guarantees to each connection regardless of the behavior of the other connections (perfect isolation), and the provision of deterministic, easily computable end-to-end delay bounds for traffic that is leaky-bucket constrained at entry. Because of its powerful properties, GPS has become the reference for an entire class of GPS-related packet-scheduling disciplines, and has been extensively studied in the literature.

The GPS discipline was first proposed as Weighted Fair Queuing (WFQ) in [4]. In their seminal papers [23], [24] on GPS, Parekh and Gallager have analyzed the GPS scheduling discipline in a deterministic setting where the source traffic of each session is regulated by a (σ, ρ)- leaky bucket regulator. In the single node and the multiple node case they obtained closed form expressions for bounds on delay and backlog for a certain class of GPS schedulers called *Rate Proportional Processor Sharing* (RPPS) schedulers. It is shown that the worst-case delay and backlog are obtained in an *All Greedy Regime* in the single node case and a *Staggered Greedy Regime* in the multiple node case. Using this they obtained a method to calculate bounds on the worst case delay and backlog even under a *Non Rate Proportional* resource allocation. In [30], Zhi Li Zhang *et al* have obtained closed-form expressions for end-to-end performance bounds for a broader class of GPS networks known as *Consistent Relative Session Treatment* (CRST) GPS networks. This result generalizes the results of Parekh and Gallager where simple, closed-form end-to-end performance bounds are derived for a special sub-class of CRST GPS networks called RPPS GPS networks, but performance bounds for the general CRST GPS networks do not have closed form. In [26], Yaron and Sidi studied GPS networks with exponentially bounded burstiness arrivals. In [31] Zhang *et al* have investigated the behavior of GPS in a stochastic setting. For a single GPS server in isolation they present statistical bounds on the distributions of backlog and delay for each session. The above mentioned papers deal with the problem of obtaining a bound on delay and backlog for a session in GPS schedulers, *given* a particular rate allocation for the session.

The inverse problem of mapping the QoS requirements (delay in particular) of the sessions to bandwidth allocations in GPS schedulers is of practical importance. Kesidis *et al* in [3] address this problem in a stochastic setting. They concentrated on the single node scenario. When traffic streams satisfy exponential burstiness bounds (EBB), stability and upper bounds on queues and delays at a GPS node have been established. Z.L.Zhang *et al* in [29] have proposed Call Admission Control schemes for a single node GPS server supporting multiple statistical QoS guarantees. The schemes are based on bounds on asymptotic decay rates of the per session backlog tail distributions, derived from the statistical analysis of the GPS scheduling discipline. The statistical QoS metric considered is the loss probability. In [6] the above problem is addressed for *Leaky Bucket Regulated* connections and statistical delay guarantees. Their main goal is to achieve statistical multiplexing gains in the presence of multiple traffic and Quality-of-Service (QoS) classes that share a common trunk. In [7] Robert Szabó *et al* addressed the call admission control in GPS schedulers for *leaky bucket* regulated traffic with deterministic delay guarantees. The CAC established in [7] iteratively searches on a vector space for suitable bandwidth allocations. This could result in an over allocation of bandwidth and sometimes a call block, even if the bandwidth necessary to guarantee the call's QoS requirements is available. The CACs are also computationally complex.

The main limitations of the existing work on nodal resource allocations to satisfy an end-to-end QoS requirement are:

• [16] and [15] do not take advantage of the properties of the schedulers at the nodes in the path of a connection. They assume unmodified connection characteristics in the network and do not incorporate the inter-hop dependencies involved along a session's route. This could result in an overly conservative resource allocation.

• The CACs for GPS do not address the problem of reserving bandwidth *optimally* in GPS schedulers. Further, all the proposed CACs are for a single node server and do not address the end-to-end call admission control for a network of GPS servers ([5] addresses this problem briefly for two heterogeneous QoS classes and does not consider the optimal bandwidth allocation). The end-to-end call admission control is practically more important and is more difficult than the single node case. It does not extend in a straightforward manner from the single-node case.

This thesis addresses these limitations.

1.3 Summary of Contributions

In this Thesis we investigate the problem of *optimal nodal bandwidth allocation to satisfy the end-to-end delay requirements of Leaky-bucket shaped connections.*

The optimal nodal resource allocation depends greatly on the schedulers used at the network nodes. We first consider a network of Rate-Based schedulers in which the allotted rate for a session at every node is atleast equal to the session's average rate. We obtain the characterization of the departure process of Fluid Rate Proportional Servers. We show that a better resource allocation (less conservative) can be obtained by taking advantage of the properties of the schedulers and the inter-hop dependencies of traffic streams than by considering upper bounds to these traffic streams. We further obtain the optimal bandwidth allocation at each node that satisfies an end-to-end delay requirement in a network of Rate Proportional Servers.

Rate Proportional resource allocation (i.e. rate allotted to a session is atleast equal to its long term average rate) could result in a gross over-allocation of network resources. This limitation can be overcome by considering a *Non-Rate Proportional* resource allocation. We present an Optimal Call Admission Control algorithm for Generalized Processor Sharing schedulers that does a *General Resource* allocation or a *Non-Rate Proportional Resource* allocation in the single node case. Its fundamental merit arises from the fact that it takes into account the bandwidth released by the sessions which have completed their backlogs, while calculating the sessions rates. The algorithm allots minimum rates to the sessions in order that their delay bounds are not violated. Numerical Results show that the Optimal Call Admission Control can accept significantly more number of connections than the CAC based on the *Rate Proportional* allocation, and hence results in better network utilization. It is further shown that a lesser resource allocation results in case of Shaped Generalized Processor Sharing schedulers (i.e. an appropriate shaper is introduced before the GPS scheduler).

Finally, we address the problem of reserving bandwidth *optimally* in a *network* of GPS schedulers to provide deterministic delay guarantees. The CAC problem in a network setting is practically more important and has not been addressed in detail in the literature. This problem is non-trivial and has several dimensions apart from those involved in a single-node case, such as the varying connection characteristics as it passes through the nodes in the network and its complex interaction with the other sessions in the network.

Bandwidth allocation in the network case starts by allocating enough bandwidth at each node such that the end-to-end delays can be guaranteed. Then, one seeks to reduce the nodal rate allocations such that the end-to-end delay guarantees are satisfied with equality, i.e., the end-to-end delays are not strictly smaller than the required values. It is observed that the "no slack" end-to-end delays can be provided by a number of allocations of the nodal bandwidths. This raises the question: What is the optimal allocation? It is shown that this problem can be viewed in the framework of a Linear Program, which allows the optimal allocation to be identified easily.

The rest of the Thesis is organized as follows. In Chapter 2 we establish the Optimal Resource Allocation in Packet Networks that use Rate Based Schedulers. Optimal Connection Admission Control in GPS and Shaped GPS are presented in Chapter 3. Connection Admission Control algorithms for a network of GPS schedulers is presented in Chapter 4. Finally, Chapter 5 presents the conclusions of this Thesis and directions for future research.

Chapter 2

Optimal Resource Allocation in Rate Based Schedulers

In this Chapter we address the problem of optimal nodal bandwidth allocation to satisfy an end-to-end delay requirement in a network of rate-based schedulers. We obtain the characterization of the departure process of Rate Proportional Servers for Leaky Bucket constrained input sessions and use it to obtain the optimal allocation at the nodes. We then find that a less conservative allocation can be obtained by taking into consideration the exact traffic characterization than by considering upper bounds to these traffic streams. Using this we find the optimal nodal allocations satisfying an end-to-end delay requirement in a network of Rate Proportional Servers.

The outline of this Chapter is the following:
In Section 2.1 we present some definitions and preliminaries. In Section 2.2 we obtain the optimal resource allocation at the nodes using the worst-case internal traffic characterization. An improvement over this allocation using a better internal traffic characterization is presented in Section 2.3. Summary is in Section 2.4.

2.1 Preliminaries

Leaky Bucket characterization of a traffic stream is based on specifying a two parameter (σ, ρ) envelope on the volume of the arriving traffic [2].

Definition 1: Given $\sigma > 0$, and $\rho > 0$, the traffic stream $A(t)$ is called $(\sigma, \rho) - regular$, if for any interval $(t_1, t_2]$

$$A(t_1, t_2) \leq \sigma + \rho(t_2 - t_1)$$

Let $r_j^{(k)}$ be the rate allotted to session j at the node k.

Definition 2: A session j busy period is a maximum interval of time $(\tau_1, \tau_2]$ such that at any time $t \in (\tau_1, \tau_2]$, the accumulated arrivals of session j since the beginning of the interval do not fall below the total service received during the interval at a rate exactly $r_j^{(k)}$. That is:

$$A_j(\tau_1, t) \geq r_j^{(k)}(t - \tau_1)$$

A Rate Proportional Server is defined in terms of the *session potential* function and the *system potential* function [25]. A session potential function $P_j(t)$ is associated with each session j in the system, describing the state of the session j at time t. A system potential function $P(t)$ describes the state of the system at time t.

Definition 3: A *Rate Proportional Server* is a work conserving server with the following properties:
1) Rate $r_j^{(k)} \geq \rho_j$ is allocated to session j and

$$\sum_{j=1}^{N} r_j^{(k)} \leq C$$

where C is the total service rate of the server and N is the total number of sessions at the server k.

2) Sessions are served at each instant t according to their instantaneous potentials as per the following rules.

a) Among the backlogged sessions, only the set of sessions with the minimum potential at time t is served.

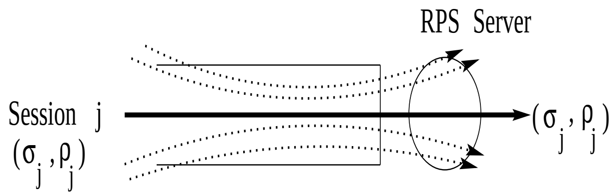

Figure 2.1: Output Traffic Characterization of a RPS scheduler

b) Each session in this set is served with an instantaneous rate proportional to its reservation. The potentials of the sessions in this set increase at the same rate.

In this Chapter we assume that:

• Arrival processes are leaky bucket constrained.

• The scheduling discipline at each node belongs to the family of *Rate Proportional Schedulers* (RPS). GPS and Fluid Virtual Clock are examples of Fluid RPS. PGPS [23], Virtual Clock [28], Frame-Based Fair Queueing (FFQ) and Starting Potential-based Fair Queueing (SPFQ) are examples of Packet-by-Packet RPS (PRPS). Further, new scheduling disciplines can be constructed using the broad framework of the *Rate Proportional Servers.*

2.2 Optimal Resource Allocation with Worst Case Arrival Processes into each Node

The arrival process of session j into the network is $(\sigma_j, \rho_j) - regular$. The departure process was shown to be $(\sigma_j, \rho_j) - regular$ for the GPS discipline in [24]. The following Theorem generalizes this and shows that the output process remains $(\sigma_j, \rho_j) - regular$ for the whole family of RPS scheduling disciplines as illustrated in Figure 2.1

Theorem 2.1 *If the arrival process into a node with a fluid RPS is (σ_j, ρ_j)-regular, then so is the departure process.*

Proof: Let τ be the beginning of a session j busy period. Let $W_j(\tau, t)$ be the total number of bits transmitted during the interval $(\tau, t]$, from session j.

Then, at any time $t > \tau$ that belongs to the same busy period, the service offered to session j is (Theorem 2 in [25]),

$$W_j(\tau, t) \geq r_j^{(k)}(t - \tau) \tag{2.1}$$

Let $Q_j(t)$ denote the backlog of session j at time t. Then:

$$Q_j(t) = A_j(\tau, t) - W_j(\tau, t) \tag{2.2}$$

Further, let $Q_j(t) \leq Q_j^{\max}$ for any (σ_j, ρ_j)-regular stream $A_j(t)$. Here $Q_j^{\max}$ is the maximum backlog achieved during the busy period. Then, it was shown in Lemma 2 of [10] that,

$$W_j(\tau, t) \;\; is \;\; (Q_j^{\max}, \rho_j) - regular \tag{2.3}$$

Equation (2.3) makes no assumptions about the arriving traffic of the other sessions or about the service discipline at the multiplexer. Suppose $Q_j^{\max}$ is achieved at some time t_1, then for a RPS, we get from (2.1), (2.2) and (2.3)

$$\begin{aligned} Q_j^{max} = Q_j(t_1) &\leq \sigma_j + (\rho_j - r_j^{(k)})(t_1 - \tau) \\ &\leq \sigma_j \end{aligned}$$

Hence from (2.3), the departure process $W_j(\tau, t) \;\; is \;\; (\sigma_j, \rho_j) - regular$

$\diamondsuit$

Consider a connection j that is (σ_j, ρ_j)-regular upon entry into the network, and requires an end-to-end delay guarantee of D. A natural question in this context is: how should the demanded end-to-end delay guarantee D be split among the nodes in the connection's path so that the network resources (i.e. bandwidth) allocated is the least? To address this question we consider a bandwidth-delay curve at node k denoted as $R_k(d_k)$: $R_k(d_k)$ gives the bandwidth that must be allocated to connection j at node k so that the delay experienced at that node is never greater than d_k. Note that the bandwidth to be allocated at node k would depend upon the arrival process into node k. In general, the characterization of the arrival process into node k could change with k, and this explains the notation $R_k(.)$. $R_k(.)$ is assumed to be a convex decreasing function. This is motivated by the reciprocal relationship between bandwidth and delay. Thus, the problem is:

$$\text{Minimize} \quad \sum_{k=1}^{n} R_k(d_k)$$

$$\text{subject to} \quad \sum_{k=1}^{n} d_k \leq D$$

Theorem 2.1 asserts that the arrival process into node k can be characterized as $(\sigma_j, \rho_j)-$ *regular* irrespective of k. Hence

$$R_k(d_k) = R(d_k), \qquad k = 1, \ldots, n.$$

Exploiting the convex decreasing nature of the resource-delay curve $R(.)$, it can be shown very simply that the total resource allocation is minimized when D is split equally among the nodes. This is stated below.

Theorem 2.2 *If a (σ_j, ρ_j)-regular stream arrives to a tandem of n RPS nodes with an end-to-end delay requirement of D, then the amount of allocated network resources is minimized when each RPS node allocates a rate sufficient to guarantee a nodal delay of D/n.*

Proof: As mentioned before, the bandwidth-delay curve $R(.)$ is assumed to be a convex function. Thus we have:

$$R(\frac{d_1}{n} + \ldots + \frac{d_n}{n}) \leq \frac{R(d_1)}{n} + \cdots + \frac{R(d_n)}{n}$$

$$n * R(\frac{D}{n}) \leq R(d_1) + \cdots + R(d_n)$$

Thus the minimized total allocated resources $= n * R(\frac{D}{n})$

$\diamondsuit$

Corollary 2.1 *It is optimal to split the worst case end-to-end delay requirement equally among the nodes in the path.*

Consider a connection j with $(\sigma_j, \rho_j) = (40, 10)$ and an end-to-end delay requirement $D = 2$, passing through a network of two nodes. Using Theorem 2.2, the optimal split is 1 at each of the nodes and the total bandwidth required is equal to 80. The next Section shows how this allocation can be improved.

2.3 Optimal Resource Allocation with Actual Arrival Processes

Theorem 2.1 provides a *worst case* characterization of the arrival processes into node $k,\ k = 1, \ldots, n$. According to this result, the arrival process into any node in the tandem can be as bursty as the arrival process at the ingress of the network. However, the simple act of passing through a RPS can smoothen out a bursty process. This Section considers this potential smoothing and arrives at a less conservative allocation. From now on, we consider that the arrival process into the network is greedy [23] from time zero, i.e.

$$A_j(0, t) = \sigma_j + \rho_j t, \quad \forall t \geq 0$$

2.3.1 Fluid RPSs

Let $r_j^{(k)}$ be the guaranteed rate allotted to connection j at node k. This means that while all connections passing through node k are backlogged, connection j is served at rate $r_j^{(k)}$. Now, consider the guaranteed rate $r_j^{(k+1)}$ allotted to connection j at the next node. If $r_j^{(k+1)} > r_j^{(k)}$, the resource allocation is wasteful because there may not be enough arrivals to use the allocated rate $r_j^{(k+1)}$. Hence $r_j^{(k+1)} \leq r_j^{(k)}$ is a reasonable allocation policy. Repeating the argument for all nodes in the tandem we arrive at the following constraint:

$$r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)} > \rho_j \tag{2.4}$$

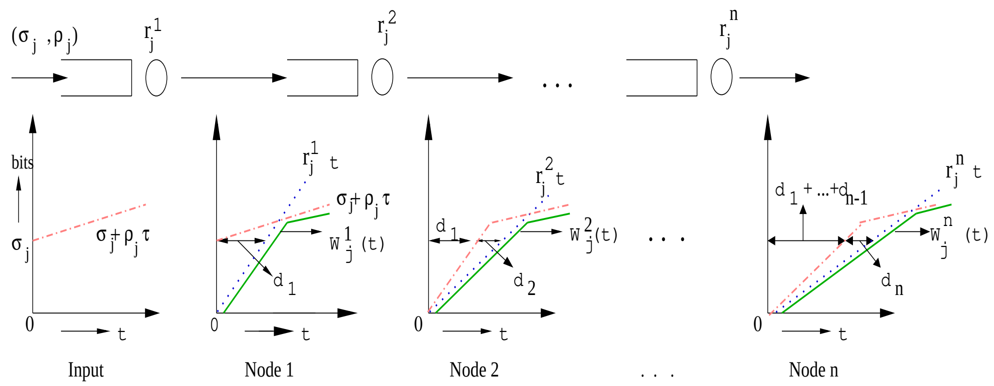

Figure 2.2: Traffic Characterization in a network of RPS schedulers

The *worst case* delay of any session j results when the arrivals of session j are maximized while the service received by session j is minimized. That is, the delay is maximized when session j is greedy starting from time 0 while the service function at a node k is a line of slope $r_j^{(k)}$. Different bits of the greedy session experience different *worst case* delays. The following Proposition shows that the maximum of these worst case delays (*maximum worst case* delay) is experienced by the last bit of the initial burst. The propagation delays are being ignored. Then the following can be shown:

Proposition 2.1 *If*

(i) connection j is (σ_j, ρ_j) − regular upon entry into a tandem of n RPS nodes,

(ii) $r_j^{(k)}$ is the guaranteed rate allocated to connection j at the k^{th} node, $k = 1, \ldots, n$,

(iii) $r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)} > \rho_j$,

then the maximum worst case delay is experienced by the last bit of the initial burst.

Proof: Let $W_j^k(t)$ denote the total number of bits from session j transmitted in $[0, t]$ at node k. Then, as illustrated in Figure 2.2 the output traffic characterization of session j at node 1 is:

$$W_j^1(t) = \min(\sigma_j + \rho_j t, r_j^{(1)} t)$$

At node 2 it is:

$$\begin{aligned} W_j^2(t) &= \min(W_j^1(t), r_j^{(2)} t) \\ &= \min(\sigma_j + \rho_j t, r_j^{(2)} t) \end{aligned}$$

And thus at node k we have:

$$\begin{aligned} W_j^k(t) &= \min(W_j^{k-1}(t), r_j^{(k)}t) \\ &= \min(\sigma_j + \rho_j t, r_j^{(k)}t) \end{aligned} \tag{2.5}$$

Now, consider the sample path of the last bit of the burst of session j, under the traffic characterization in (2.5). It can be seen from Figure 2.2 that:

$$r_j^{(1)} = \frac{\sigma_j}{d_1} \tag{2.6}$$

$$r_j^{(2)} = \frac{\sigma_j}{d_1 + d_2} \tag{2.7}$$

$$\vdots$$

$$r_j^{(n)} = \frac{\sigma_j}{d_1 + .. + d_n} \tag{2.8}$$

From (2.8) the end-to-end delay experienced by the last bit is :

$$d_1 + d_2 + \cdots + d_n = \frac{\sigma_j}{r_j^{(n)}} \tag{2.9}$$

Now, we have to show that the delay of any other bit is $< \frac{\sigma_j}{r_j^{(n)}}$. Consider the sample path of a bit within the initial burst. Let the backlog seen by this bit at the first node be $\sigma_1 (< \sigma_j)$. Then, the equations (2.6), (2.7), (2.8) and (2.9) also hold true for this bit, with the σ_j replaced by σ_1. The end-to-end delay experienced by this bit is $\frac{\sigma_1}{r_j^{(n)}}$ which is $< \frac{\sigma_j}{r_j^{(n)}}$.

In the final case, consider the sample path of a bit that arrives at the first node at some epoch $t_1 > 0$, i.e. at an epoch after the last bit of the burst arrives. It can be easily seen that if such a bit experiences a positive delay at some node k, then it must experience positive delays at the subsequent nodes $k+1, \ldots, n$ too, where $k \geq 1$. This is because of (2.4). Then if k is the first node where such a bit experiences a non-zero delay we have:

$$\sigma_j + \rho_j t_1 = (t_1 + d_k) r_j^{(k)} \; ; k \; \geq 1$$

$$
\begin{aligned}
&= (t_1 + d_k + d_{k+1}) r_j^{(k+1)} \\
&\vdots \\
&= (t_1 + d_k + \cdots + d_n) r_j^{(n)}
\end{aligned}
$$

The end-to-end delay of this bit is:

$$
\begin{aligned}
d_k + d_{k+1} + \cdots + d_n &= \frac{\sigma_j - (r_j^{(n)} - \rho_j) \;\; t_1}{r_j^{(n)}} \\
&< \frac{\sigma_j}{r_j^{(n)}}
\end{aligned}
$$

Thus under the conditions $(i), (ii)$ *and* (iii) of Proposition 2.1, the maximum worst-case delay is experienced by the last bit of the initial burst.

$\diamondsuit$

Considering the sample path of a greedy process and (2.4), it can be shown that:

$$D = \sigma_j / r_j^{(n)}$$

where D = end-to-end delay requirement of connection j.

Then the problem of minimizing allocated resources is:

$$\text{Minimize} \quad (r_j^{(1)} + r_j^{(2)} + \cdots + r_j^{(n)})$$

$$\text{subject to} \;\; (a) \;\; \sigma_j / r_j^{(n)} = D \;\; and$$

$$(b) \;\; r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)} > \rho_j$$

Clearly, the optimal resource allocation is:

$$r_j^{(k)} = \sigma_j / D, \quad k = 1, \cdots, n.$$

Theorem 2.3 *For a greedy* (σ_j, ρ_j) *− regular source with an end-to-end delay requirement D, arriving to a tandem of n fluid RPSs, the optimal resource allocation is* $r_j^{(k)} = \sigma_j / D, \quad k = 1, \ldots, n.$

Proof: It was shown in the proof of *Proposition 1* that the end-to-end delay, $D = \frac{\sigma_j}{r_j^{(n)}}$. This equation along with the condition $r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)} > \rho_j$ gives the optimal resource allocation as:

$$r_j^{(k)} = \frac{\sigma_j}{D}, \quad j = 1, \ldots, n$$

$\diamondsuit$

Theorem 2.3 can be thought as a refinement of Theorem 2.2. The results of Theorem 2.3 are obtained by considering explicitly the actual arrival process into the nodes, while those of Theorem 2.2 are based on looser upper bounds. As a result, Theorem 2.2 drastically over-estimates the resources to be allocated. Secondly, it is interesting to note the following as a consequence of Theorem 2.3.

Corollary 2.2 *The "improved" optimal resource allocation results of Theorem 2.3 imply that the optimal delay split is not the equal split; rather, the optimal delay split results when the entire end-to-end delay is allotted to the first node.*

For the example considered in Section 2.2 i.e. a connection j with $(\sigma_j, \rho_j) = (40, 10)$ and an end-to-end delay requirement $D = 2$, passing through a network of two nodes, Theorem 2.3 gives the total bandwidth required to be equal to $2\ \sigma_j / D = 40$. In comparison the bandwidth required is 80 by Theorem 2.2.

The simple analysis above explains the fact that bursts in traffic disappear after passing through a node. The problem of splitting an end-to-end delay budget into nodal delay allocations has usually been tackled by assuming equal nodal delay allocations [16]. Corollary 2.2 is interesting because it demonstrates that an equal delay split is sub-optimal.

So far, we have considered fluid arrivals and a fluid server. In practice, of course, we have packet arrivals and a packet server. We approach the realistic situation of packet arrivals and packet server by considering the intermediate scenario of packet arrivals and a fluid server.

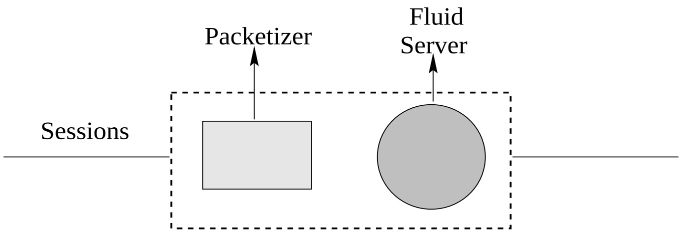

Figure 2.3: A Fluid Server with Packet Arrivals

The case of packet arrivals to a fluid server (Figure 2.3) is characterized by the fact that the service of a packet does not begin till the last bit of the packet has arrived. This implies an additional *packetization* delay at each node. If L_j is the maximum packet length of session j and $r_j^{(k)}$ is the guaranteed rate given to session j at node k, then the packetization delay experienced at node $(k+1)$ is $L_j/r_j^{(k)}$. The additional packetization delays appear in the formulation as follows:

$$\text{Minimize} \quad (r_j^{(1)} + r_j^{(2)} + \cdots + r_j^{(n)})$$

$$\text{subject to} \quad \text{(a)} \quad \frac{\sigma_j}{r_j^{(n)}} + \frac{L_j}{r_j^{(1)}} + \cdots + \frac{L_j}{r_j^{(n-1)}} \leq D \quad \text{and}$$

$$\text{(b)} \quad r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)}$$

Theorem 2.4 *The optimal solution to the above problem is:*

$$r_j^{(k)} = \frac{\sigma_j}{D}[1 + \frac{(n-1)L_j}{\sigma_j}] \quad for 1 \leq k \leq n$$

For proof see Appendix A.

Remark: We observe that on putting $L_j = 0$ in the result of Theorem 2.4, the result of Theorem 2.3 is recovered as expected.

2.3.2 Packet Servers

In this section, we do not consider general packet-by-packet RPS's. The specific scheduling disciplines considered are PGPS, $(WF)^2Q$ [1], and Virtual Clock [28]. In packet servers, only one session can be served at each time-instant and pre-emption is not possible. It is shown in [23] that for PGPS server, the additional delay incurred by a packet in a packet server (compared to that in a fluid server) is at most L_{max}/C_k where L_{max} is the maximum packet size among all the connections and C_k is the capacity of node k. Similar results have been proved in [1] for $(WF)^2Q$ and in [12] for Virtual Clock.†

The optimization problem in this case is:

$$\text{Minimize} \quad (r_j^{(1)} + r_j^{(2)} + \cdots + r_j^{(n)})$$

$$\text{subject to} \quad \text{(a)} \; \frac{\sigma_j}{r_j^{(n)}} + \frac{L_j}{r_j^{(1)}} + \cdots + \frac{L_j}{r_j^{(n-1)}} + \frac{L_{max}}{C_1} + \cdots + \frac{L_{max}}{C_n} \leq D \;\; \text{and}$$

$$\text{(b)} \quad r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)}$$

Theorem 2.5 *The optimal resource allocation for a session j in a network of PGPS, $(WF)^2Q$ and Virtual Clock servers to satisfy an end-to-end delay bound of D is:*

$$r_j^{(k)} = \frac{\sigma_j + (n-1)L_j}{D - \sum_{k=1}^{n} \frac{L_{max}}{C_k}} \quad for \quad 1 \leq k \leq n$$

For proof see Appendix A.

2.4 Summary

In this Chapter we show:

• The characterization of the departure process of fluid Rate Proportional Servers.

• A better resource allocation (less conservative) can be obtained by taking advantage of the properties of the schedulers and the inter-hop dependencies of traffic streams than by considering upper bounds to these traffic streams.

• The optimal bandwidth allocation at each node that satisfies an end-to-end delay requirement in a network of *Rate Proportional Servers.*

---

†A bound on the worst case delay introduced by a packet server for a *general packetized* RPS does not exist [25]. Hence the results of this section are specific to certain scheduling disciplines and do not apply to a general PRPS.

Chapter 3

Optimal Call Admission Control in GPS Schedulers

In Chapter 2 we addressed the problem of optimal resource allocation in a family of schedulers called the Rate Proportional Servers (RPS). In RPS, the rates of the sessions are set according to the session bandwidth demands i.e. the rate allotted to a session is at least equal to its long term average rate, ρ. This often results in an over-allocation which leads to a waste of network resources.

Thus, it is of interest to investigate the *General Resource Allocation* or the *Non-Rate Proportional Resource Allocation* which can result in a higher network utilization. Unlike in the Rate Proportional setting where a set of schedulers exhibits similar properties and shares the same delay bounds, the schedulers could have vastly different properties in the scenario of Non-Rate Proportional allocation. There is no framework in the literature to systematically analyze the scheduler properties in a Non Rate Proportional setting. It is thus necessary to treat every scheduling algorithm individually while addressing the problem of Non Rate Proportional resource allocation.

The class of scheduling algorithms based on Generalized Processor Sharing (GPS) is most popular in the literature. GPS [23], [24] is an idealized fluid discipline with a number of very desirable properties, including the provision of minimum bandwidth guarantees to each connection regardless of the behavior of the other connections (perfect isolation), and the provision of deterministic, easily computable end-to-end delay bounds for connections whose traffic is leaky-bucket constrained. Because of its powerful properties, GPS has become the reference for an entire class of GPS-related packet-scheduling disciplines, and has been extensively studied in the literature. However, the existing CAC frameworks for GPS, though simple, result in a substantial portion of the available bandwidth being wasted.

A *CAC framework for GPS which meets the connections' delay requirements (deterministic) and does an Optimal Non Rate Proportional bandwidth allocation* is the subject matter of this Chapter. We also show that a better resource allocation can be obtained by placing an appropriate shaper before the scheduler.

The Chapter is organized as follows: Section 3.1 describes the GPS scheduling policy and gives the necessary background. Section 3.2 describes the optimal Call Admission Control (CAC) algorithm. In Section 3.3 we present numerical results to demonstrate the effectiveness and the performance of the optimal CAC. Section 3.4 presents the computational complexity of the CAC. Section 3.5 describes the optimal CAC in Shaped GPS (SGPS) while Section 3.6 concludes the Chapter.

3.1 Preliminaries

3.1.1 Definitions and Notations

We assume a packet switch where each output link uses the Generalized Processor Sharing (GPS) scheduling policy to schedule packets from the set of sessions input to the link.

In this paper we assume that the arrival processes are *Leaky Bucket* constrained. A session j also specifies a delay guarantee d_j, which is the maximum difference allowed between the departure epoch and the arrival epoch of any bit of session j, at the link. Let r_j denote the bandwidth allotted to session j at the link to satisfy it's delay guarantee d_j. Let $A_j(\tau, t]$ denote arrivals from session j during the interval $(\tau, t]$ and $W_j(\tau, t]$ the amount of service received by session j during the same interval. Let $Q_j(t)$ represent the amount of session j traffic queued in the server at time t, that is

$$Q_j(t) = A_j(0,t] - W_j(0,t]$$

A session j is backlogged at time t if $Q_j(t) > 0$.

Definition 1: A **backlogged period** for a session j is any period of time during which packets belonging to that session are continuously queued in the system.

Definition 2: A **system busy period** is defined to be the maximal interval B such that for any $\tau, t \in B, \quad \tau \leq t$:

$$\sum_{j=1}^{N} W_j(\tau, t) = C\ (t - \tau)$$

where C is the server capacity and N is the number of sessions at the server.

A session j is said to be *greedy* starting from time τ if it sends traffic at maximum possible rate for all times $\geq \tau$.

Definition 3: An **All-Greedy GPS System** is one in which all the sessions are greedy starting from time 0, the beginning of a system busy period i.e.

$$A_j(0,\tau) = \sigma_j + \rho_j\ \tau, \quad \tau \geq 0$$

3.1.2 Generalized Processor Sharing (GPS) Scheduling

Generalized Processor Sharing (GPS) is a scheduling discipline that can be regarded as the limiting form of a weighted round robin policy, where the traffic from the sessions is treated as an infinitely divisible fluid. A GPS server serving N sessions is characterized by N positive real numbers, $\phi_1, \ldots, \phi_N$. These numbers denote the relative amount of service given to each session in the sense that if $W_j(\tau, t)$ is defined as the amount of session j traffic served by the GPS server during an interval $(\tau, t]$, then:

$$\frac{W_j(\tau, t]}{W_k(\tau, t]} \geq \frac{\phi_j}{\phi_k}, \quad k = 1, \ldots, N \tag{3.1}$$

for any session j that is continuously backlogged in the interval $(\tau, t]$. Thus (3.1) is satisfied with equality for two sessions j and k that are both backlogged during the interval $(\tau, t]$. Note from (3.1) that whenever session j is backlogged it is guaranteed a minimum service rate of

$$r_j = \frac{\phi_j}{\sum_{k=1}^{N} \phi_k} C \tag{3.2}$$

where C is the capacity of the server. This rate is called the session j *backlog clearing rate* since a session j backlog of size q is served in at most $\frac{q}{r_j}$ time units. The server is said to be stable if $\sum_{k=1}^{N} \rho_k < C$, i.e. the sum of the long term sustained rates of the sessions is less than the service rate.

GPS is an idealized fluid server in which the traffic is considered as infinitely divisible and all sessions are being served simultaneously, sharing the server capacity. Although GPS itself cannot be implemented in practice, many practical implementations are based on GPS (i.e. they emulate a GPS server in the background).

3.1.3 Delay Bounds of Leaky Bucket Constrained Sessions in a GPS Server

The arrival process of a session j is assumed to be $(\sigma_j, \rho_j) - regular$.

Parekh and Gallager obtained the following bounds in [23] for the maximum network queueing delay, d_j^* and the maximum network backlog, q_j^* for a session j passing through a network of *RPPS* servers:

$$q_j^* \leq \sigma_j \quad and \quad d_j^* \leq \frac{\sigma_j}{r_j} \tag{3.3}$$

Further, it was also shown that any session j in the GPS system achieves its maximum delay and backlog in an *all-greedy regime*. This gives a simple scenario for investigating the worst case behavior.

However, the bounds obtained above are quite loose since the assumption is that the session j is served with a constant rate r_j till its backlog is cleared. Also, the bound holds only for a sub-class of GPS-based systems called *Rate Proportional Processor Sharing (RPPS)* in which the guaranteed rate $r_j \geq \rho_j$. In reality, the service rates of backlogged

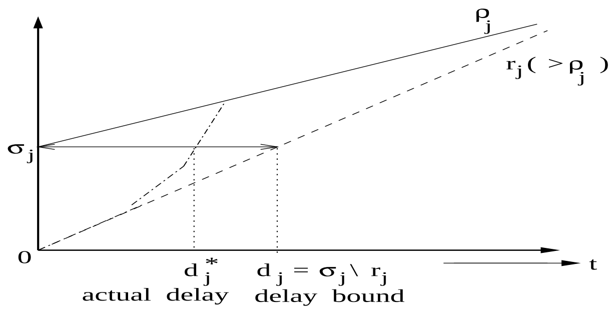

Figure 3.1: Illustrates the looseness of the delay bound

sessions increase as more and more sessions complete their backlogged periods. This is because as each session completes its backlog, the bandwidth released is distributed among the still backlogged sessions in proportion to their weights. Figure 3.1 illustrates the looseness of the bounds in (3.3) due to not accounting for the excess bandwidth received by the session j. Robert Szabó *et al* address this problem in [8] and present a two step algorithm for calculating tighter upper bounds for delay d_j^*. This algorithm takes care of the backlog finish times of the sessions and can also be used for arbitrary sets of $\{\phi\}$ values (*Non-Rate Proportional*) and thus overcomes both the problems associated with the bound in (3.3). The following notation is used in the algorithm and will be used henceforth in the subsequent sections too. Let C denote the server capacity and $\mathcal{L} = \{L(i) |\ L(i) \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, N\}\}$ be an ordered set of indices in which $L(i)$ means that the session $L(i)$ backlog is cleared i^{th} in order. That is, the backlog of session $L(1)$ is cleared first, followed by the clearing of backlog of session $L(2)$, and so on. The time instant when the backlog of session $L(i)$ becomes zero is denoted by $t_{L(i)}$. By definition, let $t_{L(0)} = 0$, the start of the system busy period with all sessions greedy. Further, let $r_j^{(k)}$ be defined as the session j service rate in the $[t_{L(k-1)}, t_{L(k)}]$ interval. During the interval $[0, t_{L(1)}]$ every session is served by its minimum guaranteed rate i.e.

$$r_j^{(1)} = r_j$$

Between $t_{L(1)}$ and $t_{L(2)}$ session $L(1)$ has no backlog, so it is served by $\rho_{L(1)}$. The service rates of the still backlogged sessions during that period are:

$$r_j^{(2)} = r_j + \frac{r_j}{\sum_{i=1}^{N} r_i - r_{L(1)}} \quad (r_{L(1)} - \rho_{L(1)}), \quad j \quad \in \{1, \ldots, N\} \backslash \{L(1)\}$$

The second term on the right hand side comes from the fact that after the backlog of session $L(1)$ is cleared, the remaining bandwidth $(r_{L(1)} - \rho_{L(1)})$ is distributed among the still backlogged sessions in proportion to their allotted rates.

It is proved in [8] that during the time interval $[t_{L(k-1)}, t_{L(k)}], \quad k = 1, \ldots, N$, within the system busy period, the backlogged session j is served at a rate

$$r_j^{(k)} = \frac{(C - \sum_{l=1}^{k-1} \rho_{L(l)}) r_j}{\sum_{i=1}^{N} r_i - \sum_{l=1}^{k-1} r_{L(l)}}, \quad j \quad \in \{1, \ldots, N\} \backslash \{\bigcup_{m=1}^{k-1} L(m)\} \tag{3.4}$$

The two step algorithm in [8] for calculating tight upper bounds on delay is described in the following:

Step 1: Algorithm for Calculating Backlog Clearing Times

In the first step the time needed for a session j to empty its backlog is expressed as $t_{j,1} = \frac{\sigma_j}{r_j^{(1)} - \rho_j}$. The session that completes its backlog first is the one whose $t_{j,1}$ takes the smallest possible value. That is:

$$t_{L(1)} = \min\{t_{j,1} | t_{j,1} > 0; \quad j \in \{1, \ldots, N\}\}$$

where $L(1)$ is the index of that session which satisfies the above minimum.

The k^{th} step candidate finishing times of sessions still backlogged are:

$$t_{j,k} = \frac{\sigma_j + \sum_{m=1}^{k-1} r_j^{(m)} (t_{L(m-1)} - t_{L(m)}) + r_j^{(k)} t_{L(k-1)}}{r_j^{(k)} - \rho_j} \tag{3.5}$$

The k^{th} step finishing time is calculated by taking the minimum of (3.5) over $j \in \{1, \ldots, N\} \backslash \{\bigcup_{m=1}^{k-1} L(m)\}$ i.e.

$$t_{L(k)} = \min\{t_{j,k} | t_{j,k} > 0; \quad j \in \{1, \ldots, N\} \backslash \bigcup_{m=1}^{k-1} L(m)\}$$

Step 2: Calculating Tighter Upper Bounds for Delay

The following Theorem gives tight bounds for the delay, denoted by d_j^*:

Theorem 3.1 *Let* $r_j(t) = r_j^{(k)}$, *where* $t \in [t_{L(k-1)}, t_{L(k)}]$

(i) Let τ_j *be defined as the epoch at which:*

$$W_j(0, \tau_j] = \int_0^{\tau_j} r_j(t)dt = \sigma_j$$

If

$$r_j(\tau_j) \geq \rho_j \tag{3.6}$$

then

$$d_j^* = \tau_j$$

(ii) If (3.6) does not hold for session j *then it starts with an accumulating phase, for which there exists a* $i \in \{1, \ldots, N\} \backslash \{j\}$ *such that*

$$\forall t < t_{L(i)} \ : \ r_j(t) < \rho_j$$

and

$$\forall t \geq t_{L(i)} \ : \ r_j(t) \geq \rho_j$$

then

$$d_j^* = t_{L(i)} - \frac{W_j(0, t_{L(i)}] - \sigma_j}{\rho_j}$$

where $W_j(0, t)$ *is the amount of session* j *traffic served up till time* t.

$\diamondsuit$

3.2 Optimal Call Admission Control Algorithm

The Connection Admission Control (CAC) for GPS scheduling used in the literature to provide deterministic delay guarantees for sessions is based on the delay bound (3.3). In this simple CAC a connection j is allotted a bandwidth of $\max(\frac{\sigma_j}{d_j}, \rho_j)$ where d_j is the delay guarantee asked for. The nice part about this CAC is that it completely eliminates interference among sessions, as if there were firewalls in between those individually reserved bandwidth channels. Thus, irrespective of the arrival patterns of other sessions, connection j will be assured at least a bandwidth of $\max(\frac{\sigma_j}{d_j}, \rho_j)$. An incoming session j is refused if this bandwidth is not available at the server. However, as we had seen in Section 3.1 such an allocation could result in the actual worst-case delay being far less than the tolerable delay d_j, thus leading to a waste of the allotted bandwidth.

The basic idea of the Optimal CAC was inspired by the method to obtain tight delay bounds, in Theorem 3.1. The algorithm takes advantage of the backlog clearing times of some sessions in allotting bandwidth for a connection j, to meet its delay requirement, d_j. Our goal is to achieve both the constrained delay requirements of the connections and to have some extent of isolation among the sessions, while at the same time preserving statistical multiplexing advantages.

The algorithm runs iteratively, where at the beginning of each iteration the rates of the sessions are calculated to satisfy their delay bounds. Then, the backlog clearing times of these sessions are calculated. The session with the minimum backlog clearing time is identified, and its rate is kept fixed for the subsequent iterations. The algorithm proceeds with the next iteration and recalculates the rates of the remaining sessions taking into account the most recent backlog clearing time (determined in the previous iteration).

In the rest of the section, we give an exact description of the Optimal CAC algorithm first and then explain its major functions.

3.2.1 The Algorithm

The CAC algorithm determines the session rates needed to support their delay guarantees in an *All-Greedy Regime* (Def. 3). It is proved in Theorem 3 of [23] that for a session j the worst-case delay, d_j^* and the worst case backlog, q_j^* are achieved in the All-Greedy Regime. Thus, the rates calculated in this scenario will satisfy the delay bounds of all the sessions, under any other arrival pattern too.

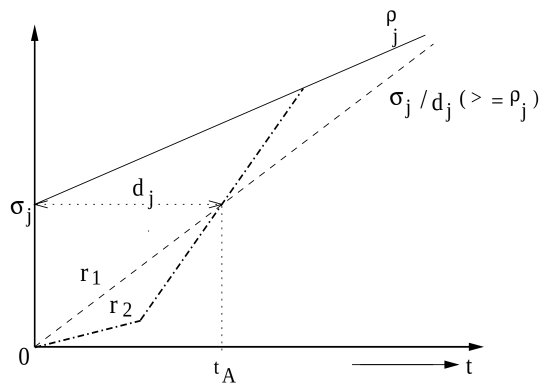

Figure 3.2: Service rate at t_A is $\geq \rho_j$ for a session with $\frac{\sigma_j}{d_j} \geq \rho_j$

Before listing the steps of the algorithm we introduce the following Lemmas and Theorems which form the basis for the sessions' rate calculations in the algorithm.

Lemma 3.1 *If for a session j, $\frac{\sigma_j}{d_j} \geq \rho_j$, and the delay requirement d_j is to be satisfied then,*
(i) The service rate of session j is $\geq \rho_j$, when the last bit of the burst is getting served.
(ii) The maximum delay is experienced by the last bit of the burst.

Proof: (i) See Figure 3.2. Let the time when the last bit of the burst is getting served be t_A. If the allotted rate $r1 = \frac{\sigma_j}{d_j}$, since $\frac{\sigma_j}{d_j} \geq \rho_j$, it is clear that the service rate at t_A is $\geq \rho_j$.
If the allotted rate $r2 < \frac{\sigma_j}{d_j}$, then in order not to violate the delay bound d_j, it is necessary that the service rate at t_A be $> \frac{\sigma_j}{d_j}$ and hence $> \rho_j$.
(ii) This is a direct consequence of Theorem 3.1(i) and the part(i) above.

$\diamondsuit$

Lemma 3.2 *If for a session j, $\frac{\sigma_j}{d_j} < \rho_j$, and the delay requirement d_j is to be satisfied, then neither (i) nor (ii) of Lemma 3.1 is necessarily true.*

Proof: This can be seen from Figure 3.3 where $\frac{\sigma_j}{d_j} < \rho_j$. The session j service rate when the last bit of the burst is being served at time t_A is $< \rho_j$, without violation of the delay bound, d_j. The maximum delay, d_j^* is experienced by some bit b at time τ and not by the last bit of the burst.

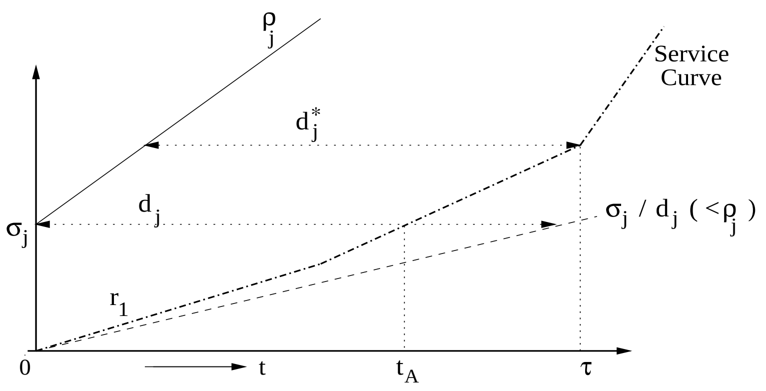

Figure 3.3: Service rate at t_A could be $< \rho_j$ for a session with $\frac{\sigma_j}{d_j} < \rho_j$

$\diamondsuit$

The following Lemma gives a method to calculate the guaranteed rate for a session j such that the last bit of the initial burst experiences a delay exactly equal to d_j.

Lemma 3.3 *Let $t_{L(1)}, t_{L(2)}, \ldots, t_{L(i-1)}$ be the session backlog clearing times which are $<$ d_j. Then, the rate r_j needed for the last bit of the initial burst to experience the delay d_j is:*

$$r_j = \frac{\sigma_j}{\sum_{k=1}^{i-1}(t_{L(k)} - t_{L(k-1)})\left(\frac{C-\sum_{m=1}^{k-1}\rho_{L(m)}}{C-\sum_{m=1}^{k-1}r_{L(m)}}\right) + (d_j - t_{L(i-1)})\left(\frac{C-\sum_{m=1}^{i-1}\rho_{L(m)}}{C-\sum_{m=1}^{i-1}r_{L(m)}}\right)} \tag{3.7}$$

Proof: Let r_j be the required rate to be allotted and $r_j^{(k)}$ be the session j rate during $[t_{L(k-1)}, t_{L(k)}]$. Then as given by (3.4)

$$r_j^{(k)} = \frac{r_j}{C - \sum_{m=1}^{k-1} r_{L(m)}}(C - \sum_{m=1}^{k-1} \rho_{L(m)}) \tag{3.8}$$

where the first factor is the proportion of bandwidth that session j receives after sessions $L(1), L(2), \ldots, L(k-1)$ complete their backlogs while the second factor is the available bandwidth itself in the interval $[t_{L(k-1)}, t_{L(k)}]$. Then (see Figure 3.4):

$$\sigma_j = \sum_{k=1}^{i-1}[(t_{L(k)} - t_{L(k-1)})\ r_j^{(k)}] \quad + \quad (d_j - t_{L(i-1)})\ r_j^{(i)} \tag{3.9}$$

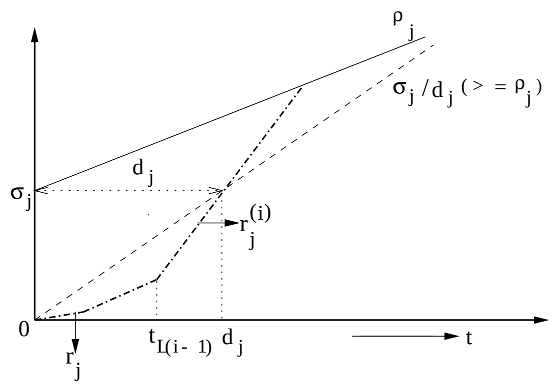

Figure 3.4: Rate calculation such that the last bit of the burst experiences a delay $= d_j$

The result in (3.7) is easily obtained by substitution of $r_j^{(k)}$ from (3.8) into the above equation.

$\diamondsuit$

The following Corollary follows easily from Lemma 3.1 and Lemma 3.3.

Corollary 3.1 *For a session with a required delay guarantee of d_j and whose $\frac{\sigma_j}{d_j} \geq \rho_j$, the allocation of the above rate (3.7) results in the worst case delay, $d_j^* = d_j$.*

Similarly, the following Corollary follows from Lemma 3.2.

Corollary 3.2 *If $\frac{\sigma_j}{d_j} < \rho_j$ and the required delay guarantee is d_j, then with the rate allocation in (3.7), the worst case delay d_j^* need not be equal to d_j, i.e, can be greater than d_j.*

The following Lemma calculates the rate needed for session j such that it experiences a delay of d_j at the epoch $t_{L(i-1)}$, with $d_j \leq t_{L(i-1)}$.

Lemma 3.4 *Let $t_{L(1)}, t_{L(2)}, \ldots, t_{L(i-1)}$ be the session backlog clearing times and $d_j \leq t_{L(i-1)}$. Then, in order for the delay d_j to occur at the break-point $t_{L(i-1)}$, the rate needed is :*

$$r_j = \frac{\rho_j(t_{L(i-1)} - d_j) + \sigma_j}{\sum_{k=1}^{i-1}(t_{L(k)} - t_{L(k-1)})\left(\frac{C - \sum_{m=1}^{k-1}\rho_{L(m)}}{C - \sum_{m=1}^{k-1} r_{L(m)}}\right)} \tag{3.10}$$

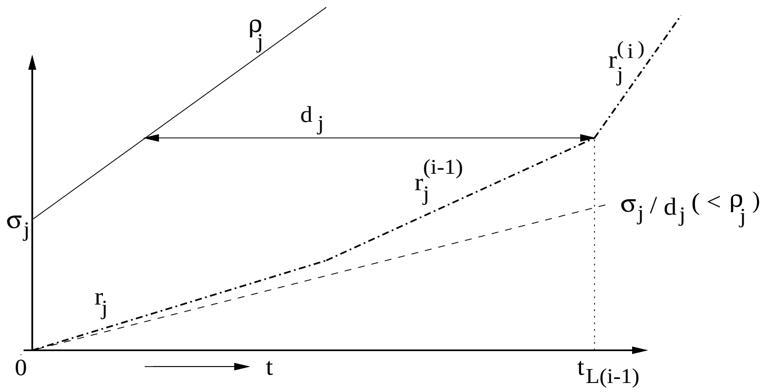

Figure 3.5: Rate calculation such that the delay d_j is experienced at epoch $t_{L(i-1)}$

Proof: Let r_j be the required rate to be allotted and $r_j^{(k)}$ be the session j rate during $[t_{L(k-1)}, t_{L(k)}]$. Then as given by (3.4)

$$r_j^{(k)} = \frac{r_j}{C - \sum_{m=1}^{k-1} r_{L(m)}} (C - \sum_{m=1}^{k-1} \rho_{L(m)}) \tag{3.11}$$

where the first factor is the proportion of bandwidth that session j receives after sessions $L(1), L(2), \ldots, L(k-1)$ complete their backlogs while the second factor is the available bandwidth itself in the interval $[t_{L(k-1)}, t_{L(k)}]$. Then (see Figure 3.5):

$$\rho_j (t_{L(i-1)} - d_j) + \sigma_j = \sum_{k=1}^{i-1} [(t_{L(k)} - t_{L(k-1)})\ r_j^{(k)}] \tag{3.12}$$

The result in (3.10) is easily obtained by substitution of $r_j^{(k)}$ from (3.11) into the above equation.

$\diamondsuit$

The following Corollary follows easily from Lemma 3.1.

Corollary 3.3 *For a session with* $\frac{\sigma_j}{d_j} \geq \rho_j$ *and required delay guarantee of* d_j*, the above allocation results in the worst-case delay* $d_j^* = d_j$ *only if* $d_j = t_{L(i-1)}$

The following Corollary follows easily from Theorem 3.1(ii):

Corollary 3.4 *For a session with* $\frac{\sigma_j}{d_j} < \rho_j$ *and required delay guarantee of* d_j*, the above allocation results in the worst-case delay* $d_j^* = d_j$ *only if* $r_j^{(i-1)} < \rho_j$ *and* $r_j^{(i)} \geq \rho_j$*.*

In each iteration, the algorithm tries to reduce the rate allocated to each connection, if possible, while ensuring that the required end-to-end delay guarantees are satisfied.

Algorithm :

Step 1: *Initialization*

$i = 0$ *Iteration Variable*

$level = 1$ *An attribute of a session indicating when the backlog of the session may begin service*

$t_{L(i)} = 0$, $r_{L(i)} = 0$ for all $1 \leq i \leq N$ *The session backlog clearing times and the session rates*

$C_s = C$ *C_s is the "assumed" sum of rates that the algorithm begins with. C is the server capacity*

The following sets are defined:

$B1 = \emptyset$ *$B1$ will consist of those sessions for which the service rate is greater than their average rate when the last bit of the burst is being served*

$B2 = \emptyset$ *$B2$ will consist of those sessions for which the service rate is not necessarily greater than the average rate when the last bit of the burst is being served*

$P = \emptyset$ *P will consist of those sessions for which the rates have been determined but not the backlog finish times*

$H = \emptyset$ *H will consist of sessions at a particular level for which the rates and the backlog finish times are both determined*

$H_{main} = \emptyset$ *H_{main} will consist of the sessions included in the set H at different levels *

Step 2: *Classification of the sessions*

$B1 = \{\text{session } j \mid \frac{\sigma_j}{d_j} \geq \rho_j\}$

$B2 = \{\text{session } j \mid \frac{\sigma_j}{d_j} < \rho_j\}$

Step 3: *Beginning of an Iteration*

$i = i + 1$

Step 4: *Rate Calculation for Sessions in B1 and B2*

(a) Sessions $\in B1$:

$\forall\ j \in B1,\ r_j\ =$

$$\frac{\sigma_j}{\sum_{k=1}^{i-1}(t_{L(k)}-t_{L(k-1)})\left(\frac{C-\sum_{m=1}^{k-1}\rho_{L(m)}}{C-\sum_{m=1}^{k-1}r_{L(m)}}\right)+(d_j-t_{L(i-1)})\left(\frac{C-\sum_{m=1}^{i-1}\rho_{L(m)}}{C-\sum_{m=1}^{i-1}r_{L(m)}}\right)}$$

(b) Sessions $\in B2$:

if ($i\ =\ 1$)

$r_j = \rho_j$

else if ($d_j\ \leq\ t_{L(i-1)}$)

{

$$r_j = \frac{\rho_j(t_{L(i-1)}-d_j)+\sigma_j}{\sum_{k=1}^{i-1}(t_{L(k)}-t_{L(k-1)})\left(\frac{C-\sum_{m=1}^{k-1}\rho_{L(m)}}{C-\sum_{m=1}^{k-1}r_{L(m)}}\right)}$$

if ($r_j^{(i-1)}\ <\ \rho_j$ and $r_j^{(i)}\ \geq\ \rho_j$)

{

$P\ =\ P\ \cup\ \{j\}$

$B2 = B2 - \{j\}$

}

else if ($r_j^{(i-1)}\ \geq\ \rho_j$)

{

$$r_j = \frac{\rho_j}{\left(\frac{C-\sum_{k=1}^{i-2}\rho_{L(k)}}{C-\sum_{k=1}^{i-2}r_{L(k)}}\right)}$$

$P\ =\ P\ \cup\ \{j\}$;

$B2\ =\ B2\ -\ \{j\}$;

}

else if ($r_j^{(i)}\ <\ \rho_j$)

$$r_j = \frac{\rho_j}{\left(\frac{C-\sum_{k=1}^{i-1}\rho_{L(k)}}{C-\sum_{k=1}^{i-1}r_{L(k)}}\right)}$$

}

else if ($d_j\ >\ t_{L(i-1)}$)

{

$r_j\ =$

$$\frac{\sigma_j}{\sum_{k=1}^{i-1}(t_{L(k)}-t_{L(k-1)})\left(\frac{C-\sum_{m=1}^{k-1}\rho_{L(m)}}{C-\sum_{m=1}^{k-1}r_{L(m)}}\right)+(d_j-t_{L(i-1)})\left(\frac{C-\sum_{m=1}^{i-1}\rho_{L(m)}}{C-\sum_{m=1}^{i-1}r_{L(m)}}\right)}$$

if $(r_j^{(i)} \geq \rho_j)$

{

$B1 = B1 \cup \{j\}$

$B2 = B2 - \{j\}$

}

else

$r_j = \frac{\rho_j}{\left(\frac{C - \sum_{k=1}^{i-1} \rho_{L(k)}}{C - \sum_{k=1}^{i-1} r_{L(k)}}\right)}$

}

Step 5: *Calculation of the backlog clearing times for sessions* $\in B1, B2$ *and* P

$t_{j,i} = \frac{S_{j,i}}{r_j^{(i)} - \rho_j} \quad \forall\, j \in B1,\ B2$ and P

where $S_{j,i} = \sigma_j + \sum_{m=1}^{i-1} [r_j^{(m)} (t_{L(m-1)} - t_{L(m)})] + r_j^{(i)} (t_{L(i-1)})$

and $r_j^{(m)} = r_j \left(\frac{C - \sum_{k=1}^{m-1} \rho_{L(k)}}{C - \sum_{k=1}^{m-1} r_{L(k)}}\right)$

Step 6: *Choose the session with the minimum backlog time and include it in* H

$L(i) =$ session j such that:

$t_{L(i)} = \min\{t_{j,i} \mid 0 < t_{j,i} < \infty;\ j \in B1,\ B2,\ P\}$

if (no such $t_{L(i)}$ exists)

then $L(i) = 0$;

else

{

$H = H \cup \{L(i)\}$

Delete $L(i)$ from the set $B1$ or $B2$ or P (whichever contained $L(i)$)

}

Step 7: *Identify sessions* $\in B1$ *whose delay is less than* $t_{L(i)}$

$P = P \cup \{\text{session } j \mid j \in B1 \text{ and } d_j \leq t_{L(i)}\}$

$B1 = B1 - \{\text{session } j \mid j \in B1 \text{ and } d_j \leq t_{L(i)}\}$

Step 8: *Check Schedulability Condition*

(a) if ($\sum_{j \in H} r_j > C$)

{

The sessions are not schedulable

STOP

}

(b) else if ($\sum_{j \in H} r_j = C$)

{

(i) if ($P \neq \emptyset$ OR $B2$ has any session with $d_j \leq t_{L(i)}$)

{

The sessions are not schedulable

STOP

}

(ii) else

{

$C = C - \sum_{j \in H} \rho_j$

Include all sessions that $\in H$ into H_{main}

level++

if ((No. of sessions in H_{main} < Total number of sessions) AND $(L(i) \neq 0)$)

{

* Reinitialize H and i *

$i = 0$

$H = \emptyset$

Update the level of the remaining

sessions in $B1$ and $B2$ to 'level'

GOTO Step 3

}

}

}

Step 9: *Check for the Loop (begun in Step 3) Termination Conditions*

if ((Number of sessions in H_{main} < Total number of sessions) AND $(L(i) \neq 0)$)

GOTO Step 3

Step 10: *Check for the Final Schedulability Conditions*

if ($\sum_j r_j > C$) for $j \in (B1 \cup B2 \cup P \cup H)$

{

The sessions are not schedulable

STOP

}

Step 11: *Termination Condition of the Algorithm*

if ($C_s - \sum_{\substack{session\ j \\ \in\ level\ 1}} r_j < \epsilon$)

STOP

else

{

C_s = sum of rates of sessions

$C = C_s$

GOTO *Step* 1 and re-execute the algorithm with the reduced C

}

Before explaining the steps of the algorithm we state the following Lemma.

Lemma 3.5 *For a fixed value of C, the rates of sessions in a particular iteration $i+1$ $(i \geq 1)$, are $\leq$ their corresponding rates in iteration i.*

Proof: For a session $\in$ set H or P in iteration i, the rate is not recalculated in the subsequent iterations and hence remains the same.

Next consider a session $j \in B1$. Let its rate in iteration i be r_j, and its rate in iteration $i+1$ be g_j. We have to show that $g_j \leq r_j$. Note that a session $j \in B1$ in iteration $i+1$, will have $d_j > t_{L(i)}$, since otherwise it would have been included in set P at the end of iteration i in Step 7. Thus, using Lemma 3.3, the rate in iteration $i+1$ is given by:

$$g_j = \frac{\sigma_j}{\sum_{k=1}^{i}(t_{L(k)} - t_{L(k-1)})\left(\frac{C-\sum_{m=1}^{k-1}\rho_{L(m)}}{C-\sum_{m=1}^{k-1}r_{L(m)}}\right) + (d_j - t_{L(i)})\left(\frac{C-\sum_{m=1}^{i}\rho_{L(m)}}{C-\sum_{m=1}^{i}r_{L(m)}}\right)}$$

The rate in iteration i is given by:

$$r_j = \frac{\sigma_j}{\sum_{k=1}^{i-1}(t_{L(k)} - t_{L(k-1)})\left(\frac{C-\sum_{m=1}^{k-1}\rho_{L(m)}}{C-\sum_{m=1}^{k-1}r_{L(m)}}\right) + (d_j - t_{L(i-1)})\left(\frac{C-\sum_{m=1}^{i-1}\rho_{L(m)}}{C-\sum_{m=1}^{i-1}r_{L(m)}}\right)}$$

It can be shown after some algebra that $g_j \leq r_j$.

Next, consider a session $j \in B2$. A session which remains in $B2$ at the end of the i^{th} iteration has $r_j^{(i)} = \rho_j$. Such a session also over-achieves its delay i.e. $d_j^* < d_j$. Such a session at the end of the $(i+1)^{th}$ iteration either achieves its worst-case delay i.e. $d_j^* = d_j$ or is allotted a rate g_j such that $g_j^{(i)} = \rho_j$. In either case we have $g_j \leq r_j$.

$\diamondsuit$

The basic idea of the algorithm is to allocate rates to sessions such that they do not over-achieve their delays. It does so by taking into account the backlog clearing times of the other sessions. The steps in the algorithm are elucidated below:

The N sessions at the GPS Server are assumed to satisfy the stability criterion, that is $\sum_{j=1}^{N}\rho_j < C$, where C is the server capacity. Every session j is characterized by six attributes: They are:

(1) Burstiness (σ_j)

(2) Average Rate (ρ_j)

(3) Delay Guarantee (d_j)

(4) Allotted Rate (r_j)

(5) Backlogged Time and

(6) level

The final attribute, *level*, needs some explanation. It is possible that some sessions could achieve their delay bounds even when allotted a rate $r_j = 0$. These sessions are not served as long as any of the sessions with non-zero rates are backlogged. However, after all the sessions with non-zero rates clear their backlogs, the excess resources will be allotted to the zero rate sessions and thus ensure a finite worst-case delay bound. Now if there is only one session with a zero rate then it is clear that all the excess bandwidth will be allocated to it. But in case there is more than one session with a zero rate then the excess bandwidth has to be distributed appropriately among these sessions so that their delay guarantees are satisfied. In such a case these sessions belong to a level = 2, in the sense that they are served with the allotted rate only after the first level sessions clear their backlog. Thus, a session could belong to any level ≥ 1 depending on its delay requirements.

In *Step 2* a session j is classified as belonging to a set $B1$ or $B2$ depending on whether $\frac{\sigma_j}{d_j} \geq \rho_j$ or $\frac{\sigma_j}{d_j} < \rho_j$ respectively. As we had seen in Lemma 3.1 the sessions in B1 necessarily have their service rates $\geq$ average rates when the last bit of the burst is being served. By Lemma 3.2 this is not necessarily true for the sessions in $B2$. As the algorithm proceeds the sessions in $B2$ which need a service rate $\geq$ average rate during the service of the last bit of the burst, are separated out and put into $B1$. This classification is necessary since the method of rate calculation in order not to over-achieve the worst case delay, is different for a session $\in B1$ and for that $\in B2$.

Step 4(a) calculates the rates of the sessions $\in B1$ using Lemma 3.3. The sessions in B1 must have their service rates $>$ their average rates when the last bit of the burst is being served (Lemma 3.1). Thus by Corollary 3.1 we know that for such a session j the rate allocation calculated in this step results in the worst case delay, d_j^* = required delay, d_j.

Step 4(b) calculates the rates of the sessions $\in B2$ in three different cases.

The first case is executed only for the first iteration ($i = 1$). In this case a session $\in B2$ is allotted a rate equal to its average rate. At this stage the backlog finish time of no session ($\in (B1 \cup B2)$) has been fixed, and hence such an allocation is necessary in order for a session $\in B2$, not to violate its delay bound.

The second and the third cases are valid only if, atleast one session's backlog finish time is fixed (*i.e.* $i \geq 2$).

The second case is when $d_j \leq t_{L(i-1)}$. Recall that $t_{L(i-1)}$ is the time instant when session $L(i-1)$ finishes its backlog, and it is the $(i-1)^{th}$ to do so. In this case the rates are calculated using Lemma 3.4. However such a rate allocation need not always result in the worst case delay d_j^* being equal to the required delay, d_j. d_j^* is equal to d_j only

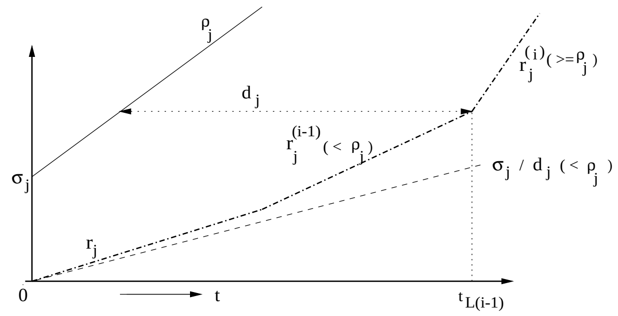

Figure 3.6: The case where the worst case delay is experienced at the break-point, $t_{L(i-1)}$.

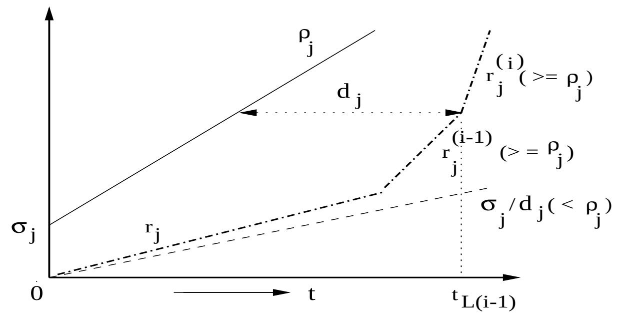

Figure 3.7: A case where the delay d_j at the break-point, $t_{L(i-1)}$ is not the worst-case delay.

if a certain condition is satisfied by the rates $r_j^{(i-1)}$ and $r_j^{(i)}$. There are three possible conditions that arise and are checked in the subsequent statements of *Step* $4(b)$.

The three conditions are illustrated in Figures 3.6, 3.7 and 3.8. Figure 3.6 illustrates the first condition when the session service rate before the break-point $(t_{L(i-1)})$, $r_j^{(i-1)} < \rho_j$ while the service rate after the break-point, $r_j^{(i)} \geq \rho_j$. By Corollary 3.4, we know that the worst-case delay d_j^*, achieved in such a scenario is equal to the required delay, d_j. In this case the rate r_j of session j cannot be further reduced in the subsequent iterations, without violating the delay bound. Hence this session is deleted from $B2$ and included in the set P (P is the set of sessions whose guaranteed rates are frozen but their backlog clearing times are not).

Figure 3.7 illustrates the second condition i.e. when the rate before the breakpoint, $r_j^{(i-1)} \geq \rho_j$. In this case the worst case delay is not d_j and intuitively, the session j does not benefit from the backlog clearing time of session $L(i-1)$, in terms of reducing its rate

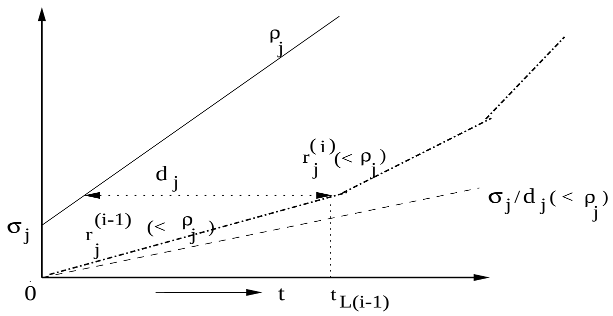

Figure 3.8: A case where the delay d_j at the break-point, $t_{L(i-1)}$ is not the worst-case delay.

allocation. The rate of such a session is updated to $\frac{\rho_j}{(\frac{C-\sum_{k=1}^{i-2}\rho_{L(k)}}{C-\sum_{k=1}^{i-2}r_{L(k)}})}$ and included in the set P.

The third condition illustrated in Figure 3.8 is when the session service rate after the break-point, $r_j^{(i)} < \rho_j$. In this case the worst-case delay is achieved beyond $t_{L(i-1)}$, so the rate of the session is updated to $\frac{\rho_j}{\left(\frac{C-\sum_{k=1}^{i-1}\rho_{L(k)}}{C-\sum_{k=1}^{i-1}r_{L(k)}}\right)}$. In such a case the session over-achieves its delay, so it is allowed to remain in $B2$ for a possible reduction in the rate r_j, in the subsequent iterations.

The third case is when $d_j > t_{L(i-1)}$. In this case the rates are calculated using Lemma 3.3. By Corollary 3.2 we know that the worst case delay, d_j^* is not necessarily equal to d_j. d_j^* is equal to d_j only if the session j service rate when the last bit of the burst is being served, is $\geq \rho_j$, in which case this session is put into set $B1$, otherwise the rate of the session is updated such that it attains its average rate after $t_{L(i-1)}$ and continues to remain in $B2$.

As proved in *Lemma 3.5*, the sessions which remain in $B2$ at the end of *Step 4(b)* in a particular iteration would possibly be allotted lesser rates in the subsequent iterations.

Step 5 computes the backlog finish times for the sessions $\in B1$, $B2$ and P using (3.5) of Section 3.2.

Step 6 chooses the session with the minimum backlog time and includes it in set H (H is the set of sessions whose guaranteed rates and the backlog finish times are frozen). Note that it is possible that the backlog clearing times of all the sessions in ($B1 \cup B2 \cup P$) are

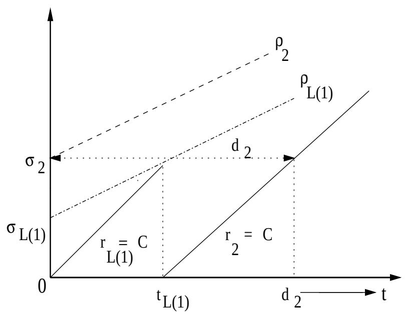

Figure 3.9: Delay guarantees are satisfied with sessions at different levels.

equal to ∞. In such a scenario $L(i) = 0$. If in a particular iteration i, $L(i) = 0$, subsequent iterations of the algorithm will not yield better rate allocations (i.e. lesser rate allocations) for the remaining sessions $\in (B1 \cup B2)$. Hence $L(i) = 0$ is a loop terminating condition, that is checked further ahead.

Step 7 identifies the sessions $\in B1$ which have their delay requirements $\leq t_{L(i)}$. The rates r_j of such sessions have to be frozen since they do not benefit from the backlog clearing time of $L(i)$. For the remaining sessions in $B1$, subsequent iterations will yield a lesser rate allocation as proved in Lemma 3.5.

Step 8 checks the schedulability conditions at the end of one iteration. Note that at this step even if $\sum_{j \in H} r_j = C$, it could still be possible to accommodate the remaining sessions depending on their delay requirements. As mentioned before, these sessions would belong to the next higher level, and their service is begun only after the sessions at the preceding level complete their backlogs (see Figure 3.9). In the example in Figure 3.9, though the sum of the rates at the end of the first iteration is $= C$, session 2 can still be supported. This is because session 2 can begin clearing its backlog at $t_{L(1)}$ and still satisfy its delay bound, d_2. Thus session 2 has a level $= 2$.
The values of C, i and H are re-initialized and the algorithm is continued.

Step 9 checks the termination conditions of the loop. The loop terminates either if all the sessions have been included in H_{main} or if there is no session $\in (B1 \cup B2 \cup P)$ which has a finite backlog finish time ($i.e.$ $L(i) = 0$).

Step 11 checks for the termination condition of the algorithm. At the end of the algorithm, the sum of the allocated rates must be equal to the "assumed" value of C. The algorithm initially begins with $C =$ server capacity. At the end of the first iteration if the sum of session rates, $\sum_{j \in N} r_j < C$ then the remaining bandwidth $C - \sum_{j \in N} r_j$ is assumed to be given to a "dummy" session, with parameters: $\sigma_d = \infty, \rho_d = 0$. This essentially means that the "dummy" session also gets a share of the excess bandwidth released when any session j , $(j \in N)$ completes its backlog period. However, our aim is to find the minimum rates of sessions $1, \ldots, N$ without any assumption of the "dummy" session. In other words, we have to find the minimum capacity C that is required to support the N sessions , when the excess released bandwidth in any iteration i of sessions $L(1), \ldots, L(i-1)$ is distributed among the remaining sessions, $N - \{\cup_{k=1}^{i-1} L(k)\}$ only.

Thus at the end of the execution of the algorithm for the first time, if the difference between the server capacity, C and the sum of session rates is $> \epsilon$, the capacity C is updated to the sum of the session rates. In general, let C_n $(n \geq 1)$ denote the minimum capacity that the n^{th} execution of the algorithm started with. Then, if at the end of the n^{th} execution of the algorithm $C_n - \sum_{j \in N} r_j > \epsilon$, the capacity that the $(n+1)^{th}$ execution starts with is $C_{n+1} =$ sum of session rates at the end of the n^{th} execution, i.e., $\sum_{j \in N} r_j$. The following Lemma proves the correctness of the above method.

Lemma 3.6 *If at the beginning of the $n^{th} (n \geq 2)$ execution of the algorithm the server capacity $C_n =$ sum of session rates at the end of $(n-1)^{th}$ execution of the algorithm then:*
(i) The sum of session rates at the end of the n^{th} execution of the algorithm is $\leq C_n$.
(ii) The algorithm terminates after a finite number of executions.

Proof: (i) It is sufficient to prove that the sessions are schedulable in the n^{th} execution of the algorithm (with server capacity $= C_n$). Suppose that the rates allotted to the sessions in the n^{th} execution are the same as their corresponding rates in the $(n-1)^{th}$ algorithm execution. Let these rates be $r_j; 1 \leq j \leq N$. Then the order in which the sessions complete their backlogs in the $(n-1)^{th}$ and the n^{th} execution is the same, i.e. $L(k); 1 \leq k \leq N$ refer to the same sessions in both the executions.

Consider a session j. The bandwidth received by a session j (which has not cleared its backlog) after k sessions complete their backlogs in the n^{th} execution of the algorithm is given by:

$$r_j + \frac{r_j}{C_n - \sum_{i=1}^{k} r_{L(i)}}(\sum_{i=1}^{k} r_{L(i)} - \sum_{i=1}^{k} \rho_{L(i)}) \tag{3.13}$$

The bandwidth received by j in the $(n-1)^{th}$ algorithm execution after k sessions complete their backlogs is:

$$r_j + \frac{r_j}{C_{n-1} - \sum_{i=1}^{k} r_{L(i)}}(\sum_{i=1}^{k} r_{L(i)} - \sum_{i=1}^{k} \rho_{L(i)}) \tag{3.14}$$

The sessions were schedulable in the $(n-1)^{th}$ algorithm execution.

$\Longrightarrow$ The sum of session rates at the end of the $(n-1)^{th}$ algorithm execution $\leq C_{n-1}$.

$\Longrightarrow C_n \leq C_{n-1}$

$\Longrightarrow$ The expression in (3.13) $\geq$ the expression in (3.14)

Thus, on allotting the same bandwidth for a session in the n^{th} execution of the algorithm as that in the $(n-1)^{th}$ execution, the bandwidth received by a session in the n^{th} execution of the algorithm is $\geq$ that received in the $(n-1)^{th}$ execution of the algorithm.

$\Longrightarrow$ The sessions over-achieve their delays in the n^{th} algorithm execution.

$\Longrightarrow$ The sessions are schedulable when the server capacity $= C_n$

Thus, we already have a rate assignment that ensures schedulability and satisfies delay bounds with server capacity $= C_n$, viz., r_j, $1 \leq j \leq N$. When the algorithm is re-executed with $C = C_n$ and a starting rate allocation r_j, $1 \leq j \leq N$, it will terminate with another allocation r'_j, $1 \leq j \leq N$, say. Now the design of the algorithm ensures that $r'_j \leq r_j$, $1 \leq j \leq N$. Hence $\sum_{i=1}^{N} r'_j \leq \sum_{i=1}^{N} r_j = C_n$.

(ii) $\{C_n\}$ is a monotone decreasing sequence that is bounded below. Therefore $\lim_{n\to\infty} C_n$ must converge to some C. In practice, the algorithm is terminated when $(C_n - C_{n+1})$ becomes sufficiently small.

$\diamondsuit$

3.2.2 Basic Properties

Optimal Bandwidth Allocation

The CAC algorithm allocates bandwidth to the sessions optimally in the sense that it prevents (as far as possible) any session from over-achieving its delay guarantee. It does so by taking into account the excess bandwidth that a session would receive as other sessions clear their backlogs, and allot just enough bandwidth to it so that its delay bound is not violated.

Bandwidth and Delay De-coupled System

Rate Proportional Processor Sharing (RPPS) form a large sub-class of GPS-based schedulers. They have a rate-proportional allocation strategy in which the guaranteed rate of a connection needs to be no less than the connection's average rate and the delay bound for the connection is inversely proportional to the guaranteed rate. In other words the guaranteed rate for a connection j is $\max(\frac{\sigma_j}{d_j}, \rho_j)$. This introduces a coupling between the end-to-end delay bound and the bandwidth provided to each connection. Thus, in order for a connection to get a low delay bound, a large bandwidth needs to be allotted. This results in a waste of resources when the low delay connection also has a low throughput.

GPS-based schedulers with a *General Resource Assignment* or a *Non-Rate Proportional Resource Assignment* do not have such a restriction [22]. The CAC in this paper does such a Non-Rate Proportional Resource Assignment which results in efficient network utilization.

Statistical Multiplexing Effect

Notice that, the fewer the number of sessions at the node, the greater is the rate needed to be allotted for a session j, to satisfy its delay requirement. For example, if session j is the only session at the server then it has to be allotted a rate of $\max(\frac{\sigma_j}{d_j}, \rho_j)$. This rate will possibly reduce as the number of sessions at the node increases and session j gets a share of the released bandwidth after some sessions clear their backlogs.

Table I

Session	σ	ρ	Required Delay (d)	Worst Case Delay d^*	Backlog Time	Allotted Rate	level
1	30	5	2	2	3	15	1
2	50	8	5	5	13.31	8.13	1
3	100	10	8	8	19.36	9.19	1

Support of Best-Effort Traffic

In case there is Best-Effort traffic that has to be supported along with N guaranteed delay sessions, then the algorithm is executed only once. At the end of the first execution if $\sum_{j \in N} r_j < C$, the remaining capacity $C - \sum_{j \in N} r_j$ can be allotted to the Best-Effort traffic without violating the delay requirements of the guaranteed delay sessions.

3.3 Numerical Results

The numerical results are presented in two parts. The first part illustrates how the algorithm allots bandwidth optimally to guarantee the delay requirements of the connections. The second part demonstrates the *Non-Rate Proportional* bandwidth allocation strategy of the Optimal CAC and its advantage over the *Rate Proportional* bandwidth allocation.

3.3.1 Computing the Optimal Bandwidth to support the Connections' Delay Guarantees

The server capacity C is fixed at 100 units. Figures (3.10), (3.11) and (3.12) illustrate the optimal bandwidth allocation for three sessions whose parameters are given in Table I. All the sessions in this case have $\frac{\sigma}{d} \geq \rho$, and hence belong to the set $B1$. Note that the worst case delay, d^* for each of the sessions is equal to the delay guarantee asked for, d. The minimum capacity required to accommodate these sessions is 32.32. Figures (3.13), (3.14) and (3.15) illustrate the case where session 1 belongs to the set $B1$, while the sessions 2 and 3 belong to the set $B2$. The minimum capacity required in this case is

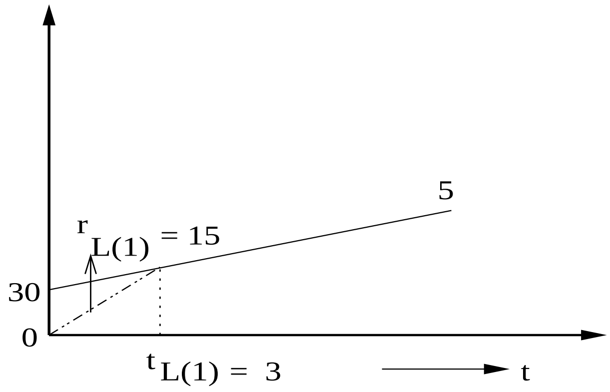

Figure 3.10: Session 1

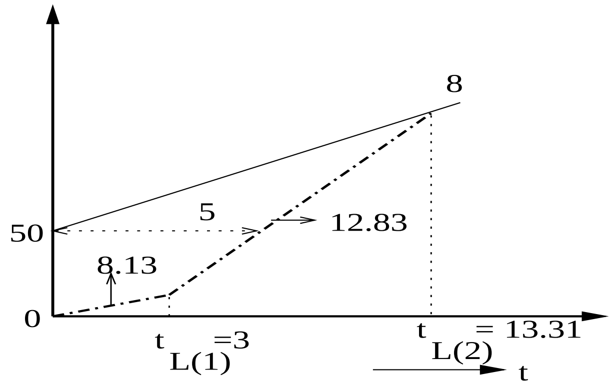

Figure 3.11: Session 2

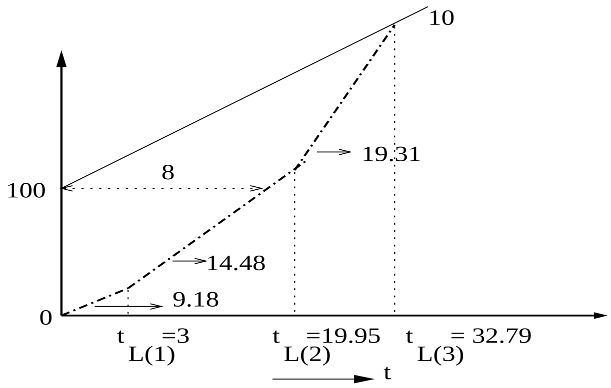

Figure 3.12: Session 3

Table II

Session	σ	ρ	Required Delay (d)	Worst Case Delay d^*	Backlog Time	Allotted Rate	level
1	100	10	5	5	10	20	1
2	50	10	6	6	40.18	9	1
3	100	30	20	8.29	∞	21.04	1

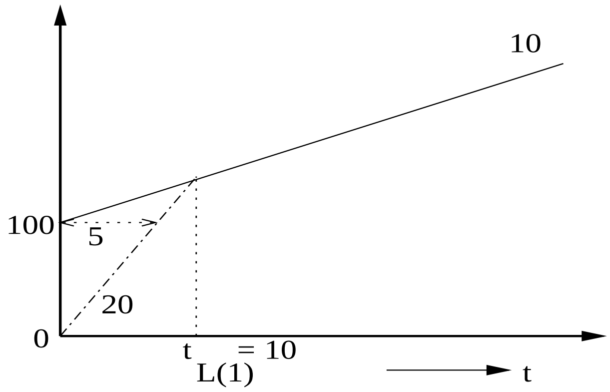

Figure 3.13: Session 1

50.04. The sessions' parameters are given in Table II.

Figures (3.16) and (3.17) illustrate a case where the sessions belong to different *levels* and satisfy their delay requirements. Sessions 1 and 2 belong to level 1 while sessions 3 and 4 belong to level 2. The sessions' parameters are given in Table III.

Table III

Session	σ	ρ	d	d^*	Backlog Time	Allotted Rate	level
1	100	20	2	2	3.33	50	1
2	150	10	3	3	3.57	50	1
3	100	10	7	7	8.788	29.167	2
4	100	5	30	27.62	∞	3.403	2

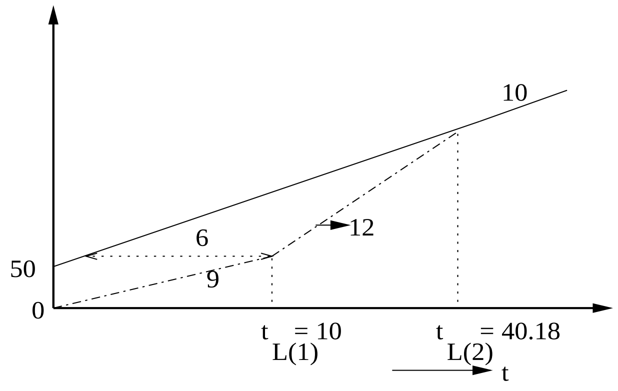

Figure 3.14: Session 2

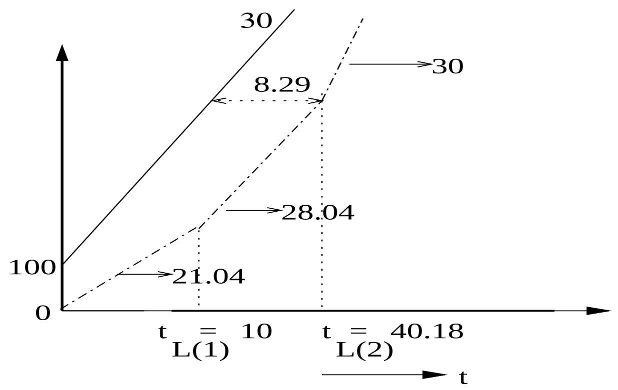

Figure 3.15: Session 3

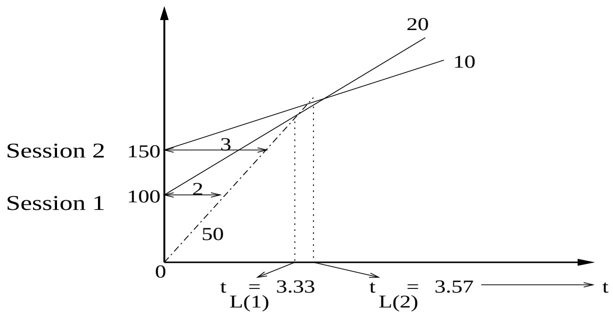

Figure 3.16: Sessions 1 and 2 at level 1

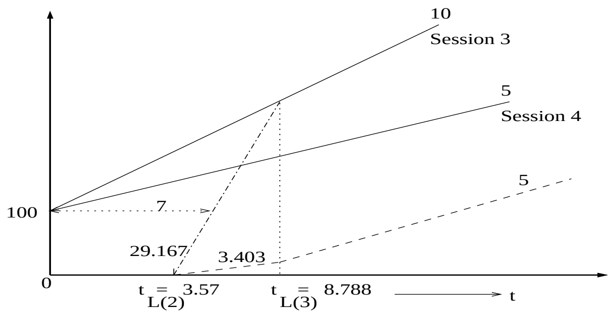

Figure 3.17: Sessions 3 and 4 at level 2

Table IV

Session	σ	ρ	Required Delay (d)
1	100	10	5
2	100	25	2
3	9	2	7.6

3.3.2 General Resource Allocation vs. Rate Proportional Resource Allocation

In this section we demonstrate the greater efficiency of the *General Resource Allocation* in the GPS scheduler as compared to the *Rate Proportional Allocation.* The server capacity, C is fixed at 1000 units. As mentioned before we consider the leaky-bucket constrained sources.

The comparison is carried out as follows. There are three types of traffic (Types 1, 2 and 3) whose input parameters and the delay requirements are fixed as shown in Table IV.

In Figure (3.18) the number of Type-1 connections is fixed while we compute the maximum number of Type-2 and Type-3 connections that can be admitted at the server, using the *Rate Proportional CAC* and the *Optimal non-Rate Proportional CAC.* Similar comparisons are made in Figure (3.19) and Figure (3.20) in which the number of Type-2 and Type-3 connections, respectively, is fixed.

In all the three cases it is clear that the General Resource Allocation results in a higher

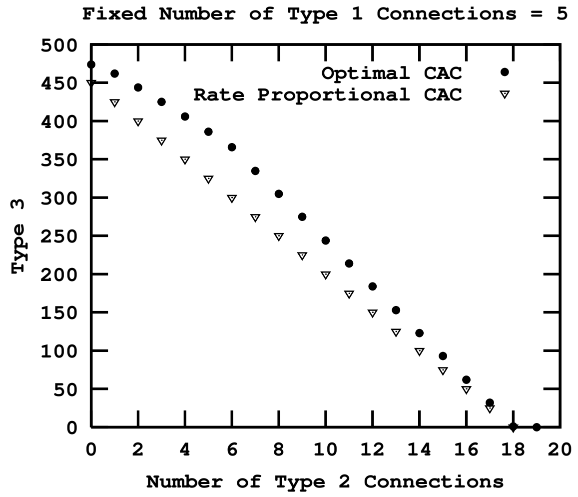

Figure 3.18: Type 3 vs. Type 2

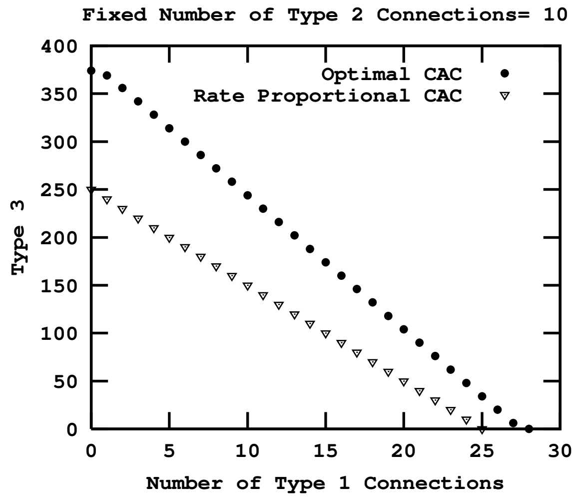

Figure 3.19: Type 3 vs. Type 1

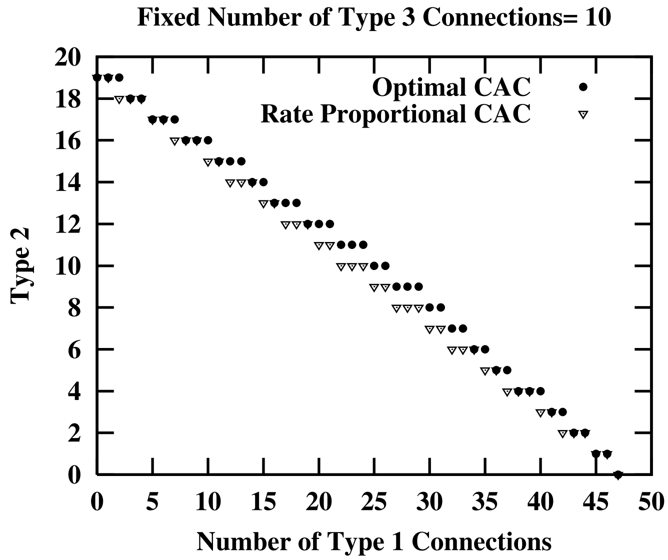

Figure 3.20: Type 2 vs. Type 1

network utilization than the Rate-Proportional Allocation. This difference will be even more prominent as the diversity of the traffic mix increases.

3.4 Computational Complexity of the Optimal CAC

The computation of the optimal bandwidths for an N session GPS system consists of at most N steps. In the i^{th} step, computing the rates of the sessions and their backlog finish times are each of $O(N)$ complex tasks. Sorting the finish times and choosing the smallest one is also done in $O(N)$ time. Thus, the complexity of the i^{th} step is $O(N)$.

Therefore, the complexity of the Optimal CAC for a given N session GPS system is $O(N^2)$.

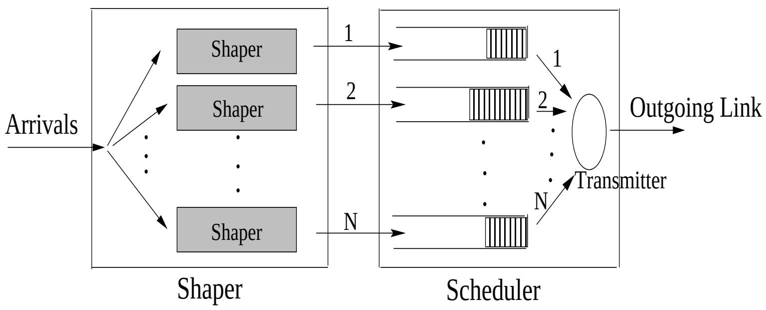

Figure 3.21: Shaped Generalized Processor Sharing Scheduler

3.5 Optimal CAC for Shaped Generalized Processor Sharing (SGPS) Schedulers

3.5.1 Shaped Generalized Processor Sharing Scheduler

Figure 3.21 illustrates the logical structure of the Shaped Generalized Processor Sharing scheduler. The arriving packets first enter a shaper and are admitted into the scheduler when they become eligible. Packets that have not become eligible for service remain in the shaper queue while all eligible packets wait for service in the scheduler queue. The eligibility criterion for admission of a packet into the scheduler queue is decided by the allowable traffic envelope out of the shaper queue. The allowable traffic envelope of a shaper j is a bound on the connection j traffic allowed out of the shaper j and is determined by the policing algorithm that enforces adherence to the arrival characteristics declared by the connection.

An example that motivates the CAC algorithm for SGPS is illustrated in Figure 3.22. Figure 3.22(a) illustrates the session j service curve at the GPS scheduler without any shaper. The session j realizes its worst case delay d_j^* at the time instant τ and completes its backlog period at the time instant τ'. When the rates are allotted according to the CAC in Section 3.2, a session other than j would possibly benefit from the released bandwidth of session j only after time τ'. Note that in Figure 3.22(a) the session j service rate will continue to increase even after the epoch τ, when the worst case delay d_j^* is realized. Thus the subsequent bits served after the epoch τ always experience a delay

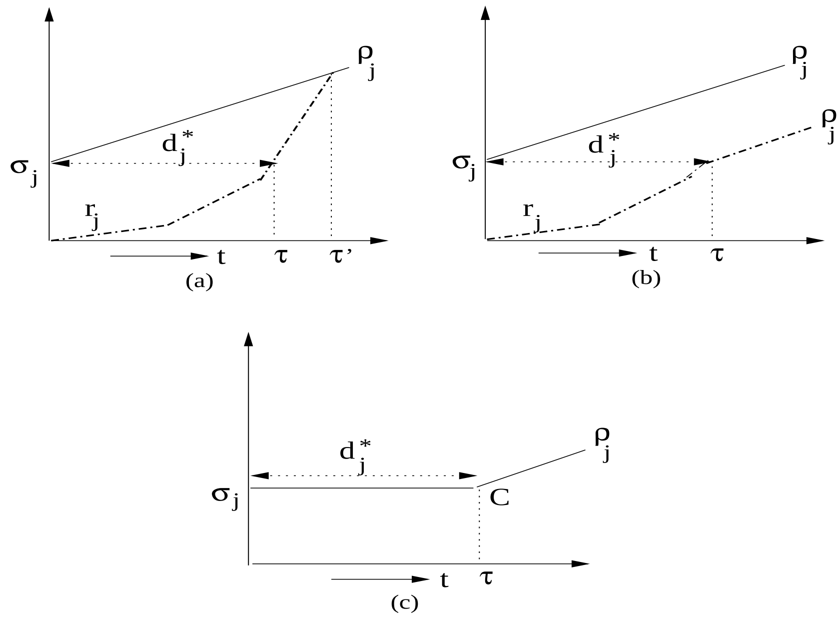

Figure 3.22: (a) Session j service curve without the shaper and with allotted rate r_j. (b) Session j service curve with the shaper and with allotted rate r_j. (c) The shaper output traffic envelope.

$\leq d_j^*$. Serving the bits at a rate ρ_j after the epoch τ will cause the remaining bits (served after τ) to experience the worst-case delay d_j^*. This is illustrated in Figure 3.22(b). This case is equivalent to the session j completing its backlog at time $t = \tau$ and the released bandwidth of session j being available for the remaining sessions from $t = \tau$ itself. Such a scenario can be realized using a shaper along with the GPS scheduler. The shaper envelope should be such that the session j backlog in the scheduler is equal to zero at the epoch that the worst case delay is realized, and it sends traffic at a rate ρ_j thereafter. This envelope is shown in Figure 3.22(c). To the right of point C entire delay d_j^* is at the shaper and there is no delay at the scheduler. In the example considered in Figure 3.22, $\frac{\sigma_j}{d_j} \geq \rho_j$ for session j. Figure 3.23 illustrates an example in which $\frac{\sigma_j}{d_j} < \rho_j$.

3.5.2 The Connection Admission Control Algorithm

The CAC algorithm determines the session rates needed to support their delay guarantees in an *All-Greedy Regime* (i.e. the traffic follows the shaper envelope from time 0). We prove further ahead that for the shaper output traffic characterization the worst case delay,

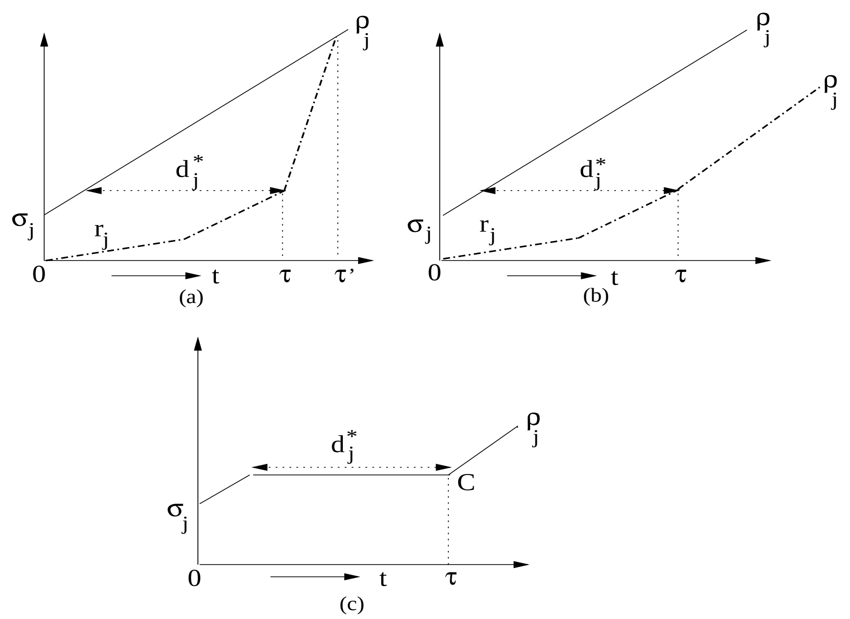

Figure 3.23: (a) Session j service curve without the shaper and with allotted rate r_j. (b) Session j service curve with the shaper and with allotted rate r_j. (c) The shaper output traffic envelope.

d_j^* for any session j is achieved in the All-Greedy Regime. Thus, the rates calculated in this scenario will satisfy the delay bounds of all the sessions, under any other arrival pattern too.

The algorithm runs iteratively, where at the beginning of each iteration the rates of the sessions are calculated to satisfy their delay bounds. The *delay finish time* of a session j is the epoch at which the session realizes its worst case delay d_j^* in the scheduler; also it is the instant when the session j backlog in the scheduler becomes zero. Therefore, the delay finish time of session j is also the backlog clearing time of session j at the scheduler. In each iteration the sessions with the minimum delay finish time are identified. The rates of these sessions are kept fixed in the subsequent iterations of the algorithm. The algorithm proceeds with the next iteration and recalculates the rates of the remaining sessions taking into account the excess released bandwidth of the sessions after the most recent delay finish time (determined in the previous iteration).In general, the algorithm is developed along the same lines as in Section 3.2. Before describing the algorithm we introduce the following notation.

Let t_i denote the minimum delay finish time in the i^{th} iteration of the algorithm. Each iteration of the algorithm decides upon a single delay finish time (the minimum delay finish time); however, there could be several sessions with the same delay finish epoch. Let $N(i)$ denote the number of sessions which have delay finish times $\leq t_i$. $L(k)$ $(1 \leq k \leq N(i))$, indexes the sessions which have their delay finish times $\leq t_i$. Here $N(i)$ is $\geq i$ since at least one session completes its delay finish time at the end of each iteration. Let $r_j^{(i)}$ denote the rate of a backlogged session j during the time interval $[t_{i-1}, t_i]$, $i = 1, \ldots, N$ ($t_0 = 0$ by definition). Then,

$$r_j^{(i)} = \frac{(C - \sum_{m=1}^{N(i-1)} \rho_{L(m)})}{(C - \sum_{m=1}^{N(i-1)} r_{L(m)})} r_j; \; j \in \{1, \ldots, N\} \backslash \{\cup_{m=1}^{N(i-1)} L(m)\} \tag{3.15}$$

The following Lemmas form the basis for the sessions' rate calculations in the algorithm.

The following Lemma calculates the guaranteed rate for a session j such that the last bit of the initial burst experiences a delay d_j.

Lemma 3.7 *Let $t_1, t_2, \ldots, t_{i-1}$ be the sessions' delay finish times which are $< d_j$. Then, the rate r_j needed in order for the last bit of the initial burst of session j to experience a delay d_j is:*

$$r_j = \frac{\sigma_j}{\sum_{k=1}^{i-1} (t_k - t_{k-1}) \left(\frac{C - \sum_{m=1}^{N(k-1)} \rho_{L(m)}}{C - \sum_{m=1}^{N(k-1)} r_{L(m)}} \right) + (d_j - t_{i-1}) \left(\frac{C - \sum_{m=1}^{N(i-1)} \rho_{L(m)}}{C - \sum_{m=1}^{N(i-1)} r_{L(m)}} \right)} \tag{3.16}$$

Proof: Let r_j be the required rate to be allotted and $r_j^{(k)}$ be the session j rate during $[t_{k-1}, t_k]$. Then as given by (3.15):

$$r_j^{(k)} = \frac{r_j}{C - \sum_{m=1}^{N(k-1)} r_{L(m)}} (C - \sum_{m=1}^{N(k-1)} \rho_{L(m)}) \tag{3.17}$$

where the first factor is the proportion of bandwidth that session j receives after sessions $L(1), L(2), \ldots, L(N(k-1))$ complete their backlogs while the second factor is just the available bandwidth in the interval $[t_{k-1}, t_k]$. Then, in order that the last bit of the initial burst experiences a delay d_j, the following must hold:

$$\sigma_j = \sum_{k=1}^{i-1}[(t_k - t_{k-1})\ r_j^{(k)}] \quad + \quad (d_j - t_{i-1})\ r_j^{(i)}$$

The result in (3.16) is easily obtained by substitution of $r_j^{(k)}$ from (3.17) into the above equation.

$\diamondsuit$

The following Corollary is obtained easily from Lemma 3.1 and Lemma 3.7.

Corollary 3.5 *For a session j such that $\frac{\sigma_j}{d_j} \geq \rho_j$, the allocation of the rate in (3.16) results in the worst case delay $d_j^* = d_j$.*

Similarly, the following Corollary follows from Lemma 3.2.

Corollary 3.6 *If $\frac{\sigma_j}{d_j} < \rho_j$, then the rate allocation in (3.16), does not ensure $d_j^* \leq d_j$.*

The following Lemma calculates the rate to be allotted to a session j such that the delay experienced by it at the epoch t_{i-1} is equal to d_j.

Lemma 3.8 *Let $t_1, t_2, \ldots, t_{i-1}$ be the session delay finish times and $d_j \leq t_{i-1}$. Then, in order for the delay d_j to occur at the break-point t_{i-1}, the rate needed is :*

$$r_j = \frac{\rho_j(t_{i-1} - d_j) + \sigma_j}{\sum_{k=1}^{i-1}(t_k - t_{k-1})\left(\frac{C - \sum_{m=1}^{N(k-1)} \rho_{L(m)}}{C - \sum_{m=1}^{N(k-1)} r_{L(m)}}\right)} \tag{3.18}$$

Proof: Let r_j be the required rate to be allotted and $r_j^{(k)}$ be the session j rate during $[t_{L(k-1)}, t_{L(k)}]$. Then as given by (3.15):

$$r_j^{(k)} = \frac{r_j}{C - \sum_{m=1}^{N(k-1)} r_{L(m)}}(C - \sum_{m=1}^{N(k-1)} \rho_{L(m)}) \tag{3.19}$$

where the first factor is the proportion of bandwidth that session j receives after sessions $L(1), L(2), \ldots, L(N(k-1))$ complete their backlogs while the second factor is the available bandwidth itself in the interval $[t_{k-1}, t_k]$. Then, in order that the session j experiences a delay of d_j at some delay finish epoch t_{i-1} the following must hold:

$$\rho_j(t_{i-1} - d_j) + \sigma_j = \sum_{k=1}^{i-1}[(t_k - t_{k-1})\ r_j^{(k)}]$$

The result in (3.18) is easily obtained by substitution of $r_j^{(k)}$ from (3.19) into the above equation.

$\diamondsuit$

The following Corollary follows easily from Lemma 3.1.

Corollary 3.7 *For sessions with $\frac{\sigma_j}{d_j} \geq \rho_j$, the above allocation (3.18) results in the worst-case delay $d_j^* = d_j$ only if $d_j = t_{i-1}$*

The following Corollary follows easily from Theorem 3.1(ii):

Corollary 3.8 *For sessions with $\frac{\sigma_j}{d_j} < \rho_j$, the above allocation results (3.18) in the worst-case delay $d_j^* = d_j$ only if $r_j^{(i-1)} < \rho_j$ and $r_j^{(i)} \geq \rho_j$.*

Algorithm :

Step 1: *Initialization*

$i = 1$ *Iteration Variable*

$level = 1$ *An attribute of a session indicating when the backlog of the session may begin service*

$t_i = 0$, $1 \leq i \leq N$ *Delay finish times of the sessions*

$t_0 = 0$

$C_s = C$ *C_s is the "assumed" sum of rates that the algorithm begins with. C is the server capacity*

The following sets are defined:

$B1 = \emptyset$ *$B1$ will consist of those sessions for which the service rate is greater than their average rate when the last bit of the burst is being served*

$B2 = \emptyset$ $\quad$ *$B2$ will consist of those sessions for which the service rate is not necessarily greater than the average rate when the last bit of the burst is being served*

$H = \emptyset$ $\quad$ *H will consist of sessions at a particular level for which the rates and the backlog finish times are both determined*

$H_{main} = \emptyset$ $\quad$ *H_{main} will consist of the sessions included in the set H at different levels *

Step 2: *Classification of the sessions*

B1 = $\{$session $j \mid \frac{\sigma_j}{d_j} \geq \rho_j\}$

B2 = $\{$session $j \mid \frac{\sigma_j}{d_j} < \rho_j\}$

Step 3: *Rate Calculation for a Session in B1*

For every session j having the minimum delay guarantee d_j, i.e $d_j \leq d_k, \forall\, k \in B1$

$$r_j = \frac{\sigma_j}{\sum_{k=1}^{i-1}(t_k - t_{k-1})\left(\frac{C-\sum_{m=1}^{N(k-1)}\rho_{L(m)}}{C-\sum_{m=1}^{N(k-1)}r_{L(m)}}\right) + (d_j - t_{i-1})\left(\frac{C-\sum_{m=1}^{N(i-1)}\rho_{L(m)}}{C-\sum_{m=1}^{N(i-1)}r_{L(m)}}\right)}$$

Step 4: *Choose the sessions with the minimum delay finish times and include them in set H*

If $B1 = \emptyset$

$t_i = 0$

else

{

$t_i = \min_{j \in B1} d_j$

$H = H \cup \{j \mid d_j = t_i\}$

$B1 = B1 - \{j \mid d_j = t_i\}$

}

Step 5: *Increment the iteration variable*

If $(t_i \neq 0)$

$i = i + 1$

Step 6: *Rate Calculation of sessions $j \in B2$*

if $(i = 1)$

$r_j = \rho_j$

else if $(d_j \leq t_{i-1})$

{

$$r_j = \frac{\rho_j(t_{i-1}-d_j)+\sigma_j}{\sum_{k=1}^{i-1}(t_k-t_{k-1})\left(\frac{C-\sum_{m=1}^{N(k-1)}\rho_{L(m)}}{C-\sum_{m=1}^{N(k-1)}r_{L(m)}}\right)}$$

if $(r_j^{(i-1)} < \rho_j$ and $r_j^{(i)} \geq \rho_j)$

{

$H = H \cup \{j\}$

$B2 = B2 - \{j\}$

}

else if $(r_j^{(i-1)} \geq \rho_j)$

{

$$r_j = \frac{\rho_j}{\left(\frac{C-\sum_{k=1}^{N(i-2)}\rho_{L(k)}}{C-\sum_{k=1}^{N(i-2)}r_{L(k)}}\right)}$$

$H = H \cup \{j\}$;

$B2 = B2 - \{j\}$;

}

else if $(r_j^{(i)} < \rho_j)$

$$r_j = \frac{\rho_j}{\left(\frac{C-\sum_{k=1}^{N(i-1)}\rho_{L(k)}}{C-\sum_{k=1}^{N(i-1)}r_{L(k)}}\right)}$$

}

else if ($d_j > t_{i-1}$)

{

$$r_j = \frac{\sigma_j}{\sum_{k=1}^{i-1}(t_k-t_{k-1})\left(\frac{C-\sum_{m=1}^{N(k-1)}\rho_{L(m)}}{C-\sum_{m=1}^{N(k-1)}r_{L(m)}}\right)+(d_j-t_{i-1})\left(\frac{C-\sum_{m=1}^{N(i-1)}\rho_{L(m)}}{C-\sum_{m=1}^{N(i-1)}r_{L(m)}}\right)}$$

if $(r_j^{(i)} \geq \rho_j)$

{

$B1 = B1 \cup \{j\}$

$B2 = B2 - \{j\}$

}

else

$$r_j = \frac{\rho_j}{\left(\frac{C-\sum_{k=1}^{N(i-1)}\rho_{L(k)}}{C-\sum_{k=1}^{N(i-1)}r_{L(k)}}\right)}$$

}

Step 7: *Check Schedulability Condition*

(a) if ($\sum_{j \in H} r_j > C$)

{

The sessions are not schedulable

STOP

}

(b) else if ($\sum_{j \in H} r_j = C$)

{

(i) if ($B2$ has any session with $d_j \leq t_i$)

{

The sessions are not schedulable

STOP

}

(ii) else

{

$C = C - \sum_{j \in H} \rho_j$

Include all sessions that $\in$ H *into* H_{main}

level ++

if ((Number of sessions in H_{main} < Total number of sessions) AND ($t_i \neq 0$))

{

* Reinitialize H and i *

$i = 1$

$H = \emptyset$

Update the level of the remaining

sessions in $B1$ and $B2$ to 'level'

GOTO Step 3

}

}

}

Step 8: *Check for the Loop (begun in Step 3) Termination Conditions*

if ((Number of sessions in H_{main} < Total number of sessions) AND ($t_i \neq 0$))

GOTO *Step 3*

Step 9: *Check for the Final Schedulability Conditions*

if ($\sum_j r_j > C$) $j \in (B1 \cup B2 \cup H)$

{

The sessions are not schedulable

STOP

}

Step 10: *Termination Condition of the Algorithm*

if ($C_s - \sum_{\substack{session\ j \\ \in\ level\ 1}} r_j < \epsilon$)

STOP

else

{

C_s = sum of rates of sessions

$C = C_s$

GOTO *Step* 1 and re-execute the algorithm with the reduced C

}

Most of the steps of the above algorithm are similar to the steps of the CAC algorithm in Section 3. The steps that are different from the previous algorithm are described below.

Step 3 calculates the rates of the sessions $\in B1$ using Lemma 3.7. The sessions in $B1$ must have their service rates > their average rates when the last bit of the burst is being served (Lemma 3.1). Thus by Corollary 3.7 we know that for such a session j the rate allocation calculated in this step results in the worst case delay, d_j^* = required delay, d_j. Note that in this case it is sufficient to calculate the rate of the session j which has the minimum delay guarantee d_j among all the sessions $\in B1$, since the rates of the remaining sessions are subject to changes in the subsequent iterations of the algorithm, after determination of the minimum delay finish epoch in the current iteration.

Step 4 chooses the sessions with the minimum delay finish epoch and includes them in the set H (H is the set of sessions whose guaranteed rates are fixed). Note that it is possible for the set $B1$ to be empty in which case $t_i = 0$. If in a particular iteration i, $t_i = 0$ subsequent iterations of the algorithm will not yield better rate allocations (i.e lesser rate allocations) for the remaining sessions $\in B2$. Hence $t_i = 0$, is a loop terminating condition (i.e. $B1 = \emptyset$) that is checked further ahead.

Step 6 calculates the rates of the sessions $\in B2$ using Lemma 3.7 and Lemma 3.8 in two different cases. The different conditions encountered here are the same as those in the previous CAC. Note however, that there is no set P here. The sessions are directly included into the set H, since the backlog clearing times are fixed once the rate allocations are fixed. In this step the rates of the sessions $\in B2$ are calculated such that the worst case delay is experienced at the epoch t_{i-1} which is the delay finish time of some session $\in B1$ included in the set H in Step 4. Thus, the sessions $\in B2$ do not have any separate delay finish epochs i.e. their delay finish epochs coincide with that of at least one session $\in B1$. This explains why it suffices to consider only the sessions $\in B1$ while choosing the minimum delay finish time in Step 4.

At the end of Step 6 the sessions which are included in the set H in iteration i from the sets $B1$ and $B2$, have the same delay finish time t_i. This information is used as an input to the connections' shapers to determine the output traffic envelopes of the shapers.

Note that the order of execution of the Steps in this algorithm is different from that in the previous CAC in 3.2. This is essentially because in the present case we know apriori the minimum delay finish time. The minimum delay finish time is always equal to the minimum delay among all sessions $\in B1$. The rates of the sessions $\in B2$ are then calculated based on this minimum finish time. This is unlike the previous CAC where the minimum backlog finish time could be the backlog finish time of any of the sessions $\in B1$ or P. Hence it was necessary to calculate the rates of the sessions $\in B1$ and $B2$ before calculating the session backlog finish times.

The remaining Steps of the algorithm are the same as in the previous CAC.

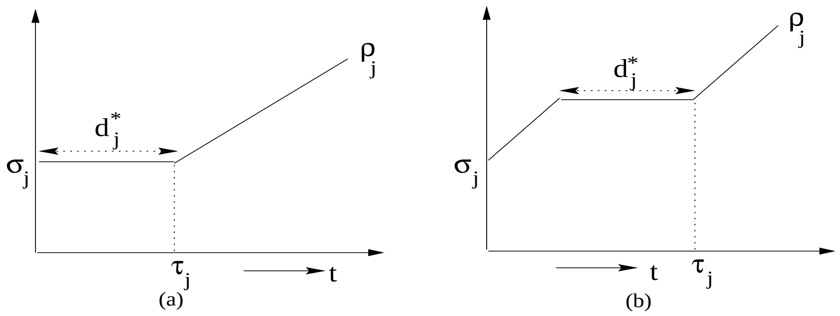

Figure 3.24: (a)The shaper envelope for a session with $\frac{\sigma_j}{d_j} \geq \rho_j$. (b) A shaper envelope for a session with $\frac{\sigma_j}{d_j} < \rho_j$.

3.5.3 Shaper Parameters

The shaper for every connection j needs four input parameters $(\sigma_j, \rho_j, d_j^*, \tau_j)$ to determine the shaper output traffic envelope of session j. (σ_j, ρ_j) are the connection parameters, d_j^* is the worst-case delay of connection j and is $\leq d_j$ while τ_j is the delay finish time of connection j. d_j^* and τ_j are determined from the CAC algorithm. For sessions $\in B1$, $d_j^* = \tau_j = d_j$, while for sessions $\in B2$ these three could be different, in general. Given the four input parameters the traffic envelope, S_j is determined as follows:

$$\begin{aligned} S_j(t_1, t_2] &\leq \sigma_j + \rho_j(t_2 - t_1);\ 0 \leq t_2 - t_1 \leq (\tau_j - d_j^*) \\ &\leq \sigma_j + \rho_j(\tau_j - d_j^*);\ \tau_j - d_j^* \leq t_2 - t_1 < \tau_j \\ &\leq \sigma_j + \rho_j(\tau_j - d_j^*) + \rho_j(t_2 - t_1 - \tau_j); t_2 - t_1 \geq \tau_j \end{aligned} \tag{3.20}$$

An example in Figure 3.24 illustrates the shaper envelopes in (3.20).

We now prove that the worst-case delay for a session j, d_j^*, is achieved in an *All Greedy Regime*, i.e. starting from time 0, the shaper sends in maximum traffic within the output traffic envelope. The motivation for the following Lemmas is from [23] where it was proved that the worst case delay and backlog for leaky-bucket constrained sources are both achieved in an *All Greedy Regime*. As defined in [23] let a *session j busy period* be a maximal interval B_j contained in a single system busy period, such that for all $\tau, t \in B_j$:

$$\frac{W_j(\tau,t)}{W_i(\tau,t)} \geq \frac{\phi_j}{\phi_i}, i = 1, 2, \ldots, N.$$

Note that it is possible for a session to have zero backlog during its busy period. However, if $Q_j(\tau) > 0$ then τ must be in a session j busy period at time τ.

The following Lemma provides a lower bound on the amount of service a session receives when it is in a busy period.

Lemma 3.9 *Assume that session j is in a busy period in the interval $(\tau, t]$. Then, for any subset M of m sessions, $1 \leq m \leq N$ and $t \geq \tau$:*

$$W_j(\tau, t] \geq \frac{(C(t-\tau) - (\sum_{k \notin M} \sigma_k + \rho_k(t-\tau)^*))\phi_j}{\sum_{k \in M} \phi_k} \quad (3.21)$$

$$\begin{aligned} \textit{where } (t-\tau)^* &= t-\tau;\ 0 \leq t-\tau \leq \tau_k - d_k^* \\ &= \tau_k - d_k^*;\ \tau_k - d_k^* \leq t-\tau < \tau_k \\ &= t-\tau-d_k^*;\ t-\tau \geq \tau_k \end{aligned} \quad (3.22)$$

Proof: Let $\phi_{kj} = \frac{\phi_k}{\phi_j}, \quad \forall k, j$

The following inequality is true since the service given to a session k in the interval $(\tau, t]$ is $\leq$ the maximum amount of arrivals in the same interval. Thus,

$$W_k(\tau, t] \leq \sigma_k + \rho_k(t-\tau)^* \ \forall k$$

Also, since the interval $(\tau, t]$ is in session $j's$ busy period:

$$W_k(\tau, t] \leq \phi_{kj} W_j(\tau, t]$$

Thus,

$$W_k(\tau, t] \leq \min\{\sigma_k + \rho_k(t-\tau)^*, \phi_{kj} W_j(\tau, t]\}$$

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

$$C(t-\tau) \leq \sum_{k=1}^{N} \min\{\sigma_k + \rho_k(t-\tau)^*, \phi_{kj} W_j(\tau, t]\}$$

$$\Longrightarrow C(t-\tau) \leq \sum_{k \notin M} \sigma_k + \rho_k(t-\tau)^* + \sum_{k \in M} \phi_{kj} W_j(\tau, t]$$

Rearranging the terms yields (3.21).

$\diamondsuit$

Let $\hat{A}$ denote the arrival process from the shaper into the scheduler in an All Greedy Regime starting at time 0.

Lemma 3.10 *Assume that session j is in a busy period in the interval $(0, t]$. Then, under $\hat{A}$, there exists a subset of sessions, M^t $\forall t \geq 0$ such that the equality holds in (3.21).*

Proof: Since all the sessions are greedy after 0 under $\hat{A}$, every session i will have a session busy period that begins at 0 and lasts up to some e_i. $Q_i(t) = 0$ for all $t \geq e_i$. The system busy period ends at time $e^* = \max_i e_i$. Define

$$M^t = \{i : e_i \geq t\}$$

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

$$W_k(0, t] = \sigma_k + \rho_k(t - d_k^*)$$

In the above equation $t \geq d_k^*$, since $Q_k(t) = 0$ $\forall k \notin M^t$. It is easy to see that equality is achieved in (3.21).

$\diamondsuit$

For every session j, let $\widehat{W}_j(0,t)$ and $\hat{D}_j(t)$ be the session j service and delay functions under $\hat{A}$. $D_j(t)$ denotes the session j delay function under any set of arrival functions $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$ conforming to the envelope specified by $\sigma_j + \rho_j(t-\tau)^*$ for any interval $(\tau, t]$.

Lemma 3.11 *Suppose that time t is contained in a session j busy period that begins at time τ. Then,*

$$\widehat{W}_j(0, t-\tau] \leq W_j(\tau, t]$$

Proof: Define B as the set of sessions that are busy at time $t-\tau$ under $\hat{A}$. Then we have from Lemma (3.9),

$$\begin{aligned} W_j(\tau, t] &\geq \frac{(C(t-\tau) - (\sum_{k \notin B} \sigma_k + \rho_k(t-\tau)^*))\phi_j}{\sum_{k \in B} \phi_k} \\ &= \widehat{W}_j(0, t-\tau], \text{ where the equality follows from Lemma 3.10.} \end{aligned}$$

$\diamondsuit$

Lemma 3.12 *For every session j, d_j^* is achieved when all sessions are greedy starting from time 0, the beginning of a system busy period.*

Proof: Consider any set of arrival functions $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$ conforming to the envelope specified by $\sigma_j + \rho_j(t-\tau)^*$ for any interval $(\tau, t]$. For a session j busy period that begins at time τ, let t^* be the smallest time in that busy period such that

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

From the definition of delay,

$$A_j(\tau, t^*] - W_j(\tau, t^* + D_j(t^*)] = 0$$

Table V

Session	σ	ρ	Required Delay (d)	Rate Prop GPS	Non Rate Prop GPS	Non Rate Prop SGPS
1	30	5	2	15	15	15
2	50	8	5	10	8.13	7.143
3	100	10	8	12.5	9.19	7.33

Then from Lemma 3.11,

$$\widehat{W}_j(0, t^* - \tau + D_j(t^*)] \leq W_j(\tau, t^* + D_j(t^*)]$$

Also,

$$\hat{A}_j(0, t^* - \tau] \geq A_j(\tau, t^*]$$

Thus,

$$\hat{A}_j(0, t^* - \tau] - \widehat{W}_j(0, t^* - \tau + D_j(t^*)] \geq A_j(\tau, t^*] - W_j(\tau, t^* + D_j(t^*)]$$

$$\Longrightarrow \hat{D}_j(t^* - \tau) \geq D_j(t^*)$$

Thus, the session j delay is maximized under an All Greedy Regime beginning at time 0 i.e. d_j^* is achieved when all the sessions are greedy starting from time 0.

$\diamondsuit$

3.5.4 Numerical Example

We consider a Numerical example in this Section. In this example we compare the total allotted bandwidth in the following three cases:

• Rate Proportional Resource Allocation

• Non Rate Proportional Resource Allocation in GPS

• Non Rate Proportional Resource Allocation in Shaped GPS (SGPS) The server capacity is fixed at 100. Table V gives the sessions' parameters and their allotted rates in all the

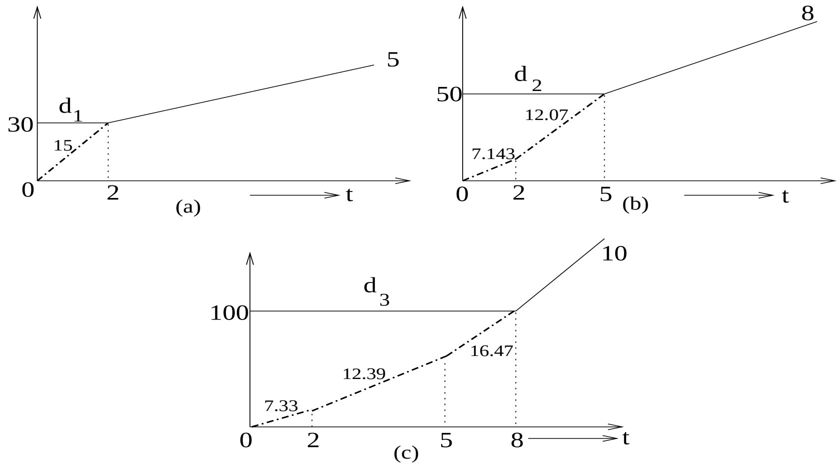

Figure 3.25: (a) Session 1 service curve (b) Session 2 service curve. (c) Session 3 service curve.

three CACs above. The Non Rate proportional allocation in case of Shaped GPS is illustrated in Figure 3.25. It is clear that the sum of the allotted rates is minimized in case of the Non Rate Proportional allocation in Shaped GPS.

3.6 Summary

In this Chapter we have presented an Optimal Call Admission Control algorithm for Generalized Processor Sharing schedulers. The CACs presented in the literature allot bandwidth according to the *Rate Proportional* resource allocation scheme and are based on loose delay bounds. These CACs though simple, would result in a gross over-allocation of bandwidth. The Optimal CAC in this Chapter does a *General Resource* allocation or a *Non-Rate Proportional Resource* allocation and overcomes the above limitations. Its fundamental merit arises from the fact that it takes into account the bandwidth released by the sessions which have completed their backlogs while calculating the sessions' rates. The algorithm allots minimum rates to the sessions such that their delay bounds are not violated. It is shown in the Numerical Results Section that the Optimal CAC can accept significantly more number of connections than the CAC based on *Rate Proportional Allocation*, and hence results in a better network utilization.

We also presented a CAC and the design of shaper parameters for optimal resource allocation in Shaped Generalized Processor Sharing scheduler (SGPS). The shaper in front of the GPS scheduler allows the traffic from a session to enter the scheduler such that it is within a pre-specified traffic envelope (which in general is different for different sessions). Allowing the traffic to enter the scheduler in this manner precludes the scheduler from serving traffic whose delay bounds are not being immediately violated. This bandwidth could instead be used to serve traffic from other sessions. This is unlike the case of a simple GPS scheduler any session that is backlogged in the scheduler is served. The SGPS can thus accept more number of connections than GPS and hence has a greater schedulable region.

Chapter 4

Optimal Call Admission Control in a network of GPS Schedulers

In Chapter 3 we proposed an optimal Non Rate Proportional bandwidth allocation for a single node Generalized Processor Sharing Scheduler. The CAC problem in a network setting is practically more important and has not been addressed in detail in the literature. This problem is non-trivial and has several dimensions apart from those involved in a single-node case, such as the varying connection characteristics as it passes through the nodes in the network and its complex interaction with the other sessions in the network. In this Chapter we address the problem of *optimal Non Rate Proportional bandwidth allocation in a network of GPS schedulers to provide deterministic end-to-end delay guarantees for Leaky-Bucket constrained sources.*

The bandwidth allocation in the network case starts by allocating enough bandwidth at each node such that the end-to-end delays can be guaranteed. The nodal rate allocations are then reduced such that the end-to-end delay guarantees are satisfied with equality, i.e., the end-to-end delays are not strictly smaller than the required values. It is observed that the end-to-end delays can be satisfied with equality by a number of allocations of the nodal bandwidths. The problem of the optimal bandwidth allocation is formulated as a Linear Program.

This Chapter is organized as follows: In Section 4.1 we describe the delay bounds of (σ, ρ) connections in GPS schedulers in the literature and discuss the tightness of the bounds. Section 4.2 discusses the nodal bandwidth allocations to guarantee end-to-end delay in a network of GPS schedulers and in Section 4.3 we present the optimal nodal bandwidth allocations to satisfy the delay guarantee. Summary is in Section 4.4.

4.1 Delay Bounds of Leaky Bucket Constrained Sessions in a network of GPS Schedulers

In this Section we describe the delay bounds obtained in the literature for (σ, ρ) leaky-bucket constrained sources in GPS schedulers. We then discuss the tightness of these delay bounds. Studying the tightness of these bounds is important since the optimal CAC framework for GPS depends on the delay bounds used.
The arrival process of a session j is assumed to be $(\sigma_j, \rho_j) - regular$.

Parekh and Gallager obtained the following bounds for the maximum queueing delay, d_j^* and the maximum backlog, q_j^* for a session j in the single node case [23] and the network case [24]:

$$q_j^* \leq \sigma_j \quad and \quad d_j^* \leq \frac{\sigma_j}{r_j} \tag{4.1}$$

where r_j is the guaranteed rate for session j. Further, it was also shown that in the single node case, any session j in the GPS system achieves its maximum delay and backlog in an *all-greedy regime*. This gives a simple scenario for investigating the worst case behavior. However, the bounds obtained above (4.1) are quite loose since the assumption is that the session j is served with a constant rate r_j till its backlog is cleared. Also, the bound holds only for a sub-class of GPS-based systems called *Rate Proportional Processor Sharing Servers*, in which the guaranteed rate $r_j \geq \rho_j$. In reality, the service rates of backlogged sessions increase as more and more sessions complete their backlogged periods. This is because as each session completes its backlog, the bandwidth released is distributed among the still backlogged sessions in proportion to their weights. The problem of obtaining tighter and more *general* (under Non-Rate Proportional Allocation) delay bounds is addressed in [8] for the single node case and in [24] for the multiple node

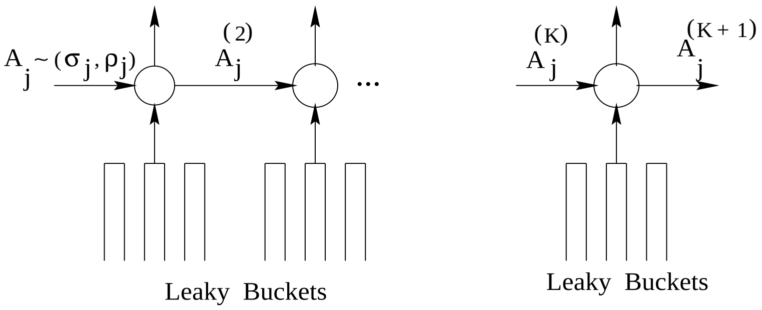

Figure 4.1: Analyzing the session j route as a whole, under the independent sessions relaxation.

case. The method to compute better delay bounds in the single node case was described in Chapter 3. Here we briefly described the method to compute tighter delay bounds in the network case.

Let K be the total number of nodes in the path of connection j. The amount of session j traffic that enters the network in the interval $[\tau, t]$ is given by $A_j(\tau, t)$. Let $S_j^{(k)}(\tau, t), k = 1, \ldots, K$, be the amount of session j traffic served by the k^{th} node in session j path in the interval $[\tau, t]$. Define $I(k)$ to be the set of sessions that are served by server k. For every session $j \in I(k)$, let the arrival function into that node be described by $A_j^{(k)} \sim (\sigma_j^k, \rho_j)$ and the departure function be described by $S_j^{(k)} \sim (\sigma_j^{k.out}, \rho_j)$. $\hat{A}_j(0, t)$ denotes the arrival process of session j at the network ingress when it is greedy starting from time 0. The rate of session j at node k is denoted by $r_{j,k}$. Figure 4.1 illustrates the system that is analyzed. The following assumptions are made:

(i) The sessions $i \in I(k) - \{j\}$ (for $k = 1, 2, \ldots, K$) are free to send traffic in any manner as long as $A_i^{(k)} \sim (\sigma_i^k, \rho_i)$. Thus the sessions in $I(k) - \{j\}$ are called the *Independent sessions* at node k $(k = 1, 2, \ldots, K)$.
(ii) Session j traffic is constrained to flow along its route so that

$$A_j^{(k)} = S_j^{(k-1)}, k = 2, 3, \ldots, K$$

Assumptions 1 and 2 are collectively called the *independent sessions relaxation.*

The worst case delay d_j^* and backlog q_j^* for a session j are achieved in a *staggered greedy* regime. In this pattern of arrivals, the independent sessions at a particular node

k become greedy simultaneously, but only after the independent sessions at node $(k-1)$ become greedy. It is found that d_j^* and q_j^* are in general achieved under different staggered greedy regimes. In order to obtain a unified framework to determine d_j^* and q_j^*, Parekh and Gallager describe a function called the session j Universal Service Curve (USC).

The USC is constructed without computing any staggered greedy regimes and both d_j^* and q_j^* can be determined efficiently and exactly from it (under the independent sessions relaxation). The function $\hat{S}_j^k$ denotes the service curve of session j at the k^{th} node in its path under the all-greedy regime. This function can be computed using the internal input traffic characterization of [24] at the node k and the rate of session j at node k, $r_{j,k}$. $\hat{S}_j^{(k)}$ is continuous, piece-wise linear and is convex-$\cup$ in the range $[0, t_k^B]$, where t_k^B is the duration of the session j busy period at node k under the *all greedy regime*. Thus, $\hat{S}_j^{(k)}$ can be specified (in the range $[0, t_k^B]$) by a list of pairs:

$$(s_1^k, d_1^k), (s_2^k, d_2^k), \ldots, (s_{n_k}^k, d_{n_k}^k)$$

where s_l^k is the slope of the l^{th} line segment, d_l^k is its duration and n_k is the number of line segments. Here

$$s_1^k < s_2^k < \cdots < s_{n_k}^k$$

and

$$\sum_{l=1}^{n_k} d_l^k = t_k^B$$

We briefly describe the construction of the USC from $\hat{S}_j^{(1)}, \ldots, \hat{S}_j^{(K)}$. Let E_j^k be the collection of all the pairs (s_j^m, d_j^m) for $m = 1, 2, \ldots, k$ for the session j i.e.,

$$E_j^k = \bigcup_{m=1}^{k} \bigcup_{l=1}^{n_m} \{(s_l^m, d_l^m)\}$$

The session j USC, $U_j(t)$ is defined as

$$U_j(t) = \min\{G_j^K(t), \hat{A}_j(0,t)\} \tag{4.2}$$

where the curve $G_j^k(t)$ (for $k = 1, 2, \ldots, K$) is a continuous curve constructed from the elements of E_j^k as follows:

(1) Set $G_j^k(0) = 0$, Remaining-in-E $= E_j^k$; Glist $= \emptyset; u = 0; t = 0$.

(2) Order the elements of E_j^k in increasing order of slope. Remove from Remaining-in-E an element of smallest slope: $e^{new} = (s^{new}, d^{new})$, i.e., Remaining-in-E = {Remaining-in-E $\backslash e^{new}$}. Append e^{new} to Glist. If Remaining-in-E is not empty then repeat Step 2.

(3) G_j^k is a piecewise linear convex-$\cup$ function defined in the range $[0, \sum_{m=1}^{k} t_m^B]$ by the elements of Glist. For $t \geq \sum_{m=1}^{k} t_m^B$ set

$$G_j^k(t) = G_j^k(\sum_{m=1}^{k} t_m^B) + \hat{A}_j(\sum_{m=1}^{k} t_m^B, t)$$

The following Theorem in [24] shows how to compute the worst case delay d_j^* and backlog, q_j^* from the session j USC.

Theorem 4.1 *For every session j:*

$$q_j^* \leq \max_{\tau \geq 0} \{\hat{A}_j(0, \tau) - G_j^K(\tau)\} \quad \textit{and}$$
$$d_j^* \leq \max_{\tau \geq 0} \left\{ \min\{t : G_j^K(t) = \hat{A}_j(0, \tau)\} - \tau \right\}$$

To find the bound on d_j^*, the maximum horizontal distance between the curves $\hat{A}_j(0, t)$ and $U_j(t)$ is computed. Similarly, q_j^* is bounded by the maximum vertical distance between the two curves. It is further shown in [24] that these bounds are achieved for a particular staggered greedy regime (different staggered greedy regimes for d_j^* and q_j^*), under the independent sessions relaxation.

Observe that for a one node network case ($K = 1$), the session j USC is the session j service function curve under an all-greedy regime. Further, the result of the worst-case delay bound in Theorem 4.1 specializes to that of Theorem 3.1. The delay bound in Theorem 4.1 for a session j is not tight due to the independent sessions relaxation. One of the assumptions is that for any node k in the path of session j, the sessions $i \in I(k) - \{j\}(k = 1, 2, \ldots, K)$ are free to send traffic in any manner as long as $A_i^{(k)} \sim$

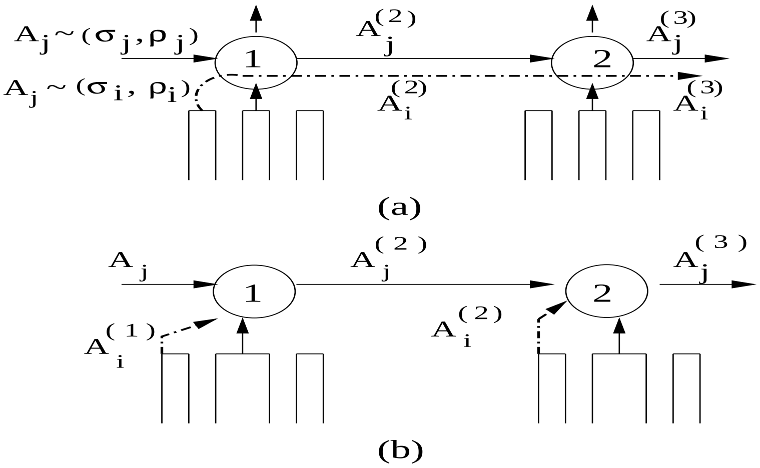

Figure 4.2: (a) Session i passes through nodes 1 and 2 along session j path. (b) Session i is modeled as an independent session at nodes 1 and 2

(σ_i^k, ρ_i). This assumption includes certain arrival functions $A_i^{(k)}$ which are consistent with (σ_i^k, ρ_i), but which may not otherwise be possible due to the network topology. Figure 4.2 illustrates an example where we have to compute the delay bound for session j. Session i passes through the nodes 1 and 2 along the path of connection j. By the independent session assumption, it is modeled as an independent session at nodes 1 and 2. The looseness of the delay bound becomes more prominent as the number of sessions that share more than one common node with session j increases. On the other hand, the delay bounds could be tight when every session's path overlaps with that of session j for atmost one node.

4.2 Nodal Bandwidth Allocations to Guarantee End-to-End Delay

In the this Section we assume that every route in the network and the union of all routes does not result in cycles being induced in the network topology, i.e., the network is acyclic. The presence of cycles can complicate the analysis of delay and can lead to feedback effects that drive the system towards instability [22]. In the case of non acyclic GPS networks there is an additional condition necessary for the assignment of $\{\phi\}$ called the *Consistent*

Relative Session Treatment (CRST) in order to deal with the effects of virtual feedback [24]. No such condition is necessary in the case of acyclic GPS networks. We deal with the case of non acyclic networks briefly in the next Section.

The Connection Admission Control (CAC) for GPS scheduling used in the literature to provide deterministic delay guarantees to sessions is based on the delay bound (4.1). In this simple CAC a connection j is allotted a bandwidth of $\max(\frac{\sigma_j}{d_j}, \rho_j)$ at all the nodes along its path. The nice part about this CAC is that it completely eliminates interference among sessions, as if there were firewalls in between those individually reserved bandwidth channels. Thus, irrespective of the arrival patterns of other sessions, connection j will be assured a bandwidth of at least $\max(\frac{\sigma_j}{d_j}, \rho_j)$. However an incoming session j is refused if this bandwidth is not available at any one of the nodes in the path. As we had seen in Section 4.1 such an allocation could result in the actual worst-case delay being far less than the tolerable delay d_j, and thus could lead to a waste of the allotted bandwidth.

In this Section we address the problem of finding the optimal nodal bandwidth allocations for *a* session j given that there are a set of connections that are already established in the network. The rates and the delay guarantees of the existing sessions in the network should not be violated. At each of the nodes in the network we assume that an elastic *dummy* session is present with $(\sigma_d = \infty, \rho_d = 0)$, that is allotted the residual rate at the node (i.e., Server capacity at the node $-$ sum of the allotted rates of the existing sessions at the node). The dummy session is like a reservoir from which the "rates can be borrowed" and to which the "rates can be returned", while keeping the rates and the delay guarantees of the other sessions (the ones of interest) unchanged. The results of this Section are used in the subsequent Sections where we address the problem of finding the minimum capacity required at each of the nodes in the network to support a given *set* of connections.

In the example in Figure 4.3 session j is allotted a bandwidth of $\frac{\sigma_j}{d_j}$ at nodes 1 and 2 in its path. The worst case delay d_j^* computed from the session j USC is $<$ the required delay d_j. This is generalized in the following Lemma.

Lemma 4.1 *If the bandwidth allotted to a session j at every node in its path is* $\max(\frac{\sigma_j}{d_j}, \rho_j)$

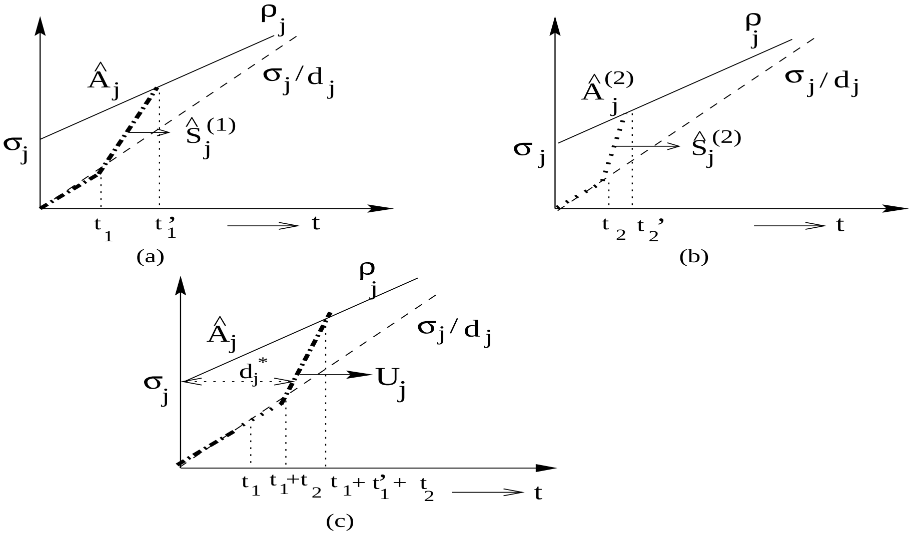

Figure 4.3: (a) Session j service curve at node 1. (b) Session j service curve at node 2. (c) Session j Universal Service Curve, U_j.

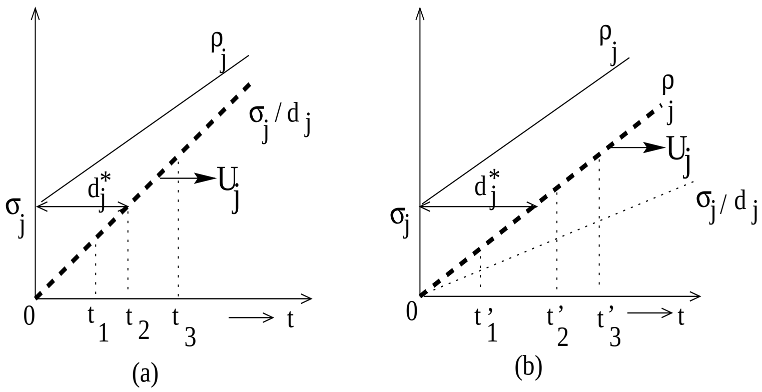

Figure 4.4: (a) Session j USC when $\max(\frac{\sigma_j}{d_j}, \rho_j) = \frac{\sigma_j}{d_j}$. (b) Session j USC when $\max(\frac{\sigma_j}{d_j}, \rho_j) = \rho_j$

then the worst case end-to-end delay $d_j^* \leq d_j$.

Proof: In the worst possible scenario the session j USC is as shown in Figure 4.4. Since the minimum slope of any segment of any session j service curve, in an all-greedy regime is given by $\max(\frac{\sigma_j}{d_j}, \rho_j)$, the Figures follow easily from the construction of the USC.

In any other case the service rates of session j increase as other sessions complete their backlogged periods. In this case the worst-case delay of session j can only reduce.

$\diamond$

Let s_t denote the slope of the segment in the session j USC at epoch t. Then, the following

Lemma gives a direct method to obtain the worst-case delay from the session j USC.

Lemma 4.2 *Let $U_j(t)$ be a USC (defined in (4.2)) and let d_σ denote a time-epoch such that $U_j(d_\sigma) = \sigma_j$. Then:*

(i) If

$$s_{d_\sigma} \geq \rho_j$$

then

$$d_j^* = d_\sigma$$

(ii) If $s_{d_\sigma} < \rho_j$ then there exists an i such that:

$$\forall t < t_i : s_t < \rho_j$$

and

$$\forall t \geq t_i : s_t \geq \rho_j;$$

moreover,

$$d_j^* = t_i - \frac{U(t_i) - \sigma_j}{\rho_j}$$

Proof: By Theorem 4.1 the maximum delay of session j is bounded by the maximum horizontal distance between the session j USC and its arrival function at the network ingress. Parts (i) and (ii) of the above Lemma follow easily from this.

$\diamondsuit$

From the above Lemma, it is clear that the worst-case delay is obtained either at d_σ or at one of the break-points $t_1, t_2, \ldots$ in the USC.

The following Lemmas hold when the allotted rates of a session j in the network satisfy its delay requirement.

Lemma 4.3 *If for a session j, $\frac{\sigma_j}{d_j} \geq \rho_j$, and the delay requirement is satisfied, then,*
(i) $s_{d_\sigma} \geq \rho_j$ where d_σ is as defined in Lemma 4.2.
(ii) The maximum delay is experienced by the last bit of the burst.

Proof: (i) If the allotted rates at all the nodes in the network are $r_{j,k} = \frac{\sigma_j}{d_j};\ 1 \leq k \leq K$, then it is clear that the service rate at d_σ is $\geq \rho_j$.

If the allotted rate at any one of the nodes in the network is $< \frac{\sigma_j}{d_j}$, then in order not to violate the delay bound d_j, it is necessary that s_{d_σ} be $> \frac{\sigma_j}{d_j}$ and hence $> \rho_j$.
(ii)This is a direct consequence of Theorem 4.1(ii) and the part(i) above.

$\diamondsuit$

Lemma 4.4 *If for a session j, $\frac{\sigma_j}{d_j} < \rho_j$ and the delay requirement of j is satisfied, then neither (i) nor (ii) of Lemma 4.3 is necessarily true.*

Proof: Consider the following situation. Suppose that in the session j USC, $\frac{\sigma_j}{d_j} \leq s_{d_\sigma} < e_j$. It is still possible that the delay requirement is not violated. In such a case, by Lemma 4.2(ii) the worst case delay is experienced at some break-point t_j in the USC ($t_j > d_\sigma$), in which case neither (i) nor (ii) of Lemma 4.2 holds.

$\diamondsuit$

The question that we address now is: how should the bandwidths be allotted at different nodes in the path of session j such that it does not over-achieve its delay requirement?

We assume that the maximum bandwidth that is allotted at any node in the session j path is $\max(\frac{\sigma_j}{d_j}, \rho_j)$. The following procedure allots the nodal bandwidths for a session j such that its worst-case delay $d_j^* = d_j$.

1) Allot a rate of $\max(\frac{\sigma_j}{d_j}, \rho_j)$ at each of the nodes in the path of session j.
2) Obtain the session j USC from the *session j nodal service curves.*

  If $(d_j^* = d_j)$

    END

  else

    Select any node k $(1 \leq k \leq K)$ to reduce the session j allotted rate.

3) Reduce the session j rate at node k such that the maximum end-to-end delay computed from the USC $= d_j$.

Step 1) allots a uniform bandwidth of $\max(\frac{\sigma_j}{d_j}, \rho_j)$ at all the nodes in the path of session i. By Lemma 4.1 we know that such an allocation will always satisfy the end-to-end delay requirement.

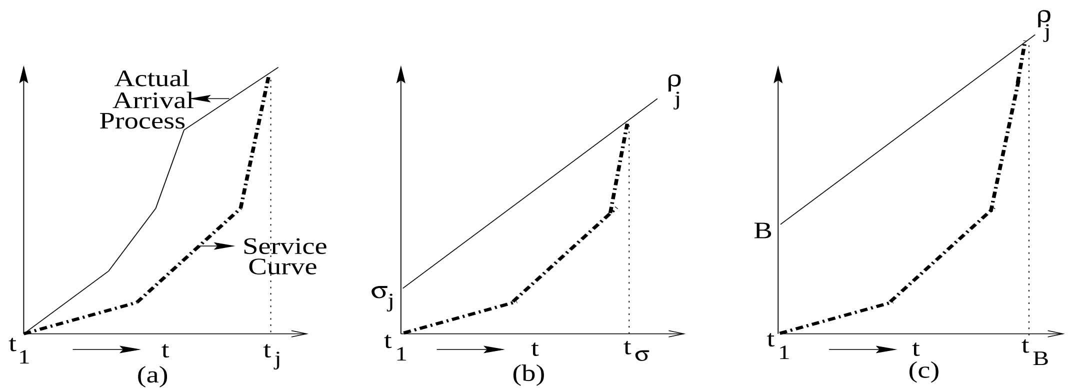

Figure 4.5: (a) Session j service curve for an actual arrival process. (b) Session j service curve for (σ_j, ρ_j) input process. (c) Session j service curve for (B, ρ_j) input process.

In Step 2) the *session j nodal service curves* are obtained with an input burst size of σ_j to the first node and a burst of size B at all the other nodes. The burst size B is defined as follows:

Definition 4: Let $\mathcal{R} = \{(r_{j,1}, r_{j,2}, \ldots, r_{j,K}) : d_j^* \leq d_j\}$ and $\bar{r}$ represents an element of R. $\mathcal{B}$ represents the set of input burstiness values for each allocation $\bar{r} \in R$, i.e. $\mathcal{B} = \{(b_{j,1}, b_{j,2}, \ldots, b_{j,K}) : \bar{r} \in \mathcal{R}\}$ where $b_{j,k}$ $(1 \leq k \leq K)$ is the burstiness of the input traffic characterization of session j into node k. The parameter B is defined as $B = \max_k(b_{j,1}, \ldots, b_{j,k}, \ldots, b_{j,K})\ \forall (b_{j,1}, \ldots, b_{j,K}) \in \mathcal{B}$.

It is not enough to consider the burstiness values of σ_j at the $2^{nd}, 3^{rd}, \ldots, K^{th}$ nodes. The following example motivates the need for B. Consider a session j passing through a network of two nodes. Let the session j be allotted a rate less than ρ_j at the first node in its path. Now suppose that all the sessions at node 1 become greedy at time 0. Since the allotted rate for session j at node 1 is $< \rho_j$, this results in an accumulation of backlog of the session j at node 1 till some epoch t_1. This accumulated backlog is $> \sigma_j$. Now, let the sessions at node 2 become greedy at the epoch t_1. Denote the session j backlog period duration at the node 2 by t_j as shown in Figure 4.5(a) (Note that in this case the exact session j traffic characterization from node 1 is being used). Let t_σ denote the session j backlog period duration at node 2 when we consider the session j input traffic characterization at the ingress of node 2 as (σ_j, ρ_j) as shown in Figure 4.5(b). Let t_B denote the session j backlog period duration when the session j input traffic characterization at the ingress of node 2 is (B, ρ_j) as shown in Figure 4.5(c). It can be seen from Figures 4.5(a) and 4.5(b) that $t_\sigma < t_j$. In this case it is possible that the USC constructed from the nodal service curves that assume a (σ_j, ρ_j) input characterization could under-estimate the worst case delay. On the other hand t_B is an upper bound to the session j backlog period at node 2. Hence it is necessary to consider a burst size of B at the ingress of the interior nodes in the network.

From Definition 4 the burst size B is the largest burst value in the input traffic characterization to any of the nodes k $(1 \leq k \leq K)$ in the network, under all possible rate allocations satisfying the delay guarantees. For all the examples in this paper we choose B to be a conveniently large value, since determining B exactly could be rather difficult.

Our aim here is to find a rate vector $(r_{j,1}, r_{j,2}, \ldots, r_{j,K})$ such that the worst-case delay d_j^* is equal to d_j. The actual internal (σ, ρ) traffic characterization at each of the nodes depends on the vector of rates $(r_{j,1}, r_{j,2}, \ldots, r_{j,K})$ allotted at the nodes in the network. Such an internal traffic characterization is obtained using the procedure in [24].

In case there is room for further increasing the worst-case delay, Step 3 reduces the session j allotted rate at any one of the nodes k in the network such that the d_j^* computed from the resulting USC is equal to d_j. The resulting set of allotted rates $\bar{r}$ at the end of Step 3 could be pessimistic (i.e. greater than the actual allocation necessary to achieve $d_j^* = d_j$) because of the assumption of a burst size of B in the input traffic characterization at all the internal nodes in the network. In order to rectify this, the actual internal (σ, ρ) traffic characterization under the rate vector $\bar{r}$ is obtained and the nodal service curves are re-constructed with the new input traffic characteristics. The session j USC is re-constructed with the new nodal service curves. The session j USC constructed with the rate vector $\bar{r}$ and the internal traffic characterization as (B, ρ), is denoted by $U_{j,\bar{r}}^B$ and the USC constructed with the exact internal (σ, ρ) traffic characterization is denoted by $U_{j,\bar{r}}$. Then we have the following Lemma.

Lemma 4.5 *The worst case delay computed from the USC $U_{j,\bar{r}}^B$ is $\geq$ the worst case delay computed from $U_{j,\bar{r}}$.*

Proof: The maximum rate that can be allotted at any node in the session j path is

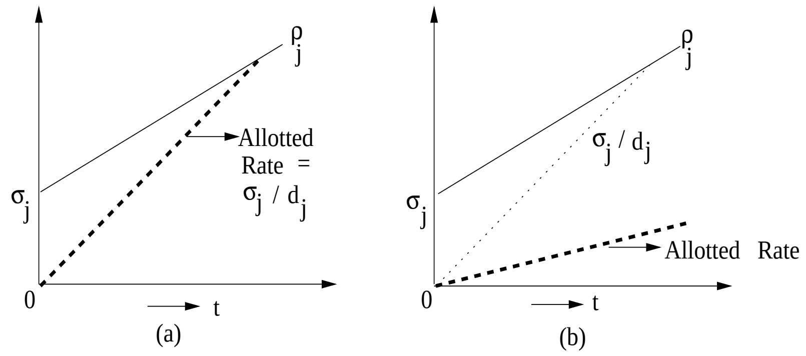

Figure 4.6: In this Figure $\max(\frac{\sigma_j}{d_j}, \rho_j) = \frac{\sigma_j}{d_j}$ (a) Maximum Backlog $= \sigma_j$ when the allotted rate $= \frac{\sigma_j}{d_j}$. (b) Maximum Backlog $> \sigma_j$ when the allotted rate $< \frac{\sigma_j}{d_j}$.

$\max(\frac{\sigma_j}{d_j}, \rho_j)$. Assume that the arrival process into an interior node is the all greedy process at the network ingress. Then, the maximum backlog is equal to σ_j when the allotted rate at the interior node is $\max(\frac{\sigma_j}{d_j}, \rho_j)$. This backlog could be greater than σ_j when the allotted rate is less than $\max(\frac{\sigma_j}{d_j}, \rho_j)$ as illustrated in Figure 4.6. The burstiness of session j in the internal traffic characterization is taken as the maximum session backlog at the preceding node (or the upstream node) in its path. Thus, the burstiness in the internal traffic characterization of the session j is $\geq \sigma_j$. We now consider the following two cases:

Case(i) $\frac{\sigma_j}{d_j} \geq \rho_j$

A session j nodal service curve constructed with the actual (σ, ρ) traffic characterization and that constructed with the (B, ρ) characterization is the same till at least σ_j amount of traffic is served since $B \geq \sigma_j$. Any segment in the session j USC below the σ_j line also appears below the σ_j line in its respective nodal service curve. This is clear from the construction of a USC. Thus the USC $U^B_{j,\bar{r}}$ and $U_{j,\bar{r}}$ are the same till the σ_j line atleast, i.e. if d_σ denotes an epoch such that $U_{j,\bar{r}}(d_\sigma) = \sigma_j$, then:

$$U_{j,\bar{r}}(t) = U^B_{j,\bar{r}}(t); \quad t \leq d_\sigma$$

By Lemma 4.3 we know that the worst case delay when $\frac{\sigma_j}{d_j} \geq \rho_j$ is d_σ. Thus, in this case the worst-case delay computed from the USC $U_{j,\bar{r}}$ and $U^B_{j,\bar{r}}$ are equal.

Case(ii) $\frac{\sigma_j}{d_j} < \rho_j$

In this case we know by Lemma 4.4 that the worst case delay is not necessarily d_σ. In Case(i) we said that $U_{j,\bar{r}}(t)$ and $U^B_{j,\bar{r}}(t)$ are equal for $t \leq d_\sigma$. Let t_u be defined as:

$$t_u = \{\max\ t : U_{j,\bar{r}}(t) = U^B_{j,\bar{r}}(t)\}$$

If the worst case delay occurs before t_u, the d^*_j computed from both the USCs is the same; otherwise, the d^*_j computed from $U^B_{j,\bar{r}}$ is $>$ d^*_j computed from $U_{j,\bar{r}}$ since,

$$U^B_{j,\bar{r}}(t) < U_{j,\bar{r}}(t)\ \forall\ t > t_u \tag{4.3}$$

(4.3) follows directly because of the following points:

(i) The USC is constructed from the session j nodal service curves by appending the service curve segments in increasing order of their slopes.

(ii) The service curve segments used to construct $U_{j,\bar{r}}$ form a subset of those used to construct $U^B_{j,\bar{r}}$.

◇

Note that the proof of Lemma 4.5 suggests that when $\frac{\sigma_j}{d_j} \geq \rho_j$ the rate allocation (such that $d^*_j = d_j$) using the (B, ρ) traffic characterization is not a pessimistic allocation and hence no rectification using the actual traffic characterization is necessary.

The following example illustrates the procedure to allot bandwidth to a session j such that $d_j = d^*_j$. Session j with $(\sigma_j, \rho_j, d_j) = (50, 5, 5)$ passes through nodes 1 and 2. It is allotted a rate $r_{j,1} = r_{j,2} = \max(\frac{\sigma_j}{d_j}, \rho_j) = 10$ at both the nodes. The session j nodal service curves with these allotted rates and the corresponding USC are shown in Figure 4.7. As seen from the USC, the worst-case delay is $4.67 <\ d_j$ and hence there is room to further increase the worst-case delay and hence decrease the allotted resources. One of the ways to achieve a worst-case delay of d_j is to reduce the session j rate at the second node, as shown in the USC in Figure 4.8(a). The worst case delay is equal to the required delay and the total bandwidth allotted at nodes 1 and 2 = 6 + 10 = 16. An alternative

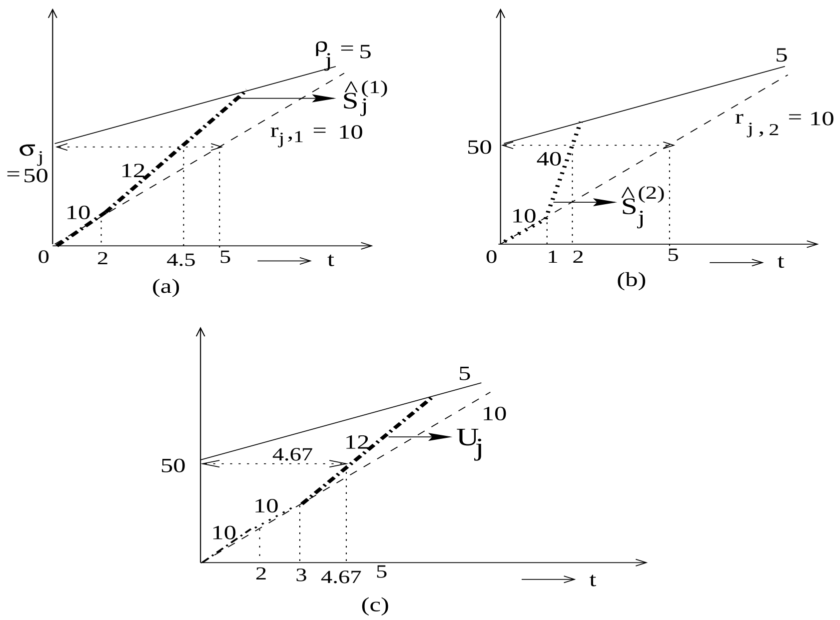

Figure 4.7: (a) Session j service curve at node 1. (b) Session j service curve at node 2. (c) Session j Universal Service Curve. (The numbers beside the line segments are their slopes).

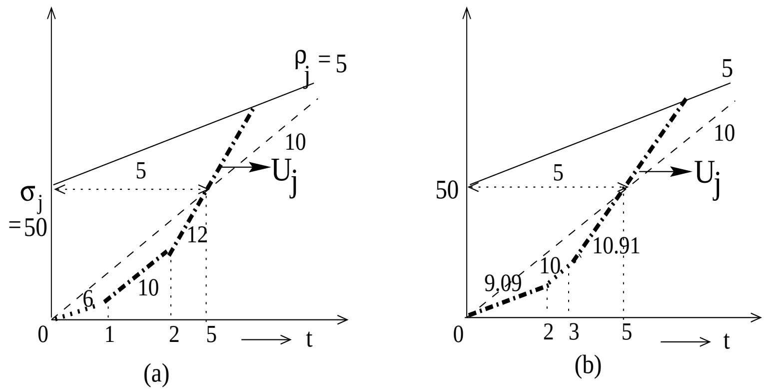

Figure 4.8: (a) Session j USC with $r_{j,1} = 6, r_{j,2} = 10$. (b) Session j USC with $r_{j,1} = 10, r_{j,2} =$ 9.09

way to achieve $d_j^* = d_j$ is to reduce the allotted rate at node 1 as shown in the USC in Figure 4.8(b). This yields the total allotted resources at the nodes = 10 + 9.09= 19.09.

It is clear from the above example that the resource allocation for a session j to achieve a worst-case delay $= d_j^*$ in a network of GPS servers is not unique. A natural question that arises at this point is: How should the resources be allotted at the nodes in the session's path so that the worst case delay $d_j^* = d_j$ and the network resources (i.e. bandwidth) allocated is the least? The next Section addresses this question.

4.3 Optimal Bandwidth Calculations to Guarantee End-to-End Delay

To address the question of optimal bandwidth allocation in a general K node network, we first gain insight into an optimal bandwidth allocation for a session j in case of a two node network.

4.3.1 The Two-Node Case

In the following we formulate the problem of finding the optimal bandwidth for a session j as a Linear Program. We first show that the *constraint* region or the *schedulable* region (used interchangeably) $C = \{\bar{r}_j : d_j^* \leq d_j\}$ where $\bar{r}_j$ is the vector of allotted rates for a session j at the two nodes, is a convex region.

As mentioned before, we assume that the maximum rate that can be allotted at any node for session j is $\max(\frac{\sigma_j}{d_j}, \rho_j)$. Thus the two immediate constraints that follow are $r_{j,1} \leq \max(\frac{\sigma_j}{d_j}, \rho_j)$ and $r_{j,2} \leq \max(\frac{\sigma_j}{d_j}, \rho_j)$ where $r_{j,k}$ represents the session j rate at node k. The worst case delay is equal to d_j all along the boundary of the constraint region except along the above "maximum rate" constraints. In the interior of the constraint region the worst case delay $d_j^* < d_j$.

We now describe with the help of an example how the constraint region is obtained in the two node network case. The example considered is that of a session j with

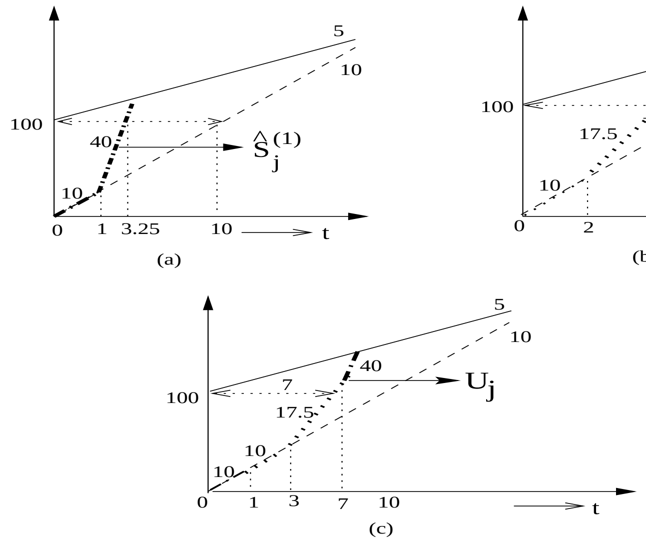

Figure 4.9: (a) Session j service curve at node 1 ($r_{j,1} = 10$). (b) Session j service curve at node 2 ($r_{j,2} = 10$). (c) Session j USC.

$(\sigma_j, \rho_j, d_j) = (100, 5, 10)$ passing through a network of two nodes. Figure 4.9 illustrates the session j service curves at nodes 1 and 2 and the corresponding USC when $r_{j,1} = r_{j,2} = \max(\frac{\sigma_j}{d_j}, \rho_j) = 10$. The worst case delay $d_j^* = 7$ and hence there is room for a reduction in the rates allotted. Figure 4.10 illustrates the constraint region obtained for session j such that its delay guarantee is satisfied. The first step in obtaining the constraint region is to find the minimum rates that have to be allotted at nodes 1 and 2 to satisfy the delay guarantee. The minimum rate that has to be allotted at any node k is found by allotting a rate of $\max(\frac{\sigma_j}{d_j}, \rho_j)$ at all the remaining nodes $K - \{k\}$ and computing the rate to be allotted at the node k such that the worst case delay d_j^* computed from the session j USC is equal to d_j. Let this minimum rate computed in the above manner for a session j at node k be denoted by $r_{j,k}^{\min}$. In the example considered the minimum rates that have to be allotted at nodes 1 and 2 are $r_{j,1}^{\min} = 2.76$ and $r_{j,2}^{\min} = 0.93$. These are represented by the points $A(2.76, 10)$ and $B(10, 0.93)$ in the constraint region in Figure 4.10. The corresponding session j USCs are represented in Figure 4.11. By Lemma 4.2 we know that the worst-case delay is always obtained either at d_σ or at one of the break-points in the USC. The number of break-points in the USC and the epoch when the worst-case delay occurs depends on $\bar{r}_j = (r_{j,1}, r_{j,2})$. The *structure* of the USC is

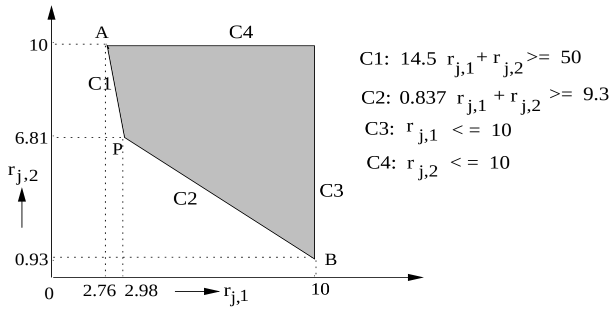

Figure 4.10: The schedulable region of session j.

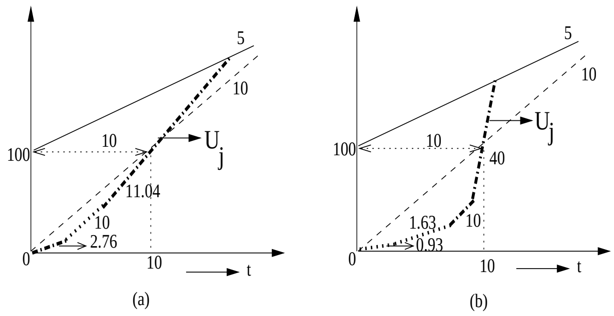

Figure 4.11: (a)Session j USC at point $A(r_{j,1}^{\min}, \frac{\sigma_j}{d_j})$ in the constraint region (b) Session j USC at point $B(\frac{\sigma_j}{d_j}, r_{j,2}^{\min})$ in the constraint region.

determined by the constituent service curve segments of the USC. Hereafter the structure of the USC $U_{j,\bar{r}_j}$ refers to the set of service curve segments that constitute the USC, $U_{j,\bar{r}_j}$. As $r_{j,1}$ and $r_{j,2}$ are varied the service curve segments constituting the USC also vary. Thus the structure of the USC varies as $r_{j,1}$ and $r_{j,2}$ are varied even while keeping the worst-case delay d_j^* the same. The constraints are obtained by starting with the session j USC at point $A(r_{j,1}^{\min}, \frac{\sigma_j}{d_j})$ and increasing $r_{j,1}$ while decreasing $r_{j,2}$ such that $d_j^* = d_j$. The varying structure of the USC as the rates vary, has to be taken into account.

The general form of a constraint which is determined easily from the USC is:

$$a\ r_{j,1} + b\ r_{j,2} \geq c \quad \text{for } r_{j,1} \in [k_1, k_2]$$

where k_1 and k_2 are some specific values of $r_{j,1}$. The structure of the USC, i.e. the number and the type of service curve segments constituting the USC remain the same for $r_{j,1} \in [k_1, k_2]$, beyond which the USC structure changes and a different constraint is obtained. The value of c depends on whether $\frac{\sigma_j}{d_j} \geq \rho_j$ or $\frac{\sigma_j}{d_j} < \rho_j$. In case $\frac{\sigma_j}{d_j} \geq \rho_j$, we know by Lemma 4.3 that the worst-case delay is experienced by the last bit of the initial burst of session j, hence the value of $c = \sigma_j$ for all intervals of $r_{j,1}$. Note that in all the examples the constraint equations are normalized by the value of b, i.e. the constant term in these equations is $\frac{c}{b}$. In the case when $\frac{\sigma_j}{d_j} < \rho_j$, we know by Lemma 4.4 that the worst case delay is not necessarily experienced by the last bit of the initial burst. Thus c could take values $\geq \sigma_j$ depending on which bit experiences the worst-case delay. The term a depends on the following three factors:

(i) The number of segments from the session j service curve at node 1 constituting the USC, till the epoch when d_j^* is achieved.

(ii) The relative slopes of the session j service curve segments from node 1 till the epoch when d_j^* is achieved.

(iii) The delay that the service curve segments from node 1 contribute to the end-to-end delay.

The term b similarly depends on the above three factors with respect to the session j service curve at node 2. Let $n_{j,k}$ denote the number of line segments from the session j service curve at node k, in the USC, till the epoch when d_j^* is achieved. Recall that the session j service curve at the node k under the all greedy regime is denoted by $\hat{S}_j^{(k)}$. It is specified by a list of pairs:

$$(s_1^k, d_1^k), (s_2^k, d_2^k), \ldots, (s_{m_k}^k, d_{m_k}^k)$$

where s_l^k is the slope of the l^{th} line segment, d_l^k is its duration and m_k is the number of line segments in the session j service curve at node k. The ratios $\frac{s_l^k}{s_1^k}$; $1 \leq l \leq m_k$ remain constant irrespective of the allotted rates. Then:

$$a = \sum_{l=1}^{n_{j,1}} (\frac{s_l^1}{s_1^1})(d_l^1) \text{ and } b = \sum_{l=1}^{n_{j,2}} (\frac{s_l^2}{s_1^2})(d_l^2) \tag{4.4}$$

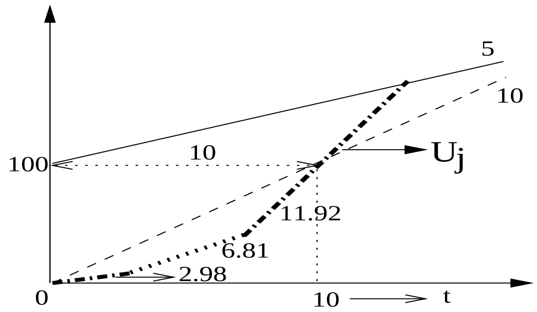

Figure 4.12: Session j USC at the optimal point P in the constraint region.

In the example considered, the constraints obtained from the USC are:

$$14.5\ r_{j,1} + r_{j,2} \geq 50; \quad r_{j,1} \in [2.76, 2.98]$$

$$.837\ r_{j,1} + r_{j,2} \geq 9.3; \quad r_{j,2} \in [2.98, 10]$$

The vertex $P(2.98, 6.81)$ in the constraint region is due to the change in the structure of the USC. The session j USC at P is illustrated in Figure 4.12.

The constraint region obtained in Figure 4.10 for the example that we considered is convex. The following Lemmas show that this is not a mere coincidence and that the constraint region of any session in the two-node network case is convex.

The general procedure for the construction of the constraint region is: Begin with the USC at the point $A(r_{j,1}^{\min}, \max(\frac{\sigma_j}{d_j}, \rho_j))$ and increase $r_{j,1}$ while reducing $r_{j,2}$ till the USC at the point $B(\max(\frac{\sigma_j}{d_j}, \rho_j), r_{j,2}^{\min})$ is reached. Then the general form of the constraint set is as follows:

$$C_i: \ a_i r_{j,1} + b_i r_{j,2} \geq c_i \text{ for } r_{j,1} \in [k_i, k_{i+1}] \text{ for } 1 \leq i \leq m \tag{4.5}$$

where m is the total number of constraints.

In the strict sense a session j USC depends on the exact rates allotted to the session at the nodes along its path. However the structure of the USC (i.e. the segments constituting the USC) remains unchanged as long as $r_{j,1}$ and $r_{j,2}$ vary satisfying a particular constraint C_i. In this sense each constraint C_i is associated with a particular structure of the USC. Henceforth we will refer to the session j USC structure associated with the constraint C_i

as U_{j,C_i}, i.e., U_{j,C_i} represents a set of USCs that satisfy the constraint C_i.

Lemma 4.6 *In the construction of the session j constraint region, as $r_{j,1}$ increases and $r_{j,2}$ decreases while keeping the end-to-end delay $= d_j$ the following hold:*
(i) The number of segments from the session j service curve at node 1 in the USC till the epoch of the maximum delay, i.e. $n_{j,1}$, is non-increasing.
(ii) The number of segments from the session j service curve at node 2 in the USC till the epoch of the maximum delay, i.e. $n_{j,2}$, is non-decreasing.

Proof: Case (a): $\frac{\sigma_j}{d_j} \geq \rho_j$

In this case the session j worst case delay is always experienced by the last bit of the initial burst at the epoch d_σ (Lemma 4.3). As $r_{j,1}$ increases, the segment slopes of the service curve at the node 1 increase and as $r_{j,2}$ decreases, the segment slopes of the service curve at the node 2 decrease. Since the USC is constructed by appending the segments from the service curves at the nodes 1 and 2 in increasing order of their slopes, (i) and (ii) follow.

Case (b): $\frac{\sigma_j}{d_j} < \rho_j$

In this case the epoch of the worst case delay need not be d_σ (Lemma 4.4). Suppose that the worst case delay occurs at the same epoch of time in every USC associated with the constraints as we move from one constraint to the other, then by the same argument as put forth in *Case (a)*, (i) and (ii) hold.

Suppose that the worst case delay does not occur at the same epoch in every USC associated with the constraints as we move from a constraint to the other. Then consider the following:
The set of USCs associated with the constraint C_i is denoted by U_{j,C_i}. As long as the rates vary satisfying the constraint C_i, the epoch at which the worst case delay is achieved remains the same in the associated set of USCs, U_{j,C_i}. Let this epoch in the USC set U_{j,C_i} be denoted by t_{C_i}. Let the set of USCs associated with the next constraint C_{i+1} be $U_{j,C_{i+1}}$ and the time epoch when the worst case delay occurs in this set of USCs be denoted by $t_{C_{i+1}}$.

Let s_t denote the slope of a line segment in the session j USC at the epoch t. Then by Lemma 4.2:
In the USC set U_{j,C_i}:

$$\begin{aligned} s_t &< \rho_j \quad \forall \quad t < t_{C_i} \text{ and} \\ s_t &\geq \rho_j \quad \forall \quad t \geq t_{C_i} \end{aligned} \tag{4.6}$$

In the USC set $U_{j,C_{i+1}}$:

$$\begin{aligned} s_t &< \rho_j \quad \forall \quad t < t_{C_{i+1}} \text{ and} \\ s_t &\geq \rho_j \quad \forall \quad t \geq t_{C_{i+1}} \end{aligned} \tag{4.7}$$

In the construction of the constraint region, the rate $r_{j,1}$ is increasing as we move from the USC set U_{j,C_i} to the USC set $U_{j,C_{i+1}}$. As $r_{j,1}$ increases the slopes of the service curve segments at the node 1 increase and hence by (4.6) and (4.7) any segment which does not appear before the worst case delay epoch, t_{C_i} in the USC set U_{j,C_i} will not appear before the worst case delay epoch $t_{C_{i+1}}$ in the USC set $U_{j,C_{i+1}}$. Thus as $r_{j,1}$ increases the number of segments from the session j service curve at node 1 in the USC till the epoch of the maximum delay, i.e. $n_{j,1}$, decreases.

The rate $r_{j,2}$ is decreasing as we move from the USC set U_{j,C_i} to the USC set $U_{j,C_{i+1}}$. As $r_{j,2}$ decreases the slopes of the service curve segments at the node 2 decrease and hence by (4.6) and (4.7) any segment which does not appear after the worst case delay epoch t_{C_i} in the USC set U_{j,C_i} will not appear after the worst case delay epoch $t_{C_{i+1}}$ in the USC set $U_{j,C_{i+1}}$. Thus as $r_{j,2}$ decreases the number of segments from the session j service curve at node 2, in the USC till the epoch of the maximum delay, i.e. $n_{j,2}$ increases.

Thus (i) and (ii) are true in both the cases.

$\diamondsuit$

Lemma 4.7 *In the constraint set in (4.5) as $r_{j,1}$ increases and $r_{j,2}$ decreases and we move* *from one constraint to the other, the following holds:*

$$a_1 > a_2 > \ldots > a_m \text{ and } b_1 < b_2 < \ldots < b_m$$

Proof: In the construction of the constraint region, as $r_{j,1}$ increases the following are true:
(i) By Lemma 4.6 the number of segments from the session j service curve at node 1 in the USC till the epoch of the maximum delay, i.e. $n_{j,1}$ decreases.
(ii) The delay that a service curve segment l, $(1 \leq l \leq n_{j,1})$ from node 1 contributes to the end-to-end delay in the USC, i.e. d_l^1, remains the same.
(iii) The relative slopes of the session j service curve segments from node 1, $\frac{s_l^1}{s_1^1}$ $(1 \leq l \leq n_{j,1})$, remain the same.

The coefficient of $r_{j,1}$ in the constraint equations is as given in (4.4) i.e.

$$a = \sum_{l=1}^{n_{j,1}} (\frac{s_l^1}{s_1^1})(d_l^1)$$

Thus as $r_{j,1}$ increases and we move from a constraint C_i to C_{i+1}, the factors (i), (ii) and (iii) work to reduce the term a and hence $a_i > a_{i+1}$.

In the construction of the constraint region as $r_{j,2}$ decreases the following are true:
(i) By Lemma 4.6 the number of segments from the session j service curve at node 2 in the USC till the epoch of the maximum delay, i.e. $n_{j,2}$ increases.
(ii) The delay that a service curve segment l, $(1 \leq l \leq n_{j,2})$ from node 2 contributes to the end-to-end delay in the USC, i.e. d_l^2 remains the same.
(iii) The relative slopes of the session j service curve segments from node 2, $\frac{s_l^2}{s_1^2}$ $(1 \leq l \leq n_{j,2})$ remain the same.

The coefficient of $r_{j,2}$ in the constraint equations is as given in (4.4) i.e.

$$b = \sum_{l=1}^{n_{j,2}} (\frac{s_l^2}{s_1^2})(d_l^2)$$

Thus as $r_{j,2}$ increases the factors (i), (ii) and (iii) work to reduce the term b and hence $b_i < b_{i+1}$.

$\diamondsuit$

Theorem 4.2 *The schedulable region for a session j in the two node network case can be expressed as* $\mathbf{Ar} \geq \mathbf{C}$ *where* $\mathbf{A}$ *is an $m * 2$ matrix (m is the total number of constraints),* $\mathbf{r} = \begin{pmatrix} r_{j,1} \\ r_{j,2} \end{pmatrix}$ *and* $\mathbf{C}$ *is an $m * 1$ matrix.*

Proof: The procedure for the construction of the constraint region yields the constraints as follows:

$$C_i : \; a_i r_{j,1} + b_i r_{j,2} \geq c_i; \quad r_{j,1} \in [k_i, k_{i+1}]$$

These can be written as:

$$C_i : \; r_{j,2} \geq \frac{c_i}{b_i} - \frac{a_i}{b_i} r_{j,1}; \quad r_{j,1} \in [k_i, k_{i+1}]$$

Now consider any two adjacent constraints C_i and C_{i+1} i.e.,

$$\begin{aligned} C_i \quad &: \quad r_{j,2} \geq \frac{c_i}{b_i} - \frac{a_i}{b_i} r_{j,1}; \; r_{j,1} \in [k_i, k_{i+1}] \\ C_{i+1} \quad &: \quad r_{j,2} \geq \frac{c_{i+1}}{b_{i+1}} - \frac{a_{i+1}}{b_{i+1}} r_{j,1}; \; r_{j,1} \in [k_{i+1}, k_{i+2}] \end{aligned}$$

Also consider a point $A(\alpha_1, \beta_1)$ such that $\alpha_1 \in [k_i, k_{i+1}]$ and satisfies the constraint C_i. Then the following conditions hold:

$$\beta_1 \geq \frac{c_i}{b_i} - \frac{a_i}{b_i} \alpha_1 \tag{4.8}$$

$$-\frac{a_i}{b_i} < -\frac{a_{i+1}}{b_{i+1}} \quad \text{(From Lemma 4.7)} \tag{4.9}$$

Since the constraints C_i and C_{i+1} are adjacent constraints the following is the continuity condition:

$$\frac{c_i}{b_i} - \frac{a_i}{b_i} k_{i+1} = \frac{c_{i+1}}{b_{i+1}} - \frac{a_{i+1}}{b_{i+1}} k_{i+1} \tag{4.10}$$

We now show that the point A also satisfies the inequality $r_{j,2} \geq \frac{c_{i+1}}{b_{i+1}} - \frac{a_{i+1}}{b_{i+1}} r_{j,1}$ in the constraint C_{i+1}, i.e., $\beta_1 \geq \frac{c_{i+1}}{b_{i+1}} - \frac{a_{i+1}}{b_{i+1}} \alpha_1$

Consider the term $-\frac{a_{i+1}}{b_{i+1}}\alpha_1 + \frac{c_{i+1}}{b_{i+1}}$. It can be shown after some algebra from (4.9) and (4.10) that:

$$-\frac{a_{i+1}}{b_{i+1}}\alpha_1 + \frac{c_{i+1}}{b_{i+1}} \leq -\frac{a_i}{b_i}\alpha_1 + \frac{c_i}{b_i} \tag{4.11}$$

From (4.8) and (4.11) we get:

$$\beta_1 \geq -\frac{a_{i+1}}{b_{i+1}}\alpha_1 + \frac{c_{i+1}}{b_{i+1}}$$

i.e. the point $A(\alpha_1, \beta_1)$ satisfies $r_{j,2} \geq -\frac{a_{i+1}}{b_{i+1}}r_{j,1} + \frac{c_{i+1}}{b_{i+1}}$. Similarly it can be shown that any point $B(\alpha_2, \beta_2)$ satisfying C_{i+1} also satisfies $C_i : r_{j,2} \geq -\frac{a_i}{b_i}r_{j,1} + \frac{c_i}{b_i}$

Thus, any point satisfying C_i or C_{i+1} also satisfies the inequality in the other constraint (C_{i+1} or C_i respectively). Thus every pair of adjacent constraints C_i and C_{i+1} can be written as:

$$\begin{aligned} C_i \quad &: r_{j,2} \geq \frac{c_i}{b_i} - \frac{a_i}{b_i}r_{j,1}; \\ C_{i+1} \quad &: \ r_{j,2} \geq \frac{c_{i+1}}{b_{i+1}} - \frac{a_{i+1}}{b_{i+1}}r_{j,1}; \end{aligned}$$

with $r_{j,1} \in [k_i, k_{i+2}]$, which represents a convex region. Continuing thus for every pair of adjacent constraints, it can be concluded that a point that satisfies any one of the constraints C_i must also satisfy the inequality in all the other constraints. The schedulable region can be simply expressed as:

$$C_i : r_{j,2} \geq \frac{c_i}{b_i} - \frac{a_i}{b_i}r_{j,1}; \ 1 \leq i \leq m$$

where the "maximum-rate" constraints are also accommodated into the above set of constraints with an appropriate change of sign. Thus the schedulable region can be expressed as $\mathbf{Ar} \geq \mathbf{C}$ where $\mathbf{A} = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_m & b_m \end{pmatrix}$, $\mathbf{C} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}$, $\mathbf{r} = \begin{pmatrix} r_{j,1} \\ r_{j,2} \end{pmatrix}$ where a_i and b_i for the constraint C_i are obtained from U_{j,C_i} according to the equation (4.4).

$\diamondsuit$

We now turn to the question that was posed earlier: What is the optimal bandwidth allocation for a session j in the two node network case? From Theorem 4.2 it is clear that this problem can be expressed as follows:

$$\text{minimize} \quad (r_{j,1} + r_{j,2})$$

$$\text{subject to} \quad \mathbf{Ar} \geq \mathbf{C}$$

For the example considered earlier, the LP is:

$$\text{minimize} \quad (r_{j,1} + r_{j,2})$$

$$\text{subject to} \quad 14.5 r_{j,1} + r_{j,2} \geq 50$$

$$0.837\ r_{j,1} + r_{j,2} \geq 9.3$$

$$r_{j,1} \leq 10$$

$$r_{j,2} \leq 10$$

From the constraint region illustrated in Figure 4.10, the optimal bandwidth allocation is obtained at the vertex P (where $r_{j,1} = 2.98$ and $r_{j,2} = 6.81$). The corresponding session j USC is illustrated in Figure 4.12.

In the above example the worst case delay is always obtained at the epoch d_σ in the USC. In case $\frac{\sigma_j}{d_j} < \rho_j$, the worst case delay could occur at different break-points. By Theorem 4.2 the constraint region is convex and this is illustrated in the following example in which $\frac{\sigma_j}{d_j} < \rho_j$*. Session j with $(\sigma_j, \rho_j, d_j) = (100, 10, 20)$ passes through nodes 1 and 2. The constraint region and the USCs at the points A and B is illustrated in Figure 4.13. The optimal bandwidth allocation is $r_{j,1} = 6.88$ and $r_{j,2} = 4.97$, as can be seen from the constraint region.

*In this example the choice of B did not matter due to the specific nature of the session j service curves.

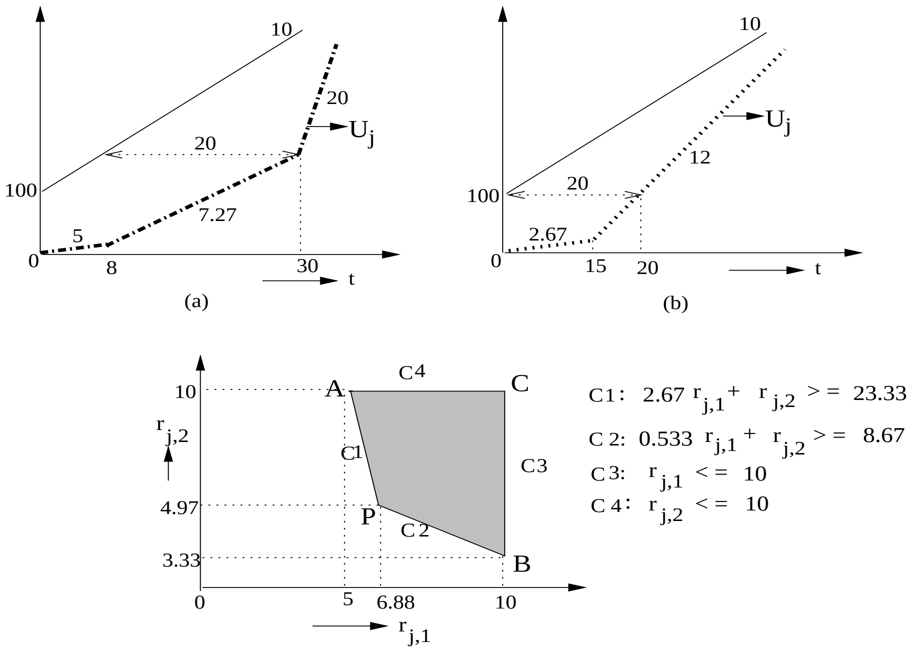

Figure 4.13: (a)Session j USC at point A (b) Session j USC at point B. (c) The constraint region.

4.3.2 Optimal Bandwidth Allocation in Nonacyclic Networks

As mentioned in the last Section, we assumed an acyclic network till this point. We now investigate the optimal resource allocation in general non acyclic networks for a two node case. In non acyclic networks an arbitrary $\{\phi\}$ assignment can drive the system into instability and hence it is important to avoid the assignments of the $\{\phi\}$ for which reasonable delay guarantees cannot be made. For these networks it is shown in [24] that for GPS assignments in which each session is treated "consistently" well relative to the other sessions, the network is stable and tight bounds on delay can be derived. This leads to an extra condition in these networks called the *Consistent Relative Session Treatment* [24] which is defined below.

Definition 1: Session j is said to *impede* a session i at the node k if

$$\frac{r_{i,k}}{r_{j,k}} < \frac{\rho_i}{\rho_j}$$

For any two sessions, i and j, that contend for a node k, either session i impedes session j or vice versa, unless $\frac{r_{i,k}}{r_{j,k}} = \frac{\rho_i}{\rho_j}$ in which case neither session impedes the other.

Definition 2: 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 optimal nodal allocation for a two node network is found using the *2NodeSched*() algorithm. In the case of acyclic networks it does not matter if this optimal solution is not a CRST solution too, since acyclic networks are stable and internal traffic characterization is possible even under a non-CRST allocation. This is not the case however with nonacyclic networks. We now describe how to obtain an optimal CRST solution from the optimal non-CRST solution that is obtained from the *2NodeSched*() procedure. We are interested in finding an optimal CRST solution for a session j with parameters (σ_j, ρ_j, d_j) and passing through a network of two nodes. The sessions which are already established in the network conform to the CRST allocation.

Let S denote the set of common sessions at the nodes 1 and 2. Let $A_j^{(k)} \subset S$ denote the set of sessions at node k that impede the session j at node k, while $B_j^{(k)}$ denotes the set of sessions at node k that do not impede session j at the node k. A session i at the node k for which the following holds:

$$\frac{r_{i,k}}{r_{j,k}} = \frac{\rho_i}{\rho_j}$$

is neither included in the set $A_j^{(k)}$ nor $B_j^{(k)}$, since in this case neither session impedes the other. In case of a non-CRST allocation, the sets $A_j^{(1)}(B_j^{(1)})$ and $A_j^{(2)}(B_j^{(2)})$ are not equal. However, it is always true (in case of a non-CRST solution) that either $A_j^{(2)} \subset A_j^{(1)}$ or $A_j^{(2)} \supset A_j^{(1)}$. This is because the sessions $\in S$ conform to the CRST allocation themselves. This is illustrated in Figure 4.14 where session j is a fresh incoming session to a two-node network. There are four sessions that share the same path as that of session j, i.e., pass through the nodes 1 and 2. Their rate allocations conform to CRST. Suppose that

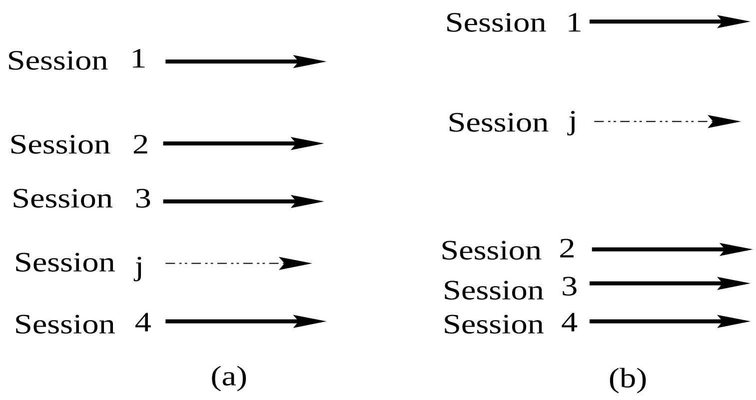

Figure 4.14: The positions of the sessions at each node are indicative of the relative values of their (allotted rate \ ρ) (a) Sessions 1 and 2 impede j, $\frac{r_{1,1}}{\rho_1} > \frac{r_{2,1}}{\rho_2} > \frac{r_{j,1}}{\rho_j} > \frac{r_{3,1}}{\rho_3} > \frac{r_{4,1}}{\rho_4}$ (b) Session 1 impedes session j, $\frac{r_{1,2}}{\rho_1} > \frac{r_{j,2}}{\rho_j} > \frac{r_{2,2}}{\rho_2} > \frac{r_{3,2}}{\rho_3} > \frac{r_{4,2}}{\rho_4}$. In this case $A_j^{(1)} \supset A_j^{(2)}$

$\forall\ m\ <\ n$ $(1 \leq m, n \leq 4)$, session m impedes session n at node 1, then the following holds:

$$\frac{r_{1,1}}{\rho_1} > \frac{r_{2,1}}{\rho_2} > \frac{r_{3,1}}{\rho_3} > \frac{r_{4,1}}{\rho_4}$$

Since the sessions' rate allocations conform to the CRST, the following must hold at node 2:

$$\frac{r_{1,2}}{\rho_1} > \frac{r_{2,2}}{\rho_2} > \frac{r_{3,2}}{\rho_3} > \frac{r_{4,2}}{\rho_4}$$

From such an ordering of sessions it is clear that the set of sessions that impede session j at node 2, $A_j^{(2)}$ has to be a subset or a superset of those that impede session j at node 1, $A_j^{(1)}$ (in case of a non-CRST solution).

Now, suppose that for a non-CRST optimal solution $A_j^{(2)} \supset A_j^{(1)}$ and $B_j^{(2)} \subset B_j^{(1)}$, i.e., a larger set of sessions that $\in S$ impede the session j at node 2 than at node 1. In order to approach the optimal CRST solution, the session j rate at node 2, $r_{j,2}$ is increased in increments while the rate at node 1, $r_{j,1}$ is decreased by an appropriate amount such that the session j end-to-end delay $= d_j$. Geometrically, this would amount to moving along a constraint in the schedulable region to approach a CRST solution, as illustrated in the Figure 4.15. In this process the set of sessions that impede session j at the node 2, $A_j^{(2)}$ becomes smaller while the set of sessions that impede session j at the node 1, i.e. $A_j^{(1)}$ becomes larger. Similarly, the set $B_j^{(2)}$ is increasing while $B_j^{(1)}$ is decreasing. The

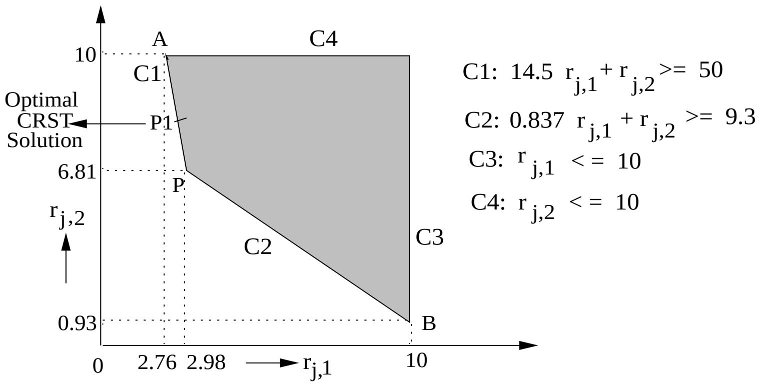

Figure 4.15: P is the optimal non-CRST solution while $P1$ is the optimal CRST solution.

optimal CRST solution is obtained when $A_j^{(1)} = A_j^{(2)}$ and $B_j^{(1)} = B_j^{(2)}$. A session i for which $\frac{r_{i,k}}{r_{j,k}} = \frac{\rho_i}{\rho_j}$ holds, may be included in either of the sets to achieve equality.

The procedure stated above is put in the form of the algorithm below:

OPTIMAL-CRST()

Step 1:

$k_1 = \{\text{node } k : A_j^{(k_1)} \subset A_j^{(k_2)}\}$

$k_2 = \{\text{node } k : A_j^{(k_2)} \supset A_j^{(k_1)}\}$

$(r^*_{j,k_1}, r^*_{j,k_2})$ = Optimal non-CRST allocation at the nodes k_1 and k_2.

Step 2:

$r_{j,k_2} = r_{j,k_2} + \epsilon$

* Decrement r_{j,k_1} by δ such that $d_j = d_j^*$ *

$r_{j,k_1} = (r_{j,k_1} - \delta)_{d_j = d_j^*}$

Step 3:

If $(A_j^{(k_1)} = A_j^{(k_2)})$

{

Optimal CRST solution = (r_{j,k_1}, r_{j,k_2})

STOP

}

else

GOTO *Step 2*

The above Procedure increments r^*_{j,k_2} and decrements r^*_{j,k_1}. In this process the set $A_j^{(k_2)}$ becomes smaller while the set $A_j^{(k_1)}$ becomes larger. Before starting the Procedure $A_j^{(k_2)} \supset A_j^{(k_1)}$. If the Procedure does not find that $A_j^{(k_2)} = A_j^{(k_1)}$ at any point then it finally terminates at the point $r_{j,k_1} = r_{j,k_1}^{\min}$ and $r_{j,k_2} = \max(\frac{\sigma_j}{d_j}, \rho_j)$. At this point we would still have $A_j^{(k_2)} \supset A_j^{(k_1)}$. In such a case the Procedure does not terminate at a CRST solution.

Remark: The above Procedure can find the CRST solution if it exists on the boundary of the constraint region. Since we did not incorporate CRST as a constraint in the problem for the optimal solution, we do not necessarily terminate at a CRST solution in all cases. Further work needs to be done in this direction to incorporate the CRST constraints and express the optimal bandwidth allocation problem as a Linear Program.

4.3.3 The K-Node Case

The procedure to construct the schedulable region and hence obtain the constraints in the case of a two node network is described in the previous Section. Let this procedure be called as *2NodeSched*(). In this Section we use the procedure recursively to obtain the schedulable region for a K-node ($K > 2$) network. This procedure which obtains the schedulable region and hence the constraints in the K node case is called the *NodeSched*(K). Before executing the procedure, the minimum rate that is required at any node k in the network $(1 \leq k \leq K)$ such that the session j delay guarantee is not violated, i.e., $r_{j,k}^{min}$ is obtained. $r_{j,k}^{min}$ is obtained from the session j USC by allotting the rates at the nodes $K - \{k\}$ equal to $\max(\frac{\sigma_j}{d_j}, \rho_j)$. The procedure is as follows:

NodeSched(K):

{

* Initialization *

Rates at nodes $2, \ldots, K = \max(\frac{\sigma_j}{d_j}, \rho_j)$

Rate at node $1 = r_{j,1}^{min}$

```
        if (K = 2)
            2NodeSched();
        else
            while (r_{j,K} > r^{min}_{j,K})
                {
                    NodeSched(K-1);
                    r_{j,K} = r_{j,K} - ε;
                }
}
```

The above procedure computes the schedulable region of any K node ($K \geq 3$) network recursively. The procedure fixes the rate at the K^{th} node and then obtains the schedulable region of the remaining subnetwork of $(K-1)$ nodes. It then decrements the rate of the K^{th} node by ϵ and repeats the procedure. This is continued till the rate at the K^{th} node $= r^{\min}_{j,K}$. For every execution of the Procedure *NodeSched(K)* for a K-node subnetwork, the schedulable region and hence the constraints obtained are K-dimensional.

It is proved in Theorem 4.2 that the schedulable region in the case of a two node network is a convex region. In the following Theorem we prove that the schedulable region obtained by the procedure *NodeSched(K)* for a K-node network is convex.

Theorem 4.3 *The schedulable region for a K-node network of GPS schedulers is a convex region.*

Proof: The schedulable region for any K-node network is obtained from the *NodeSched(K)* procedure. The schedulable region can be expressed by the K-dimensional constraints as follows:

$$\begin{aligned} C_i \quad &: \quad a_{i,1}r_{j,1} + a_{i,2}r_{j,2} + \cdots + a_{i,K}r_{j,K} \geq c_i; \ 1 \leq i \leq m \\ \text{for} \quad &r_{j,1} \in [\alpha_{i,1}, \beta_{i,1}], r_{j,2} \in [\alpha_{i,2}, \beta_{i,2}], \ldots, r_{j,K} \in [\alpha_{i,K}, \beta_{i,K}] \end{aligned} \tag{4.12}$$

where $a_{i,1}, a_{i,2}, \ldots, a_{i,K}$ are the coefficients of $r_{j,1}, r_{j,2}, \ldots, r_{j,K}$ respectively in the constraint C_i. The constraints in (4.12) follow from the USCs in the $K - node$ case (The

USC is the fundamental construction. The Procedure is a way to explore the "rate space" systematically). As noted in Section 4.3.1 the coefficient of $r_{j,c}$ in the i^{th} constraint, i.e. $a_{i,c}$, depends on:

(i) The number of line segments from the session j service curve at node c in the set of USCs associated with the constraint C_i, U_{j,C_i}, till the epoch when d_j^* is achieved.

(ii) The relative slopes of the session j service curve segments from node c.

(iii) The delay that the service curve segments from node c contribute to the end-to-end delay.

Any two constraints in K-dimensions from the set in (4.12) are said to be *adjacent* if they have a $(K-1)$-dimensional hyperplane as the region of intersection, *inside* the constraint region. Suppose C_i and C_{i+1} are a pair of *adjacent* constraints obtained as $r_{j,c}$ (for some node c) decreases. Let the set of USCs associated with the constraint C_i be denoted by U_{j,C_i} and that associated with the constraint C_{i+1} be denoted by $U_{j,C_{i+1}}$. The constraint C_{i+1} results due to a structure change in the USC set U_{j,C_i} (resulting in the USC set $U_{j,C_{i+1}}$), as $r_{j,c}$ decreases. Note that the c here could be different in general for different pairs of *adjacent* constraints.

Now consider the two adjacent constraints C_i and C_{i+1} which result when $r_{j,c}$ decreases (for some c). Then C_i and C_{i+1} can be written as:

$$C_i : r_{j,c} \geq \frac{c_i}{a_{i,c}} - \frac{a_{i,1}}{a_{i,c}} r_{j,1} - \frac{a_{i,2}}{a_{i,c}} r_{j,2} - \cdots - \frac{a_{i,K}}{a_{i,c}} r_{j,K};\; r_{j,k} \in [\alpha_{i,k}, \beta_{i,k}]$$

$$C_{i+1} : r_{j,c} \geq \frac{c_{i+1}}{a_{i+1,c}} - \frac{a_{i+1,1}}{a_{i+1,c}} r_{j,1} - \frac{a_{i+1,2}}{a_{i+1,c}} r_{j,2} - \cdots - \frac{a_{i+1,K}}{a_{i+1,c}} r_{j,K};\; r_{j,k} \in [\alpha_{i+1,k}, \beta_{i+1,k}]$$

where $1 \leq k \leq K$ and $k \neq c$

As $r_{j,c}$ decreases the minimum allowable values of $r_{j,k}; 1 \leq k \leq K, k \neq c$ such that the delay guarantee is not violated can only increase. By Lemma 4.7:

$$a_{i,k} > a_{i+1,k};\quad 1 \leq k \leq K,\quad k \neq c \;\text{ and }\; a_{i,c} < a_{i+1,c}$$

$$\Rightarrow -\frac{a_{i,k}}{a_{i,c}} < -\frac{a_{i+1,k}}{a_{i+1,c}};\quad 1 \leq k \leq K,\quad k \neq c \tag{4.13}$$

As was shown in Theorem 4.2, it can be shown after some algebra using (4.13) that any point satisfying C_i or C_{i+1} also satisfies the inequality in the other constraint (C_{i+1} or C_i respectively). Continuing thus for every pair of adjacent constraints, it can be concluded that a point that satisfies any one of the constraints C_i must also satisfy the inequality in all the other constraints. The schedulable region can thus be expressed simply as follows:

$$C_i: \; a_{i,1}r_{j,1} + a_{i,2}r_{j,2} + \cdots + a_{i,K}r_{j,K} \geq c_i; \; 1 \leq i \leq m$$

The "maximum rate" constraints are also accommodated into the above constraints with an appropriate change of sign. The above set of linear constraints enclose a convex region.

$\diamondsuit$

Thus for the K-node network the optimal bandwidth allocation problem can be expressed as:

$$\begin{aligned} \text{Minimize:} \quad & (r_{j,1} + r_{j,2} + \cdots + r_{j,K}) \\ \text{subject to} \quad & \mathbf{Ar} \geq \mathbf{C} \end{aligned}$$

where $\mathbf{A}$ is an $m * K$ matrix (m is the total number of constraints), $\mathbf{r}$ is the vector of rates at the nodes in the network and $\mathbf{C}$ is an $m * 1$ matrix. The optimal rate allocation can be found by standard LP techniques.

4.3.4 Optimal Bandwidth Allocation for a set of Sessions

In the last Section we obtained the optimal bandwidth allocation for a session j passing through a network of K GPS schedulers, *given* the rate allocations for the already established sessions in the network. In this Section we deal with a more general scenario where we are given a set of J sessions with the parameters (σ_j, ρ_j, d_j) $1 \leq j \leq J$, and their source-destination pairs. The problem now is to find an optimal allocation for every session j $(1 \leq j \leq J)$ at each of the nodes in the network. In the following we give the sketch of the algorithm, *CACNetwork* to compute the optimal rates of the sessions

$1 \leq j \leq J$ and the minimum total rate needed at every node k ($1 \leq k \leq K$) in the network to support the given set of sessions optimally. More study needs to be done for a detailed algorithm.

CACNetwork:

Step 1: * *Initialization* *

$k = 0$ * Indexes the node in the network *

$i = 0$ * Iteration Variable *

m_j * Number of hops in session j's path *

C_k = Link capacity serving node k * Sum of allotted session rates at node k *

N_k * Set of sessions at the node k *

$H_k = \emptyset$ * Set of sessions at the node k whose rates are fixed *

Step 2:

$k = k + 1$

Step 3:

$i = i + 1$

Keep the rates of the sessions at the nodes $l = 1, \ldots, (k-1)$ constant
and find the optimal allocation for every session j passing
through the node k and $\notin H_k$, i.e, $j \in N_k \backslash H_k$ using the *NodeSched(m_j)* procedure

Step 4:

Calculate the Backlog Clearing Times of the sessions $\notin H_k$ at the node k

Step 5:

If ($N_k = \emptyset$)

  $L(i) = 0$

else

  {

    Let $L(i)$ = session with the minimum backlog clearing time at node k

    $N_k = N_k - \{L(i)\}$

$H_k = H_k \cup \{L(i)\}$

}

Step 6:

If $L(i) \neq 0$

Goto *Step 3*

Step 7:

If $|C_k - \sum_{j \in N_k} r_{j,k}| > \epsilon$

{

$C_k = \sum_{j \in N_k} r_{j,k}$

$H_k = \emptyset$ and $i = 0$

Goto *Step 3* and repeat the process for the new C_k

}

Step 8:

If $k \neq K$

{

i = 0

Goto *Step 2*

}

The algorithm essentially consists of three nested loops. The innermost loop is executed for a particular node k. In this loop the optimal rates of the sessions are computed using the procedure *NodeSched(K)*. Their backlog clearing times are also computed. For every execution of the innermost loop, the rate of a session which has the minimum backlog clearing time is frozen and the session is included in the set H_k. In the next execution of the innermost loop, the optimal rates of the remaining sessions are computed using the $NodeSched(K)$, taking into account the backlog finish time of the most recent session included into the set H_k.

Once the rates of all the sessions at the node k are frozen, the middle loop checks if the sum of the allotted rates of all the sessions is equal to the server capacity that the loop had started with, i.e., C_k. At the end of the first iteration, if the sum of session rates $\sum_{j \in N_k} r_{j,k} < C_k$, then the remaining bandwidth $(C_k - \sum_{j \in N_k} r_{j,k})$ is assumed to be given to a "dummy" session, with parameters: $\sigma_d = \infty, \rho_d = 0$. This essentially means that the "dummy" session also gets a share of the excess bandwidth released when any session j $(j \in N_k)$ completes its backlog period. However, our aim is to find the minimum rates of sessions $1, \ldots, N_k$ at the node k without any assumption of the "dummy" session. In other words, we have to find the minimum capacity C_k that is required to support the N_k sessions, when the excess released bandwidth in any iteration i of sessions $L(1), \ldots, L(i-1)$ is distributed among the remaining sessions, $N_k - \{\cup_{l=1}^{i-1} L(l)\}$ only.

At the end of a few executions of the middle loop, the algorithm converges to a certain minimum capacity C_k at the node k that is needed to support the connections.

The outermost loop of the algorithm increments the node number k and computes the minimum capacity required at the next node in the network to support the connections passing through it. Note that while executing the algorithm for the k^{th} node in the network, the rates of the sessions at the nodes $l = 1, \ldots, (k-1)$ are fixed and hence the traffic characterization into the node k is known (There is no need to assume an upper bound B on the input burstiness). For the other nodes downstream of the node k the traffic characterization (B, ρ_j), is used.

Also note that in the algorithm the rates of the sessions are fixed node by node (i.e. the rates of all the sessions at a particular node k are fixed before moving on to the next node $(k+1)$). An alternative to this approach would be to fix the rates session by session (i.e. fix the rate of a session j at all the nodes in its path in one iteration before beginning the next iteration). Such a session j (whose rates at all the nodes in its path are fixed) would be chosen such that its network backlog finish time computed from its USC is the least among all the sessions' network backlog finish times. The two drawbacks of this approach are:

(i) The network backlog finish times of different sessions cannot be compared since they are computed under different staggered greedy regimes.

(ii) A session which has the minimum network backlog finish time does not necessarily

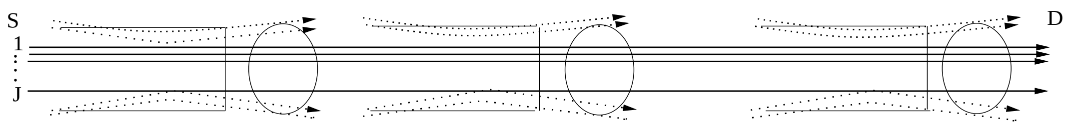

Figure 4.16: J Sessions from the Source S to Destination D. The cross traffic is shown in dotted.

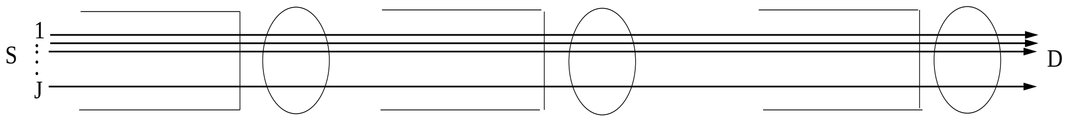

Figure 4.17: J Sessions from the Source S to Destination D and no cross traffic.

have the minimum backlog finish at each of the nodes.

We now consider a special case of the scenario considered above. This is illustrated in Figure 4.16. Some sessions are already established in the network. It is of interest to find the optimal allocation for the sessions $1, \ldots, J$. There are two ways of doing this:

(i) Use the algorithm *CACNetwork* and allot bandwidths to each of the J sessions at the K nodes.

(ii) Use the algorithm for the single node case (*CACSingleNode*) described in [18] to obtain the bandwidth at the node in the network with the maximum server capacity. Allot this rate to the sessions at all the other nodes too in the network.

In this scenario the CAC which yields a lesser bandwidth allocation (the *CACNetwork* or the *CACSingleNode*) depends on the nature and the amount of cross-traffic present. As was mentioned in 4.1 and illustrated in Figure 4.1 the USC could yield loose end-to-end delay bounds due to the independent sessions relaxation. Thus the *CACNetwork* which is based on the USC could result in a conservative resource allocation. This becomes more prominent as the number of sessions which share more than one common node along their path increase. On the other hand, the delay bounds could be tight when every session's

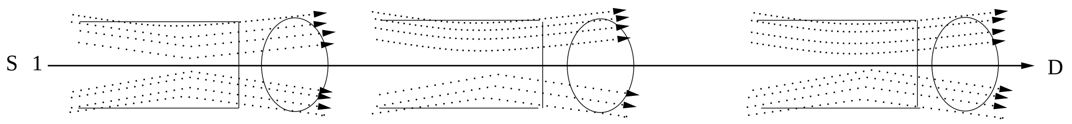

Figure 4.18: 1 Session from S to D and the rest is cross traffic.

path overlaps with one another for atmost one node.

In such cases where it is not clear which of the CACs result in a better allocation, we take the optimal allocation as:

$$\min(\text{Rate allotted by } \mathit{CACNetwork}, \text{ Rate allotted by } \mathit{CACSingleNode})$$

In some cases the CAC which yields a better resource allocation is very clear as illustrated in Figures 4.17 and 4.18. In Figure 4.17, there is no cross traffic. The optimal allocation for sessions $1, \ldots, J$ is obtained from *CACSingleNode*. Note that the independent sessions assumption fails completely here. In Figure 4.18 the *CACNetwork* does a better allocation. In this case the *CACSingleNode* would allot a bandwidth of $\max(\frac{\sigma_j}{d_j}, \rho_j)$.

4.4 Summary

In this Chapter we have presented an Optimal Call Admission Control algorithm for Generalized Processor Sharing schedulars. The CACs presented in the literature allot bandwidth according to the *Rate Proportional* resource allocation and are based on loose delay bounds. These CACs, though simple, would result in a gross over-allocation of bandwidth. The Optimal CAC in this Chapter does a *General Resource* allocation or a *Non-Rate Proportional Resource* allocation and overcomes the above limitations. We showed that:

- There are a number of allocations of the nodal bandwidths such that the worst case end-to-end delay is equal to the required delay guarantee.
- The problem of optimal Non Rate Proportional nodal bandwidth allocations for a network of GPS schedulers can be formulated as a Linear Program.

Chapter 5

Conclusions

Provision of Quality-of-Service (QoS) guarantees is an important issue in the design of Integrated Services packet networks. Call Admission Control is an integral part of the problem. Clearly without Call Admission Control, providing QoS guarantees will be impossible. In this context, a key issue is how to provide the nodal resources in order to meet the end-to-end QoS requirements of each connection. We investigate the problem of *optimal* nodal bandwidth allocation to satisfy the end-to-end delay requirements of Leaky-bucket shaped connections.

In this Chapter we summarize the contributions of this Thesis and then present some directions for future research.

5.1 Summary of Contributions

The optimal nodal resource allocation depends greatly on the schedulers used at the network nodes. We first considered a network of Rate-Based schedulers in which the allotted rate for a session at every node is atleast equal to the session's average rate. We showed:

- The characterization of the departure process of fluid Rate Proportional Servers.
- A better resource allocation (less conservative) can be obtained by taking advantage of the properties of the schedulers and the inter-hop dependencies of traffic streams than by

considering upper bounds to these traffic streams.

• The optimal bandwidth allocation at each node that satisfies an end-to-end delay requirement in a network of *Rate Proportional Servers.*

Rate Proportional resource allocation (i.e. rate allotted to a session is atleast equal to its long term average rate) could result in a gross over-allocation of network resources. This limitation can be overcome by considering a *Non-Rate Proportional* resource allocation. We presented:

• An Optimal Call Admission Control algorithm for Generalized Processor Sharing schedulers that does a *General Resource* allocation or a *Non-Rate Proportional Resource* allocation in the single node case. Its fundamental merit arises from the fact that it takes into account the bandwidth released by the sessions which have completed their backlogs, while calculating the sessions rates. The algorithm allots minimum rates to the sessions in order that their delay bounds are not violated. Numerical Results show that the Optimal Call Admission Control can accept significantly more connections than the CAC based on the *Rate Proportional* allocation, and hence results in a better network utilization.

• An Optimal Call Admission Control Algorithm in Shaped Generalized Processor Sharing schedulers (i.e. an appropriate shaper is introduced before the GPS scheduler). This results in a lesser resource allocation than that required in a scheduler (GPS) without a shaper.

Finally, we addressed the problem of reserving bandwidth *optimally* in a *network* of GPS schedulers to provide deterministic end-to-end delay guarantees. We showed that:

• There are a number of allocations of the nodal bandwidths such that the worst case end-to-end delay is equal to the required delay guarantee.

• The problem of optimal Non Rate Proportional nodal bandwidth allocations for a network of GPS schedulers can be formulated as a Linear Program.

5.2 Directions for Future Research

There are several directions in which future research on *optimal nodal bandwidth allocation* can be pursued. Some of these are:

• *Efficient CAC Implementation*: The algorithms proposed in this Thesis must be implemented efficiently to enable routers or switches allot bandwidth at the nodes in a real-time scenario. The computational complexity of the algorithms must be reduced. There will be a trade off between the computational complexity and the optimality of the final nodal allocations.

• *Statistical Guarantees*: We have addressed the problem of optimal nodal bandwidth allocation for providing deterministic end-to-end delay guarantees. However, most real-time applications are adaptive to an extent and can tolerate certain levels of packet loss and end-to-end delay bound violations. Hence, to enable a network to achieve efficient utilization of resources, it is desirable to provide statistical guarantees. Allotting the nodal resources optimally such that the end-to-end *statistical* guarantees are satisfied under different schedulers is a problem open for future research.

• *Non Rate Proportional or General Resource Allocation*: General or Non Rate Proportional resource allocation results in a more efficient use of the network resources than a Rate Proportional Allocation. It is of interest to investigate the problem of General Resource Allocation in scheduling algorithms other than GPS such as the Virtual Clock.

• *Optimal Delay Split in NPEDF*: Scheduling disciplines like NPEDF which are easy to implement are becoming increasingly popular in a Differentiated Services environment. It is of interest to address the problem of *Optimal Delay Split* in Non Preemptive Earliest Deadline First.

Appendix A

Proof of Theorem 2.4: Let $(1-\frac{L_j}{\sigma_j}) = a$. The Optimization problem can also be written as follows:

$$\text{Minimize:} \qquad -\ (\frac{\sigma_j}{d_1} + \frac{\sigma_j}{d_1(1-\frac{L_j}{\sigma_j}) + d_2} + \cdots +$$
$$\frac{\sigma_j}{d_1(1-\frac{L_j}{\sigma_j})^{n-1} + d_2(1-\frac{L_j}{\sigma_j})^{n-2} + \cdots + d_{n-1}(1-\frac{L_j}{\sigma_j}) + d_n}) \tag{A.1}$$

$$\text{subject to} \qquad D - d_1 - d_2 - \ldots - d_n \geq 0 \tag{A.2}$$

$$d_1 \geq 0 \tag{A.3}$$

$$d_2 + (a-1)d_1 \geq 0 \tag{A.4}$$

$$\vdots$$

$$d_1 a^{n-2}(a-1) + d_2 a^{n-3}(a-1) + \cdots + d_{n-1}(a-1) + d_n \geq 0 \tag{A.5}$$

where the set of constraints in (A.4), (A.5) are due to the constraints $r_j^{(1)} \geq r_j^{(2)} \geq \cdots \geq r_j^{(n)}$.

In order to obtain the optimal solution we use the Theorem of *Kuhn Tucker* which is stated below:

Theorem of Kuhn and Tucker: Let f be a concave function mapping U into $\mathcal{R}$ where $U \subset \mathcal{R}^n$ is open and convex. For $i = 1, 2, \ldots, l$, let $h_i : U -> \mathcal{R}$ also be concave functions.

Suppose there is some $\bar{x} \in U$ such that:

$$h_i(\bar{x}) > 0, \quad i = 1, \ldots, l$$

Then x^* maximizes f over:

$$\mathcal{D} = \{x \in U | h_i(x) \geq 0, \quad i = 1, \ldots, l\}$$

if and only if there is $\lambda^* \in \mathcal{R}^l$ such that the Kuhn-Tucker first-order conditions hold:

$$[KT - I] \quad Df(x^*) + \sum_{i=1}^{l} \lambda_i^* Dh_i(x^*) = 0$$

$$[KT - II] \quad \lambda^* \geq 0, \quad \sum_{i=1}^{l} \lambda_i^* h_i(x^*) = 0$$

The objective function in (A.1) and the constraints in (A.2) ... (A.5) satisfy the conditions in the *Kuhn Tucker Theorem.* We find that if:

$$(d_1^*, d_2^*, \ldots, d_n^*) = (\frac{\sigma_j D}{\sigma_j + (n-1)L_j}, \frac{L_j D}{\sigma_j + (n-1)L_j}, \ldots, \frac{L_j D}{\sigma_j + (n-1)L_j})$$

then $\exists (\lambda_1^*, \ldots, \lambda_{n+1}^*) \in \mathcal{R}^{n+1}$ such that the *Kuhn Tucker (KT)* first-order conditions hold. The $(\lambda_1^*, \ldots, \lambda_{n+1}^*)$ which satisfy the first-order KT equations are:

$$\begin{aligned}
\lambda_1^* &= n\sigma_j (\frac{1 + (n-1)\frac{L_j}{\sigma_j}}{D^2}) \\
\lambda_2^* &= 0 \\
\lambda_l^* &= (l-2)(\frac{1 + (n-1)\frac{L_j}{\sigma_j}}{D^2})(\sigma_j - L_j); \quad 3 \leq l \leq (n+1)
\end{aligned}$$

Thus from the KT Theorem we know that $(d_1^*, d_2^*, \ldots, d_n^*) = (\frac{\sigma_j D}{\sigma_j + (n-1)L_j}, \frac{L_j D}{\sigma_j + (n-1)L_j}, \ldots, \frac{L_j D}{\sigma_j + (n-1)L_j})$ maximizes the objective function.

On substitution of $(d_1^*, d_2^*, \ldots, d_n^*)$ in the objective function, we obtain the minimum resource allocation to be:

$$r_j^{(i)} = \frac{\sigma_j}{D}[1 + \frac{(n-1)L_j}{\sigma_j}] \text{ for } 1 \le i \le n$$

Proof of Theorem 2.5: Let $(1 - \frac{L_j}{\sigma_j}) = a$. The optimization problem in this case may be written as:

$$\text{Minimize} \quad -\left(\frac{\sigma_j}{d_1 - \frac{L_{max}}{C_1}} + \frac{\sigma_j}{(d_1 - \frac{L_{max}}{C_1})a + (d_2 - \frac{L_{max}}{C_2})} + \cdots + \frac{\sigma_j}{(d_1 - \frac{L_{max}}{C_1})a^{n-1} + \cdots + (d_n - \frac{L_{max}}{C_n})}\right)$$

$$\text{subject to } D - d_1 - d_2 - \ldots - d_n \ge 0 \tag{A.6}$$

$$(d_1 - \frac{L_{max}}{C_1}) \ge 0 \tag{A.7}$$

$$(d_2 - \frac{L_{max}}{C_2}) + (a-1)(d_1 - \frac{L_{max}}{C_1}) \ge 0 \tag{A.8}$$

$$\vdots$$

$$(d_1 - \frac{L_{max}}{C_1})a^{n-2}(a-1) + \cdots + (d_{n-1} - \frac{L_{max}}{C_{n-1}})(a-1) + (d_n - \frac{L_{max}}{C_n}) \ge 0 \tag{A.9}$$

On substituting $d_i - \frac{L_{max}}{C_1} = d_i^{'}$; $1 \le i \le n$, the above problem is of the same form as that in Theorem 4. Thus the proof of this problem is along the same lines as that of Theorem 4 and we obtain the optimal split as:

$$\begin{aligned} d_1 &= \frac{\sigma_j(D - \sum_{l=1}^{n} L_{max}C_l)}{\sigma_j + (n-1)L_j} + \frac{L_{max}}{C_1} \\ d_i &= \frac{L_j(D - \sum_{l=1}^{n} L_{max}C_l)}{\sigma_j + (n-1)L_j} + \frac{L_{max}}{C_i}; \quad 2 \le i \le n \end{aligned}$$

On substitution of $(d_1^*, d_2^*, \ldots, d_n^*)$ in the objective function, we obtain the minimum resource allocation as: $r_j^{(i)} = \frac{\sigma_j + (n-1)L_j}{D - \sum_{l=1}^{n} \frac{L_{max}}{C_l}}$; $1 \le i \le n$

Bibliography

[1] J.C.R. Bennet and H. Zhang. wf^2q: Worst-case fair weighted fair queuing. In *Proceedings of IEEE INFOCOM*, pages 120–128, March 1996.

[2] R. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Trans. on Information Theory*, 37(1):114–131, January 1991.

[3] G. de Veciana and G. Kesidis. Bandwidth allocation for multiple qualities of service using generalized processor sharing. *IEEE Trans. Inform. Th.*, 42(1):268–271, January 1996.

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

[5] A. Elwalid and D. Mitra. Design of generalized processor sharing schedulers which statistically multiplex heterogeneous qos classes. In *Proceedings of IEEE INFOCOM*, 1999.

[6] K. Kumaran *et al.* Novel techniques for the design and control of generalized processor sharing schedulers for multiple qos classes. In *Proceedings of IEEE INFOCOM*, 2000.

[7] R. Szabó *et al.* Call admission control in generalized processor sharing schedulers using non-rate proportional weighting of sessions. In *Proceedings of IEEE INFOCOM*, 1999.

[8] R. Szabó *et al.* Non-rate proportional weighting of generalized processor sharing schedulers. In *Proceedings of IEEE GLOBECOM*, 1999.

[9] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan. Efficient network qos provisioning based on per node traffic shaping. In *Proceedings of IEEE INFOCOM*, pages 102–110, March 1996.

[10] S. Golestani. Network delay analysis of a class of fair queuing algorithms. In *Proceedings of IEEE INFOCOM*, pages 636–646, April 1994.

[11] P. Goyal, S. S. Lam, and H.M. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proc. Workshop on Network and Operating System Support for Digital Audio and Video*, pages 287–289, April 1995.

[12] P. Goyal and H. Vin. Generalized guaranteed rate scheduling algorithms: A framework. *IEEE/ACM Trans. on Networking*, 5(4):561–571, August 1997.

[13] R. Guérin and A. Orda. Qos-based routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Trans. on Networking*, 7(3):350–363, June 1999.

[14] R. Guérin and V. Peris. Quality-of-service in packet networks: basic mechanisms and directions. *Computer Networks*, 31:169–189, 1999.

[15] D.H. Lorenz and A. Orda. Optimal partition of qos requirements on unicast paths and multicast trees. In *Proceedings of IEEE INFOCOM*, pages 246–253, March 1999.

[16] R. Nagarajan, J.F. Kurose, and D. Towsley. Allocation of local quality of service constraints to meet end-to-end requirements. In *IFIP Workshop on the Performance Analysis of ATM Systems*, January 1993.

[17] D. Nandita, J. Kuri, and H.S. Jamadagni. Optimal call admission control in a network of generalized processor sharing schedulers. Technical Report TR-00-05, Centre for Electronics Design and Technology, Indian Institute of Science, http://shravana.cedt.iisc.ernet.in/~dnandita/, June 2000.

[18] D. Nandita, J. Kuri, and H.S. Jamadagni. Optimal call admission control in generalized processor sharing schedulers. Technical Report TR-00-02, Centre for Electronics Design and Technology, Indian Institute of Science, http://shravana.cedt.iisc.ernet.in/~dnandita/, June 2000.

[19] D. Nandita, J. Kuri, and H.S. Jamadagni. Optimal resource allocation in packet networks that use rate based schedulers. In *Proceedings of IEEE Conference in Communications, Controls and Signal Processing*, July 2000.

[20] D. Nandita, J. Kuri, and H.S. Jamadagni. Optimal resource allocation in packet networks that use rate based schedulers. Technical Report TR-00-01, Centre for Electronics Design and Technology, Indian Institute of Science, http://shravana.cedt.iisc.ernet.in/~dnandita/, June 2000.

[21] A. Orda. Routing with end-to-end qos guarantees in broadband networks. *IEEE/ACM Trans. on Networking*, 7(3):365–374, June 1999.

[22] A. K. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Dept. Elec. Engg. Comput. Sci., M.I.T., 1992.

[23] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. on Networking*, 1(3):344–357, Jun. 1993.

[24] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Trans. on Networking*, 2(2):137–150, Apr. 1994.

[25] D. Stiliadis and A. Varma. Rate-proportional servers: A design methodology for fair queuing algorithms. *IEEE/ACM Trans. on Networking*, 6(2):164–174, April 1998.

[26] O. Yaron and M. Sidi. Generalized processor sharing networks with exponentially

bounded burstiness arrivals. In *Proceedings of IEEE INFOCOM*, pages 628–634, June 1994.

[27] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. IEEE*, 83(10):1374–1396, October 1995.

[28] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM*, pages 19–29, August 1990.

[29] Z.L. Zhang, Z. Liu, J. Kurose, and D. Towsley. Call admission control schemes under generalized processor sharing scheduling. *Telecommunication Systems*, In Print.

[30] Z.L. Zhang, Z. Liu, and D. Towsley. Closed-form deterministic end-to-end performance bounds for the generalized processor sharing scheduling discipline. *Journal of Combinatorial Optimization (Special Issue on Scheduling)*, In Print.

[31] Z.L. Zhang, D. Towsley, and J. Kurose. Statistical analysis of the generalized processor sharing scheduling discipline. *IEEE Journal on Selected Areas in Communications*, 13(6):1071–1080, Aug. 1995.