ADMISSION AND FLOW CONTROL IN GENERALIZED PROCESSOR SHARING SCHEDULERS

Ph.D. Theses

By

Róbert Szabó

Research Supervisors:

Dr. Tibor Trón Dr. József Bíró

Department of Telecommunications and Telematics

Budapest University of Technology and Economics

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
BUDAPEST UNIVERSITY OF
TECHNOLOGY AND ECONOMICS
BUDAPEST, HUNGARY
MAY 2000

BUDAPEST UNIVERSITY OF
TECHNOLOGY AND ECONOMICS

Date: **May 2000**

Author: **Róbert Szabó**

Title: **Admission and Flow Control in Generalized Processor Sharing Schedulers**

Department: **Telecommunications and Telematics**

Degree: **Ph.D.** Convocation: Year: **2000**

Permission is herewith granted to Budapest University of Technology and Economics to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

To the precious memory of my Father.

Table of Contents

List of Tables

List of Figures

Abstract

In packet service disciplines Generalized Processor Sharing (GPS) is reported as the most important ideal fluid scheduling discipline in guaranteed quality of service (QoS) networks, which supports well defined delay and loss bounds on leaky bucket constrained traffic. Its packetized versions (WFQ, WF^2Q, etc...) are considered as the packet scheduler of choice in IP routers and switches of the future. These practical packet scheduling algorithms are based on GPS, but all of them bear the bandwidth-delay coupling property which is the well-known problem of GPS-based rate-proportional servers that leads to inefficient resource utilization. Though statistical analysis of GPS schedulers has been studied in the literature [ZTK94], in many cases the approach for handling and design GPS-based schedulers is deterministic. However, the commonly used deterministic bounds are too loose mainly due to the rate-proportional weighting. Recently we have introduced the concept of non rate-proportional (or arbitrary) weighting of sessions in GPS systems. Such an approach in a GPS node of network constrained by leaky buckets can handle bandwidth and delay parameters independently, thus allowing better utilization of network resources. A numerically inexpensive algorithm, which works in any arbitrary weighted GPS system has also been developed for computing the (often tighter) delay bounds independently of weight assignments.

Former call admission control (CAC) solutions were based on the bandwidth-delay coupled system and were overly conservative due to the applied loose delay bound that led to poor network utilization. In the case of arbitrary weighting, which further relax the drawback of bandwidth and delay coupling of sessions, the weight assignment is a highly non-trivial task. In this work we propose a complete solution for CAC and weight

assignment using a new computationally effective recursive algorithm taking into account our tighter delay bounds. Furthermore, beside considering the quality of service class we also present a CAC algorithm with weight assignment that can handle best-effort services as well. Besides the analytical framework numerical examples as well as the complexity analysis of different CAC algorithms are also presented. We believe that our work can establish a new deterministic approach to control future QoS IP network resources more efficiently. Our proposed approach suits IETF's Integrated Services model and could be used to extend the Guaranteed Quality of Service network element.

Acknowledgements

Several people supported my efforts during my graduate years that I would like to thank them all. I owe special thanks to Dr. Gábor Fodor who put me to the right track to go for a Ph.D. I would also like to express my greatest gratitude to Dr. József Bíró, Péter Barta and Felicián Németh who helped me achieving all the results contained in the present dissertation.

I thank the selfless support coming from the High Speed Network Laboratory especially from my mentor Dr. Tibor Trón and our research lab manager Dr. Tamás Henk.

My fruitful relation to the industry is thanked to Dr. Miklós Boda, who first introduced me to Ericsson's Applied Research Laboratory - SwitchLab. Here, thanks to the informal manner and the selfless support coming from all the members of SwitchLab we could build up a relationship I could always rely on. Without completeness I would like to mention Dr. Carl-Gunnar Pertz, who as the manager of SwitchLab, made me possible to have experience in their laboratory and sponsored me in conferences. Dr. Patrik Evaldsson and Harald E. Brandt also helped a lot concerning issues related to routing and quality of service in IP networks. I would still like to mention Gunnar Olsson - a networking guru -, Anders Eriksson who with his ever up to date view of things going in the Internet community gave me stimulating discussions.

Last but not least, the environment I was "growing up" in. First of all my fellow students whose support was essential to get here, namely Tamás Marosits, Balázs Szviatovszki, Szabolcs Malomsoky etc.... Also former students later Ph.D. students that I worked with

played very important role in my studies just to mention Csaba Lukovszki and Ádám Marquetant.

Finally, with my highest respect I would like to thank the endless support coming from my family. Throughout my whole studies they provided me the necessary background, stimulated me to go further and supported me with the greatest thing in the world, i.e., their love. My father, without whom I would never be what I am now and whom I will always see as an ideal unfortunately could not live to see my success. This work is dedicated to his precious memory!

The research done herein was supported by Ericsson Applied Research - SwitchLab.

Chapter 1

Introduction

Nowadays communication networking trends are moving toward high speed packet switching networks like ATM or future IP based solutions. These technologies will replace the telephone-type circuit switched networks as soon as their performance guarantees besides other attributes become satisfactory. Though the wired telephone network provides good performance guarantees it lacks flexibility and wastes resources when used for accessing new types of services. These new services lay the challenging task of ensuring quality of service (QoS) guarantees on these integrated networks while application demands increase dramatically and the number of Internet users are growing exponentially.

More specifically, controlling and handling network congestion have been intensively researched during the last several years. Different, sometimes even conflicting approaches have appeared in the literature. Nevertheless, all of them agreed in one question i.e. the need of *traffic scheduling algorithm* [BZ94, BZ96, CSZ92, Gol90, Gol94, Zha95, SV96, SV97, DLS98][1] to efficiently handle QoS guarantees.

The reason why the circuit switched telephony network failed the competition with the integrated services packet switched approach is its lack of efficient adaptation to recent and

[1]Throughout the dissertation we used 4 types of bibliography notations, i.e., standard Harvard for literature references while our publications are denoted by J - indicating journal papers, C - indicating conference papers and W - indicating workshops or others

future communications need. Nowadays popular applications like multimedia communications (voice, video) are using advanced coding techniques thus can be characterized by varying rate. The circuit switched telephony network with its peak rate allocation does not efficiently support these needs, i.e. it lacks the possibility of *statistical multiplexing* unlike ATM and the packet switched Internet. However with the gain of statistical multiplexing networks may become *congested* that degrade their performance either for short or longer terms. User bursts may accumulate in buffers hence degrading the performance of the whole system. Also users may misbehave or be greedy and try to gain access to more resources than their share or contract thus degrading the performance of other conforming traffics. Hence the need for some control mechanism that prevents or recovers congestion is evident. This control mechanism must support a certain separation of traffics or traffic types while also ensuring certain performance guarantees.

The offered service of a packet switched router or a switch depends on the arrival pattern of the application's traffic, the number of available buffer and congestion information from the neighbors. Most importantly it depends on the mechanism that determines, which packet to transmit next on the output link. The control mechanism establishing this function is referred as *traffic scheduling* algorithm.

Traffic management - including the scheduling - plays a key role in operating the network efficiently and also providing the desired level of quality for the users. Traffic management algorithms can be classified in several ways with respect to different characteristics. The time-scale of the effective operation of such algorithms can be cell/packet, burst or call/connection. One can also group these algorithms in terms of the underlying control options, i.e. there can be preventive or reactive approaches for traffic control. Packet service disciplines are good examples for cell/packet level algorithms whose task is to provide end-to-end performance bounds by means of classifying and scheduling the traffic. Among call/connection level preventive traffic control approaches *admission control* is one of the most important ones. The main task of such algorithms is to decide whether the newly arrived connection can enter the network without performance degradation of the other existing connections or not. Because current IP-based networks tend to provide real-time/QoS

services other than best-effort, appropriate services including resource reservation, call admission control and scheduling mechanisms must be devised. Both of the above-mentioned traffic management algorithms will play very important role in future QoS guaranteed IP-based networks. The IETF Integrated Services framework clearly declared the need for packet schedulers and admission control for Controlled Load Service [Wro97] and Guaranteed Quality of Service [SPG97] network elements, however, the question how to do that is left to the researchers and developers to solve. The efficient integration of scheduling and admission control algorithms can also be very crucial in providing quality of service for IP, although, it was not addressed extensively so far by the research community [ZLKT97].

In this dissertation we first propose some novel extensions to the well known Generalized Processor Sharing scheduling discipline that relaxes its bandwidth and delay coupling. Thus the investigated system bears tighter delay bounds than that introduced by Parekh et al. in [PG93a]. The introduction of the bandwidth and delay decoupled system has the drawback of losing the unambiguous weight assignment property of the coupled system. Therefore, characteristics of arbitrary weighted GPS systems are analysed and algorithms to assign weights, control flow and admission are proposed. These algorithms are also performance evaluated in the means of computational complexity. Numerical examples illustrating the reason for the existence of our introduced approach are also shown. Finally, application possibilities are mentioned and conclusion is drawn.

The structure of the dissertation is as follows. First, a short introduction is given to the class of fair schedulers also detailing Generalized Processor Sharing in Chapter 2. In Chapter 3 a new approach of setting session weights are presented while Chapter 4 gives an algorithmic approach to calculate deterministic worst-case delay bounds with the weighting introduced. Next, in Chapter 5 detailed delay analysis of arbitrary weighted GPS schedulers are presented. Lastly in Chapter 6, connection admission control methods are investigated for a single node of GPS server and application possibilities are shortly discussed in Chapter 7.

Chapter 2

Fair Scheduling Revisited

The historically first fair queueing algorithm was proposed by Nagle in [Nag85]. His main contribution was to provide a fair allocation of network resources among concurrent users. It was achieved by replacing the single first-in first-out (FIFO) queue associated with each outgoing link with multiple queues, one for each source host in the entire network. These queues were served in a round-robin fashion, taking one packet from each non-empty queue in turn and transmitting them on the appropriate outgoing link. Empty queues are skipped over and lose their turn.

Though this queueing algorithm did not provide fair bandwidth allocation in case of varying packet sizes, it offered some kind of protection against ill-behaved sources (separation).

2.1 Bit-by-Bit Round Robin

Since Nagle's proposal [Nag85] left packets sizes out of consideration, it was necessary to develop a new algorithm that provided fair bandwidth and buffer allocation. This new algorithm was described by Demers et al. in [DKS89][DKS90].

In order to achieve fair bandwidth allocation Demers et al. introduced a hypothetical service discipline, named bit-by-bit round robin. This bit-by-bit round robin discipline is an idealized one that does not transmit packets as entities. The server can serve multiple

sessions simultaneously and the traffic is infinitely divisible. It is clear that this kind of service discipline cannot be implemented in reality. Therefore a packetized version was derived that emulated the bit-by-bit round robin discipline in the background. Packets of the real system were transmitted in the order as their corresponding service order in the idealized model.

Of course, the bit-by-bit round robin and its packetized version could be in disharmony, though the maximal lag of the two systems were limited by the maximal packet sized further denoted by $L_{\max}$.

Even though the bit-by-bit round robin and its packetized version already implemented total separation of different connections it lacked the ability to handle connections with different bandwidth requirements. Its extension that handles bandwidth demands too is known as *weighted fair queueing* or generalized processor sharing as its idealized model [PG93a][PG92][PG94][PG93b].

2.2 Leaky Bucket

Though the leaky bucket or equivalently named token bucket scheduler does not belong to the class of fair schedulers its role in traffic shaping and characterization that we will later use demand a short description here.

In a leaky bucket, traffic enters an infinite queue as shown in Figure 2.1. The output of this infinite queue is regulated by the means that traffic can only be transmitted by the same amount of tokens, i.e., if no token present no traffic is transmitted. Note that this introduces non work-conserving property. Tokens are generated at a rate of ρ and are limited to the size of σ.

Now, if the leaky bucket is used to characterize traffic streams then for a stream with rate $R(t)$ is said to satisfy such a constraint if there exist some constant ρ and σ such that

$$\int_s^t R(u)\mathrm{d}u \leq \rho(t-s) + \sigma \tag{2.2.1}$$

holds for all $0 \geq s < t$[Cru91a, Cru91b]. This concept ignores the stochastic nature of the traffic, and must hold for any sample path of it.

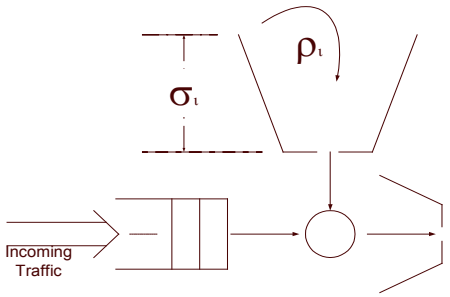


Figure 2.1: Leaky bucekt

2.3 Generalized Processor Sharing (GPS)

The Generalized Processor Sharing service discipline first appeared in [DKS90]. Let us recall the definition of GPS.

Definition 2.3.1. A **GPS server** serving N sessions is characterized by N positive real numbers, $\phi_1, \phi_2, \ldots, \phi_N$. The server operates at a *fixed rate* r and is work-conserving. A server is work-conserving if it is never idle whenever there are packets to send. Let $W_i(t_1, t_2)$ be the amount of session i traffic[1] served in the interval $[t_1, t_2]$, then a GPS server is defined as one for which

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j} \quad j = 1, 2, \ldots, N, \tag{2.3.1}$$

for any session i that is continuously backlogged[2] in the interval $[t_1, t_2]$.

Further, the server is said to be *stable* if $\sum_{i=1}^{N} \rho_i < r$, i.e., the sum of long term sustained rates of sessions is less than the service rate. The immediate consequence of this definition is that every session has a minimum guaranteed service rate which is

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r. \tag{2.3.2}$$

The structure of the system is visualized in Figure 2.2. Every session has its own separate queue in which the traffic (packets) are stored until they become eligible for service.

[1] no assumptions on traffi c characteristics are required

[2] the session buffer is not empty

The sessions share the server capacity according to their assigned weights (ϕ). Each input traffic for the session queues is shaped by leaky bucket [Wor94, YSS92, Rob90] that is characterized by a token pool depth (σ) and a token generation rate (ρ) as described previously. Although the use of leaky bucket is not necessary for GPS, it can be beneficial, because the network needs some kind of rate admission control on the input traffic. Furthermore, as Parekh and Gallager have identified in [PG93a] the usage of leaky bucket allows the separation of packet delays into two components: delay in the leaky bucket and delay in the network. The first component is independent of other (active) sessions, while the second one is independent of the incoming traffic.

The output of a leaky bucket provides the traffic entering for schedulers. In this way, the amount of incoming traffic in the interval $[t_1, t_2)$ from an active source i can be characterized by the function $A_i(t_1, t_2)$. If C_i, the capacity of the link interconnecting the leaky bucket with the queue of the GPS, is finite then greedy arrival pattern is described by

$$A_i(0,t) = \min\{C_i t, \sigma_i + \rho_i t\} \tag{2.3.3}$$

while if $C_i = \infty$ then

$$A_i(0,t) = \sigma_i + \rho_i t. \tag{2.3.4}$$

This way *greedy* behavior is interpreted as session i starts with its maximal burst at time zero and continues to transmit with its maximal rate ρ_i. If all sessions start greedy one gets a *greedy GPS system*.

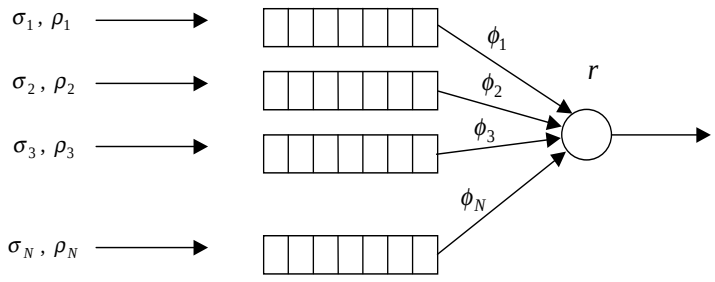


Figure 2.2: The GPS scheduler

Since the minimum throughput of a GPS server is given by (2.3.2) we are rather interested in delay characteristics. By the delay of a GPS server we mean the amount of time a traffic unit spend in the GPS buffer before transmitted. Note that the leaky bucket delay is not accounted here. A conservative delay calculation for GPS has been given by Parekh and Gallager in [PG93a] though throughout this dissertation we give a method to calculate tighter delay bounds.

This service discipline is an ideal (fluid) model in which the traffic is considered as infinitely divisible and every session is being served simultaneously sharing the server capacity. Although such a system can not be implemented in practice GPS has the following significance. On one hand many practically realizable schedulers are based on GPS, that is several schedulers emulate a GPS server at the background during their operation maintaining state information on every session. The packet service order in the packetized (packet-by-packet) schedulers is based on the "so-to-say" packet service process in GPS. On the other hand, GPS thanks to its idealized bandwidth sharing property is commonly regarded as an ideal *reference* system to which the packet versions are compared.

A possible packet-by-packet version of GPS called PGPS is introduced in [PG93a] and the relation between GPS and PGPS is also analyzed with respect to the finishing times of packets and the service received by sessions under GPS and PGPS. The results are as follows.

Theorem 2.3.1 ([PG93a]). *For all packets p,*

$$\hat{F}_p - F_p \leq \frac{L_{max}}{r}, \tag{2.3.5}$$

where L_{max} is the maximum packet length[3] *and r is the (constant) service rate of the server, $\hat{F}_p$, F_p is the finishing times of packet p under PGPS and GPS, respectively.*

Let $W_i(t_1, t_2)$ and $\hat{W}_i(t_1, t_2)$ be the service received by the session i under GPS and PGPS in the interval $[t_1, t_2]$.

[3] *used to call maximum transfer unit (MTU) in the IP world*

Theorem 2.3.2 ([PG93a]). *For all times τ and session i:*

$$W_i(0,\tau) - \hat{W}_i(0,\tau) \leq L_{max} \tag{2.3.6}$$

holds for any session i that is backlogged throughout the interval $[0,\tau]$.

These results can also be used for establishing relation between the performance bounds obtained for GPS and PGPS, i.e., most of the cases it is enough to analyze the fluid GPS model and then the bounds can be transformed to PGPS.

The most important result of [PG93a] is as follows:

Theorem 2.3.3 ([PG93a]). *Suppose that $C_j > r$ for every session j, where C_j is the internal link capacity between the session j leaky bucket and the session j queue, and r is the GPS server capacity. Then, for every session i, the maximum delay D_i and the maximum backlog Q_i are achieved (not necessarily at the same time) when every session is greedy starting at time zero, the beginning of a system busy period.*

The significance of this result is that for worst case behavior one 'only' has to analyze a greedy system, which makes the analysis more tractable compared to any arbitrary arrival patterns imposed to the system.

Therefore, hereafter in the paper without losing generality we only consider greedy GPS systems, where all sessions start greedy at time zero, the beginning of a system busy period. Theorem 2.3.3 is also valid when the internal link speeds are infinite ($C_i = \infty$). Thus, we perform our analysis for the infinite capacity case , however, the extension to the finite capacity case is straightforward [Bou96]. Note, that results obtained with infinite capacity links bound similar results with finite capacities.

Another important result from former works regards the delay bounds of a GPS node. Parekh and Gallager gave a worst case delay bound for a special case of a GPS systems, i.e., when the bandwidth is coupled to the delay.

Theorem 2.3.4 ([PG94, PG93b]). *If $g_i \geq \rho_i$ for session i, then*

$$Q_i \leq Q_i^* = \sigma_i \tag{2.3.7}$$

$$D_i \leq D_i^* = \frac{\sigma_i}{g_i} \tag{2.3.8}$$

where

$$g_i = \frac{\phi_i}{\sum_{i=1}^{N} \phi_i} r, \tag{2.3.9}$$

Q_i, D_i are defined by Theorem 2.3.3 and Q_i^*, D_i^* are the worst case backlog and delay bounds for session i, respectively.

Chapter 3

Arbitrary Weighting of GPS Servers

A large subclass of GPS-based schedulers is the rate-proportional servers (RPS)[1], in which weights are set according to session bandwidth demands. It introduces coupling between bandwidth and delay, i.e., the delay bound is inversely proportional to the long term allocated bandwidth (ρ).

For instance, in Rate-Proportional Processor Sharing (RPPS), which is a special case of WFQ or PGPS, ϕ's are allocated proportional to the bandwidth required by connections. If the traffic is constrained by leaky bucket parameters (σ, ρ) the delay bound of the connection will be $(\sigma + L_{max})/\rho$ where L_{max} is the maximum packet size of the connection [Zha95]. Note that the delay bound is inversely proportional to the long term allocated bandwidth (ρ).

In what follows we consider a set of N sessions (traffic sources) to be scheduled by a GPS server, each constrained by a separate leaky bucket with parameters (σ_i, ρ_i), where σ_i is the token pool depth and ρ_i is the token generation rate for the i^{th} session.

Moreover, to every session a weight is attached, which determines the minimum guaranteed service rate for that session. As previously, let us denote the weight of session i by ϕ_i.

In the case of rate-proportional weighting session i is assigned a weight proportional to its long term sustained rate, i.e., its ρ_i. Even though rate-proportional weighting makes

[1] the definition of RPS used here does not coincide with the one in [SV98]

analysis and implementation of a single GPS node or network of GPS nodes simpler, it comes with a big drawback. Namely, the simple upper bounds for delay and bandwidth demand is strongly coupled to each other. It means that if one wants to decrease the delay bound for a session he/she also has to increase the corresponding allocated weight/bandwidth.

For making the situation more clear, let's assume that one would like to half the delay bound for session i. Since the worst case delay is σ_i/g_i where $g_i = \phi_i/\sum_j \phi_j$ (see Theorem 2.3.4), ϕ_i should be doubled, which results also double guaranteed rate of service for that session. In this way, due to the rate-proportional weighting of sessions, the long term sustainable throughput ρ_i should also be doubled. This concludes that the more delay sensitive a session is the more bandwidth it requires. Nevertheless, there can be sessions like voice over IP (VoIP) sessions, which are delay sensitive but do not require high bandwidth compared to other sessions. In this case, bandwidth (server capacity) might be wasted for providing acceptable upper bound for delay.

An other problem that arises when using rate proportional weighting is that the worst case delay bounds proposed by Parekh et al. in Theorem 2.3.4 turned to be too loose in general as we will show later. Event though the computation of these bounds is quite simple they fail to capture an important property of GPS servers, i.e., the service rate updates that we will describe in details in the following sections.

Summarizing, the above the problem is two-fold: on one hand the delay bounds for sessions are quite loose, although they are also very simple. On the other hand, these bounds are inversely proportional to the minimum guaranteed service rate and also to the long term sustainable rate due to the rate-proportional weighting of sessions.

Now, we are about to present a solution for the two-fold problem. First, we consider a case when the weight ϕ_i is set independently of the parameter ρ_i. This is referred to as *non-rate-proportional weighting* or *arbitrary weighting* of sessions. For detailed weighting scenarios see Figure 3.1. After characterizing the behaviour of the non-rate-proportional system, in the following sections we provide an algorithmic approach to compute delay

bounds, which i) are tighter than the traditional ones in the case of rate proportional weighting; ii) hold not only for locally stable sessions but also for any session regardless of its weight assignment.

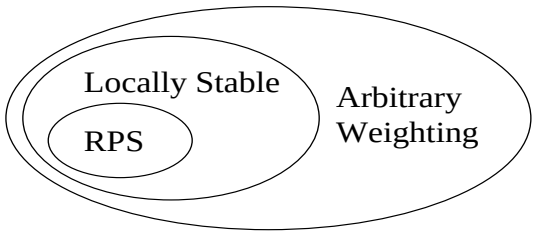


Figure 3.1: Weighting possibilities for sessions

As it was shown in the work of Parekh et al. in [PG93a] for a given set of parameters (σ, ρ, ϕ) the worst case delay and backlog for every session occurs when all sessions start greedy, i.e., all sessions start to send data simultaneously with maximum possible rate. Let us assume that ϕ_i's are set in a non-rate-proportional way. Thus, for some sessions the minimum guaranteed service rate can be smaller than the corresponding token generation rate ρ. Since $\sum_{i=1}^{N} \rho_i$ is kept below r for stability and $\sum_i g_i = r$ there should be at least one session for which $g_i > \rho_i$. Backlogs of these sessions during a system busy period are decreasing and finally disappearing. Now pick the session(s) whose backlog disappears first. After the time instant when the session backlog becomes zero, the session is served by the rate of its corresponding ρ. Note that ρ is strictly lower than the previously allocated g, since the session's backlog has been emptied. The excess bandwidth, i.e., $g - \rho$ is distributed among other sessions still having backlogs according to their weights. Thus, at every time instant a backlog is cleared, service rates of sessions still backlogged increase. It is easy to see that if $\sum_{i=1}^{N} \rho_i < r$, then the system is stable and every initial backlog will be cleared within the system busy period regardless the actual weights (ϕ_i's).

3.1 Service rates of a GPS server

Now, let us turn on a more formal description of the system behaviour. For the sake of simplicity and without losing generality let $r = 1$, or in other words we consider all the

quantities normalized to the server rate r.

Definition 3.1.1 ($\mathcal{L}$). Let $\mathcal{L} = \{L(i)|\ i = 1, \ldots, N\}$ be a mapping function in which $L(i)$ means that the session $L(i)$ backlog is cleared as i^{th} in order. Since $L(i)$ is a one-to-one mapping, it has an inverse $L^{-1}(j)$, which says that session j clears its backlog at the $L^{-1}(j)^{\text{th}}$ step.

Let us denote the time instant by t_i when the backlog of session $L(i)$ becomes zero or more formally:

Definition 3.1.2 (Backlog clearing time). The backlog clearing time of session i is defined as the minimal t where the service function (W_i) is equal to the arrival function (A_i) of a greedy system. In a mathematical form:

$$t_i = \min\{\ t \mid t \geq 0 \wedge W_i(t_0, t) = A_i(t_0, t)\}, \tag{3.1.1}$$

where t_0 is the beginning of a system busy period where all sessions starting greedily.

If we further assume $t_0 = 0$, the start of the system busy period, then the above formula will reduce to

$$t_i = \min\{\ t \mid t \geq 0 \wedge W_i(t) = A_i(t)\}. \tag{3.1.2}$$

Definition 3.1.3 (r_k^i). Let r_k^i be the service rate of session i prior to the k^{th} backlog clear, i.e.,

$$r_k^i = r_i(t) \quad t \in [t_{k-1}, t_k), \quad k = 1, \ldots, N. \tag{3.1.3}$$

By definition, after the N^{th} backlog clear all sessions are served with their arrival rates[2], i.e., for each session i

$$r_N^i = \rho_i. \tag{3.1.4}$$

During the interval $[0,\ t_1)$ every session is served by its minimum guaranteed service rate, i.e.,

$$r_1^j = g_j. \tag{3.1.5}$$

[2]note the greedy system

Between t_1 and t_2 session $L(1)$ has no backlog, so it is served by $\rho_{L(1)}$. Its excess rate is re-allocated among other still backlogged sessions, in proportion to their original weights, therefore their service rates increase to

$$r_2^j = g_j + \frac{\phi_j}{\sum_{i=1}^N \phi_i - \phi_{L(1)}}(g_{L(1)} - \rho_{L(1)}),\ j \in \{1,\dots,N\} \setminus \{L(1)\}. \tag{3.1.6}$$

The amount of bandwidth $L(1)$ releases after clearing its backlog is the difference between its guaranteed and sustained rate, i.e., $(g_{L(1)} - \rho_{L(1)})$ as seen in the former expression.

Theorem 3.1.1 (Discrete Rate Function [C6, C7, C9]). *In the time interval* $[t_{k-1},t_k)$, $k = 1,\dots,N$ *within the system busy period the backlogged session* j *is served at a rate*

$$r_j(t) = r_k^j = \frac{(1 - \sum_{l=1}^{k-1} \rho_{L(l)})\phi_j}{\sum_{i=1}^N \phi_i - \sum_{l=1}^{k-1} \phi_{L(l)}}\ r,\ j \in \{1,\dots,N\} \setminus \{\bigcup_{m=1}^{k-1} L(m)\}. \tag{3.1.7}$$

Proof. The theorem is to be proved by induction. First, we check that the statement is valid for r_m^j when $m = 2$. From equation (3.1.6)

$$r_2^j = \frac{\phi_j}{\sum_{i=1}^N \phi_i} + \frac{\phi_j}{\sum_{i=1}^N \phi_i - \phi_{L(1)}} \frac{\phi_{L(1)}}{\sum_{i=1}^N \phi_i} - \frac{\phi_j}{\sum_{i=1}^N \phi_i - \phi_{L(1)}} \rho_{L(1)}. \tag{3.1.8}$$

Since

$$\frac{\phi_j}{\sum_{i=1}^N \phi_i} + \frac{\phi_j}{\sum_{i=1}^N \phi_i - \phi_{L(1)}} \frac{\phi_{L(1)}}{\sum_{i=1}^N \phi_i} = \frac{\phi_j}{\sum_{i=1}^N \phi_i - \phi_{L(1)}} \tag{3.1.9}$$

the following equality holds

$$r_2^j = \frac{(1 - \rho_{L(1)})\phi_j}{\sum_{i=1}^N \phi_i - \phi_{L(1)}}. \tag{3.1.10}$$

Now, let us assume that the statement in the theorem is valid for $m = k - 1$, which means that

$$r_{k-1}^j = \frac{(1 - \sum_{l=1}^{k-2} \rho_{L(l)})\phi_j}{\sum_{i=1}^N \phi_i - \sum_{l=1}^{k-2} \phi_{L(l)}},\ j \in \{1,\dots,N\} \setminus \{\bigcup_{v=1}^{k-2} L(v)\}. \tag{3.1.11}$$

The question is how this rate changes when the session $L(k-1)$ backlog is cleared at time instant t_{k-1}. The session $L(k-1)$ service rates are $r_{k-1}^{L(k-1)}$ and $\rho_{L(k-1)}$ respectively

before and after emptying its backlog. In this way the remaining capacity $r_{k-1}^{L(k-1)} - \rho_{L(k-1)}$ is distributed among sessions still backlogged in the system proportional to their weights. So one can write

$$r_k^j = r_{k-1}^j + \frac{\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}}(r_{k-1}^{L(k-1)} - \rho_{L(k-1)}), \tag{3.1.12}$$

$$\text{where } j \in \{1,\ldots,N\} \setminus \{\bigcup_{v=1}^{k-1} L(v)\}.$$

Substituting r_{k-1}^j from equation (3.1.11) it is obtained that

$$\begin{aligned} r_k^j &= \frac{(1-\sum_{l=1}^{k-2}\rho_{L(l)})\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-2}\phi_{L(l)}} + \\ &+ \frac{\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}}\left(\frac{(1-\sum_{l=1}^{k-2}\rho_{L(l)})\phi_{L(k-1)}}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-2}\phi_{L(l)}} - \rho_{L(k-1)}\right). \end{aligned} \tag{3.1.13}$$

Since

$$\begin{aligned} \frac{(1-\sum_{l=1}^{k-2}\rho_{L(l)})\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-2}\phi_{L(l)}} &+ \frac{(1-\sum_{l=1}^{k-2}\rho_{L(l)})\phi_{L(k-1)}}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-2}\phi_{L(l)}} \times \\ \times \frac{\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}} &= \frac{(1-\sum_{l=1}^{k-2}\rho_{L(l)})\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}} \end{aligned} \tag{3.1.14}$$

and

$$r_k^j = \frac{(1-\sum_{l=1}^{k-2}\rho_{L(l)})\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}} - \rho_{L(k-1)}\frac{\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}}. \tag{3.1.15}$$

By simplifying the above equation one can conclude that the theorem holds, i.e.,

$$r_k^j = \frac{(1-\sum_{l=1}^{k-1}\rho_{L(l)})\phi_j}{\sum_{i=1}^{N}\phi_i - \sum_{l=1}^{k-1}\phi_{L(l)}}. \tag{3.1.16}$$

□

From the theorem above the corollary stating the increasing attribute of the discrete rate function follows:

Corollary 3.1.2 ([C6, C7, C9]). *r_k^j is a monotone increasing function of k for every backloged session j for which $\phi_j > 0$.*

Proof. Using the formula of (3.1.12) we only have to prove that the increment in the iterative formula is greater than or equal to zero. Now assume that all ϕ's are greater than zero, i.e., a non-degenerated GPS system. In this case the first factor of the multiplications is strictly greater than zero. Since $L(k-1)$ identifies the session that last emptied its backlog, its backlog function can be expressed in a form

$$Q_{L(k-1)}(t) \;=\; Q_{L(k-1)}(\tau) + \rho_{L(k-1)}(t-\tau) - r_{k-1}^{L(k-1)}(t-\tau), \qquad (3.1.17)$$

where: $Q_{L(k-1)}(\vartheta)$ is the session $L(k-1)$ backlog (in units of traffic) at time ϑ and τ is the time when session $L(k-2)$ emptied its backlog.

Since at time τ session $L(k-2)$ emptied its backlog and session $L(k-1)$ is still backlogged, so $Q_{L(k-1)}(\tau) > 0$. On the other hand, the next session to empty its backlog is $L(k-1)$, so for some $t > \tau$, $Q_{L(k-1)}(t)$ will be zero, thus yielding for the inequality $r_{k-1}^{L(k-1)} > \rho_{L(k-1)}$, which proves our corollary. □

Note that if a session's weight is zero, it is only served after all other sessions emptied their backlogs. However setting zero weight to more than one session is useless since their share of capacity is undefined in this case. If this scenario is still needed for some reason e.g. traditional best-effort service class has to be partitioned, then methods like hierarchical queueing should be applied to overcome these limitations [BZ96].

In order for us to tell something about the size of the maximum backlog, we have to introduce two lemmas first that will provide bases to determine the maximum backlog.

My first lemma states that at the beginning of the system busy period there is at least one session that starts clearing its backlog. My second lemma is an extension of the former one that puts its result to continuous time.

Lemma 3.1.3 ([C6, C7]). *For a system busy period where all sessions start greedy there is at least one session that starts its service with higher rate than its arrival rate.*

Proof. This Lemma is proved by contradiction. For a stable system $\sum_{i=1}^{N} \rho_i < r$ and the initial service rate, i.e., the guaranteed rate for each session is defined as $g_i = \frac{\phi_i}{\sum_{i=1}^{N} \phi_i}\, r$.

Now assume there exists no session whose guaranteed service rate is higher than its arrival rate, i.e., $g_i \leq \rho_i,\ \forall i \in \{1, \ldots, N\}$. Now from the stability

$$r > \sum_{i=1}^{N} \rho_i \geq \sum_{i=1}^{N} g_i = r \tag{3.1.18}$$

which is a contradiction, thus ensuring the existence of at least one session with strictly higher service rate than its arrival rate that proves our Lemma. □

Lemma 3.1.4 ([C6, C7]). *In a system busy period where all sessions start greedy, at any time t there is at least one session that is served with higher rate than its arrival rate.*

Proof. This Lemma is proved by contradiction. At any time t during the system busy period sessions can be divided into two disjoint sets; one corresponding to those already emptied their backlogs $\mathcal{F}(t)$ and the other still having backlogs at time t further denoted by $\mathcal{B}(t)$. At any time t $\mathcal{F}(t) \cap \mathcal{B}(t) = \emptyset$ and $\mathcal{F}(t) \cup \mathcal{B}(t) = \{1, \ldots, N\}$. We still have the stable system criterion, i.e., $\sum_{i=1}^{N} \rho_i < r$ however this can be reorganized according to the two sets: $\sum_{i \in \mathcal{F}(t)} \rho_i + \sum_{i \in \mathcal{B}(t)} \rho_i < r$.

Further denote session i service rate at time t by $r_i(t)$ in a way that for $\forall k \in \{1, \ldots, N\}$

$$r_i(t) = r_k^i, \ \text{ if } t \in [t_{k-1}, t_k). \tag{3.1.19}$$

Now assume that there exists a t that $r_i(t) \leq \rho_i\ \forall i \in \mathcal{B}(t)$ while $r_i(t) = \rho_i\ \forall i \in \mathcal{F}(t)$. The stability criterion is rewritten according to the grouping:

$$r - \sum_{i \in \mathcal{F}(t)} \rho_i > \sum_{i \in \mathcal{B}(t)} \rho_i \geq \sum_{i \in \mathcal{B}(t)} r_i(t) \tag{3.1.20}$$

From Theorem 3.1.1 one can write

$$r_i(t) = \frac{\phi_i}{\sum_{j \in \{\mathcal{B}(t) \cup \mathcal{F}(t)\}} \phi_j - \sum_{j \in \mathcal{F}(t)} \phi_j} \left(r - \sum_{j \in \mathcal{F}(t)} \rho_j\right) \tag{3.1.21}$$

that is summed for all $i \in \mathcal{B}(t)$ will yield for

$$\begin{aligned} \sum_{i \in \mathcal{B}(t)} r_i(t) &= \frac{\sum_{i \in \mathcal{B}(t)} \phi_i}{\sum_{i \in \mathcal{B}(t)} \phi_i + \sum_{j \in \mathcal{F}(t)} \phi_j - \sum_{j \in \mathcal{F}(t)} \phi_j} \left(r - \sum_{j \in \mathcal{F}(t)} \rho_j\right) \\ &= \left(r - \sum_{j \in \mathcal{F}(t)} \rho_j\right) \end{aligned} \tag{3.1.22}$$

From (3.1.22) the contradiction of (3.1.20) immediately follows, which proves the Lemma. □

Now the theorem that defines where the maximum backlog of each session is achieved.

Theorem 3.1.5 (Maximal Backlog [C6, C7, C9]). *Each session i experiences its maximal backlog either at time zero, when its maximal backlog is equal to σ_i, or at a time instant when its rate increases (changes).*

Proof. From Corollary 3.1.2 it follows that there are at most two phases for each session: one corresponding to the accumulation phase, where the backlog of session j increases and the other that corresponds to backlog emptying phase.

Those sessions starting their services with higher service rates than their arrival rates will only take part in the backlog emptying phase. Since after their initial bursts their backlogs will continuously decreasing, their maximal backlog is limited to their initial burst sizes.

The other set of sessions, which start with smaller service rates than their arrival rates, will step by step increase their rates by each time a session empties its backlog (result of Corollary 3.1.2).

Lemma 3.1.3 and Lemma 3.1.4 ensures that each session sooner or later changes its state from accumulating to backlog clearing, thus at that transition time instant its backlog is maximal. □

The importance of the above theorem is that to determine the maximum backlog one only has to trace backlogs at times sessions clear their backlogs. This allows an efficient algorithmic approach to compute possible backlog maximums.

3.2 Computing backlog clearing times

In the previous section we have seen that the time instances when backlogs are cleared and their order play key role in the dynamics of the traffic service. Here we provide an algorithmic approach to compute the backlog clearing times (see Definition 3.1.2) t_k, $k =$

$1, \ldots, N$. Further, based on these quantities a tighter upper bound for the maximum delay of every session can be calculated.

As the main result of [PG93a] the maximum backlog and delay of sessions occur (not necessarily at the same time) when all sessions are greedy, i.e. they start to send data at the same time and with maximum possible rate.

Theorem 3.2.1 (k^{th} order backlog clearing times [C6, C7, C9]). *The backlog clearing times at the k^{th} step of an arbitrary weighted GPS system can be calculated by the following recursive formula:*

$$t_k = \min\{t_{i,k} |\ t_{i,k} > 0 \wedge i \in \{1, \ldots, N\} \setminus \bigcup_{j=1}^{k-1} L(j)\} \tag{3.2.1}$$

where

$$t_{i,k} = \frac{S_{i,k}}{r_k^i - \rho_i} \tag{3.2.2}$$

and

$$S_{i,k} = \sigma_i + \sum_{j=1}^{k-1} r_j^i\ (t_{j-1} - t_j) + r_k^i\ t_{k-1}. \tag{3.2.3}$$

Hence the recursive formula can be constructed

$$S_{i,k+1} = S_{i,k} + (r_{k+1}^i - r_k^i)\ t_k, \tag{3.2.4}$$

$$S_{i,1} = \sigma_i. \tag{3.2.5}$$

Proof. The theorem is proved by induction. In the first step, we should consider the following N equations which describe the time-evolution of incoming and outgoing (being served) traffic of sessions:

$$\sigma_i + \rho_i(t - t_0) = r_1^i(t - t_0), \quad i = 1, \ldots, N. \tag{3.2.6}$$

On the left-hand side the incoming traffic of session i is expressed until time t and on the right-hand side the amount of served traffic is represented until time t provided the service rate is r_1^i. Assuming the former service rate and the definition of $t_0 = 0$, the time needed for session i to empty its backlog is expressed

$$t_{i,1} = \frac{\sigma_i}{r_1^i - \rho_i}. \tag{3.2.7}$$

It is clear that among t_i's there can be negative values in case of those sessions whose backlog is temporarily increasing. Apparently, that session finishes its backlog first whose $t_{i,1}$ takes the smallest positive value. More formally

$$t_1 = \min\{t_{i,1} \mid t_{i,1} > 0; \;\; i \in \{1, \ldots, N\}\} \tag{3.2.8}$$

and $L(1)$ is the index of session i that satisfies the above minimum.

After session $L(1)$ empties its backlog at time t_1, there remain $N-1$ backlogged sessions in the server. The service rate of session i backlogged is changed (increased) to r_2^i. In the next (second) step for determining t_2 consider the following $N-1$ equations:

$$\sigma_i + \rho_i(t - t_0) = r_1^i(t_1 - t_0) + r_2^i(t - t_1). \tag{3.2.9}$$

After decomposing $\rho_i(t - t_0)$ we can write

$$\sigma_i + \rho_i(t - t_1) + \rho_i(t_1 - t_0) = r_1^i(t_1 - t_0) + r_2^i(t - t_1), \tag{3.2.10}$$

where $i \in \{1, \ldots, N\} \setminus L(1)$.

By reducing the above equation, one can get

$$t_{i,2} = \frac{\sigma_i - r_1^i t_1 + r_2^i t_1}{r_2^i - \rho_i} \tag{3.2.11}$$

as the candidate finishing times of still backloged sessions. Taking the minimum of positive backlog times through $i \in \{1, \ldots, N\} \setminus L(1)$ will yield for the next finishing time of the system

$$t_2 = \min_i\{t_{i,2} \mid t_{i,2} > 0; \;\; i \in \{1, \ldots, N\} \setminus L(1) \;\} \tag{3.2.12}$$

where $L(2)$ corresponds to the session index i that satisfies the minimum.

In general, the k^{th} step can be expressed by the equation

$$\sigma_i + \sum_{j=1}^{k-1} \rho_i(t_j - t_{j-1}) + \rho_i(t - t_{k-1}) = \sum_{j=1}^{k-1} r_j^i(t_j - t_{j-1}) + r_j^i(t - t_{k-1}). \tag{3.2.13}$$

Reducing the above equation for t yields the k^{th} step candidate finishing times of sessions still backlogged

$$t_{i,k} = \frac{\sigma_i + \sum_{j=1}^{k-1}(\rho_i - r_j^i)(t_j - t_{j-1}) + (r_k^i - \rho_i)t_{k-1}}{r_k^i - \rho_i}, \tag{3.2.14}$$

but as $t_0 = 0$ by definition, ρ_i's can be eliminated from the numerator yielding for the following result

$$t_{i,k} = \frac{\sigma_i + \sum_{j=1}^{k-1} r_j^i \, (t_{j-1} - t_j) + r_k^i \, t_{k-1}}{r_k^i - \rho_i}. \tag{3.2.15}$$

The k^{th} step finishing time is calculated by taking the minimum of 3.2.15 over $i \in \{1, \ldots, N\} \setminus \bigcup_{j=1}^{k-1} L(j)$ i.e.

$$t_k = \min\{t_{i,k} \mid t_{i,k} > 0; \;\; i \in \{1, \ldots, N\} \setminus \bigcup_{j=1}^{k-1} L(j) \; \}. \tag{3.2.16}$$

However, to reduce the computation complexity of subsequent $t_{i,k}$'s, we found the following recursion:

$$t_{i,k} = \frac{S_{i,k}}{r_k^i - \rho_i} \tag{3.2.17}$$

where

$$S_{i,k} = \sigma_i + \sum_{j=1}^{k-1} r_j^i (t_{j-1} - t_j) + r_k^i t_{k-1}, \quad k = \{2, \ldots, N\}, \tag{3.2.18}$$

and

$$\begin{aligned} S_{i,k+1} &= S_{i,k} - r_k^i \, t_k + r_{k+1}^i \, t_k \\ &= S_{i,k} + (r_{k+1}^i - r_k^i) \, t_k. \end{aligned} \tag{3.2.19}$$

From (3.2.7) the initial value of $S_{i,1}$ follows, i.e.

$$S_{i,1} = \sigma_i. \tag{3.2.20}$$

□

The calculation of backlog clearing times is quite simple and can be highly automated using our proposed recursive formula. One only has to maintain the rolling sum of S_i^k's, then calculating r_k^i in each step using (3.1.12) and further taking the minimum of (3.2.17) to get the next step finishing time.

Figure 3.2 further shows the flowchart of the proposed algorithm.

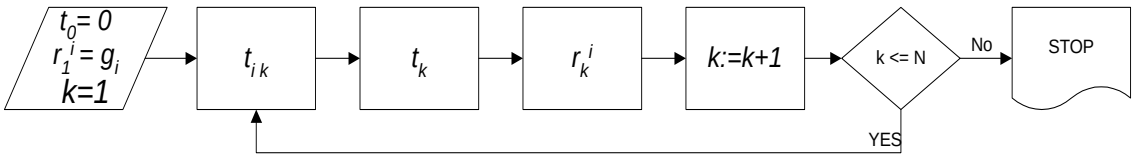


Figure 3.2: Flowchart of the iterative algorithm

Chapter 4

Deterministic Delay Bounds for Arbitrary Weighted GPS Servers

4.1 Arbitrary weighting

The delay bounds of GPS systems - or its packetized versions - have been widely analyzed under leaky bucket constrained input traffics. However, the simple delay bound introduced by Parekh et al. in [PG93a], i.e.,

$$D_i^* = \frac{\sigma_i}{g_i} \tag{4.1.1}$$

where g_i is the guaranteed service rate of session i only holds when $g_i \geq \rho_i$ for session i [PG93a, Zha95].

In order for us to investigate arbitrary weighting we define the above constrained subset of possible weight assignments as *locally stable*, where results of Parekh et al. holds.

Definition 4.1.1 (Locally Stable Weighting). A weight assignment of a GPS system is said to be locally stable if and only if the calculated guaranteed service rate is greater than or equal to the corresponding long term sustained rate, i.e.,

$$g_i \geq \rho_i. \tag{4.1.2}$$

This very constraint however does not allow to set the weights (ϕ) without restrictions,

thus limiting the usable parameter space. To overcome this limitation, we introduce a theorem on how the delay of session i can be calculated for any arbitrary sets of ϕ's. Further, we state that the delay bound we calculate is at least as good as that given by (4.1.1).

Theorem 4.1.1 (Delay Bound [C6, C7, C9]).
(i) If $r_i(t)$ as defined in (3.1.7) satisfies

$$r_i(\tau_i) \geq \rho_i \tag{4.1.3}$$

then

$$D_i = \tau_i \tag{4.1.4}$$

where τ_i is defined:

$$W_i(0, \tau_i) = \int_0^{\tau_i} r_i(t)\, \mathrm{d}t = \sigma_i. \tag{4.1.5}$$

(ii)If (4.1.3) does not hold for session $i = L(I)$ then it starts with an accumulating phase, for which there exists a $j \in \{1, \ldots, I-1\}$ that

$$\forall t < t_j : \; r_i(t) < \rho_i \tag{4.1.6}$$

and

$$\forall t \geq t_j : \; r_i(t) \geq \rho_i \tag{4.1.7}$$

then

$$D_i = t_j - \frac{W_i(0, t_j) - \sigma_i}{\rho_i} \tag{4.1.8}$$

where $W_i(0, t)$ is the amount of session i traffic served up till time t.

Proof. We want to calculate the maximal delay of session i, i.e.,

$$D_i = \max_t \{t - A_i^{-1}(W_i(t))\}, \tag{4.1.9}$$

where $A_i^{-1}(W)$ is the inverse of the arrival function. As we consider worst case maximal delay, we assume a greedy system, where the arrival function and its inverse are

$$A_i(t) = \sigma_i + \rho_i t \tag{4.1.10}$$

$$A_i^{-1} = t(A_i) = \max\left\{\frac{A_i - \sigma_i}{\rho_i},\ 0\right\}. \tag{4.1.11}$$

Now the delay of session i can be written as

$$D_i(t) = t - \max\left\{\frac{\int_0^t r_i(\tau)\mathrm{d}\tau - \sigma_i}{\rho_i},\ 0\right\}, \forall t \geq 0. \tag{4.1.12}$$

Since (4.1.12) is an increasing function for $t \in [0, \tau_i]$ where τ_i is defined by (4.1.5), the smallest t when $D_i(t)$ achieves its maximum is τ_i, so (4.1.12) can be rewritten as

$$D_i(t) = t - \frac{\int_0^t r_i(\tau)\mathrm{d}\tau - \sigma_i}{\rho_i}, \forall t \geq \tau_i. \tag{4.1.13}$$

The delay function ($D_i(t)$) is continuous and linear between breaking points determined by the backlog clearing times. Hence, it's maximum is achieved at a breaking point where its left and right hand side derivate change sign. Thus, taking the left and right hand side derivation of (4.1.13) with respect to t, one gets the maximum where the derivate turns from positive to negative. The left hand side derivate is

$$D_i'^{-}(t) = 1 - \frac{r_i^-(t)}{\rho_i} \tag{4.1.14}$$

and the right hand side is expressed similarly, thus the condition of sign changing can be rewritten as

$$r_i^-(T_i^*) < \rho_i, \quad r_i^+(T_i^*) > \rho_i. \tag{4.1.15}$$

Since $r_i(t)$ is an increasing function in t (see Corollary 3.1.2), there are two cases:

(a) If $r_i(\tau_i) \geq \rho_i$, then (4.1.13) is a decreasing function, so the maximal delay is achieved at time τ_i, which proves (i).

(b) If $r_i(\tau_i) < \rho_i$, then T_i^* satisfying (4.1.15) is greater than τ_i. However as $r_i(t)$ is a step function (see Theorem 3.1.1) we seek for a $t_{L(j)}$ finishing time that satisfies (4.1.6) and (4.1.7). Further this $t_{L(j)}$ will also maximize the delay, since the step function of $r_i(t)$ will cross ρ_i at $t_{L(j)}$, which proves (ii).

□

4.2 Locally stable weighting

The above theorem defines the worst case delay of a GPS system for any arbitrary weighting, however for the case of *locally stable weighting* the following property can be proved:

Theorem 4.2.1 (Locally Stable Weighting: Tighter Delay Bound [C6, C7, C9]). *In the case of locally stable weighting (see Definition 4.1.1), where results of Parekh et al. holds, the following relation can be derived*

$$D_i \leq D_i^* \tag{4.2.1}$$

where D_i is defined by Theorem 4.1.1 and D_i^* is defined by (4.1.1)*.*

Proof. Since D_i^* holds only if $g_i \geq \rho_i$, so from Theorem 4.1.1 we only have to consider case (i). From Corollary 3.1.2 it follows that

$$r_i(t) \geq g_i \ , \quad \forall t \in [0, \tau_i] \tag{4.2.2}$$

where τ_i is defined by (4.1.5), thus replacing $r_i(t)$ by g_i in (4.1.5) we get

$$\int_0^{\tau_i^*} g_i \mathrm{d}t = g_i \tau_i^* = \sigma_i. \tag{4.2.3}$$

Now using (4.2.2), (4.2.3) and Theorem 4.1.1, one can write

$$D_i = \tau_i \ \leq \ \tau_i^* = \frac{\sigma_i}{g_i} = D_i^* \tag{4.2.4}$$

that proves our Theorem. □

For illustration of arrival and service rates and worst case delays see Figure 4.1. The figure shows a four session scenario where the arrival and service curves of the last session to clear its backlog is shown. The vertical lag between the arrival and the service curve corresponds to the backlog function, while the horizontal lag defines the experienced delay. The figure nicely shows, how the backlog clears of other sessions influence the service rate of our session. This way the shown worst case delay calculated with our proposed algorithm is significantly tighter than the one given by the guaranteed rate calculation.

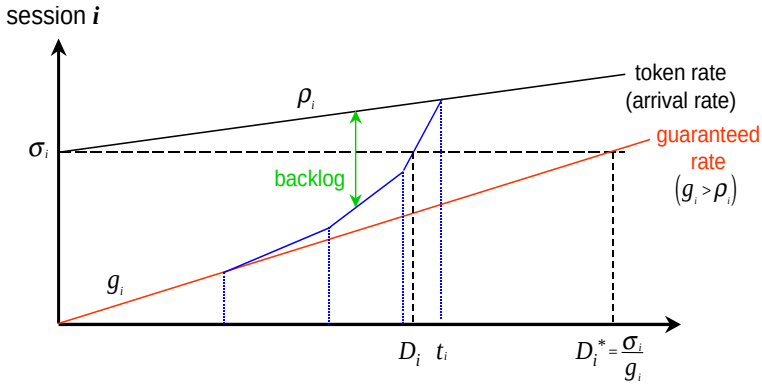


Figure 4.1: Service and arrival functions of a 4 session GPS system

4.3 Numerical examples

In this section we show some numerical examples on our approach to calculate delay and backlog bounds for a GPS node. For the sake of simplicity and without losing generality we assume in all examples that the service rate of the GPS node is 1.

4.3.1 Two sessions scenario

First consider a GPS node serving two sessions. Both start greedily. The parameters describing the sessions and the applied weights can be seen in Table 4.1. Note that the set ϕ^1 corresponds to the traditional case where weights are assigned proportional to long term sustained rates.

Session parameters	σ	ρ	ϕ^1	ϕ^2	ϕ^3	ϕ^4	ϕ^5	ϕ^6
Session 1 (red - dashed)	1	.2	2	6	1	12	1	100
Session 2 (blue - cont.)	3	.6	6	2	12	1	100	1

Table 4.1: Parameters for two sessions

Using our formula to calculate delays, one gets the result shown in Table 4.2. This gives the base of our comparison. It strikes out immediately that the achievable delays are

limited to certain intervals, i.e., $D_1 \in [1, 10]$ and $D_2 \in [3, 5]$[1]. Despite the correlation of the two delays, one can favor any of the sessions to come close to its theoretical minimum delay.

Delay bounds	ϕ^1	ϕ^2	ϕ^3	ϕ^4	ϕ^5	ϕ^6
D_1	4	4/3	10	13/12	10	101/100
D_2	4	5	13/4	5	303/100	5

Table 4.2: Delays for two sessions

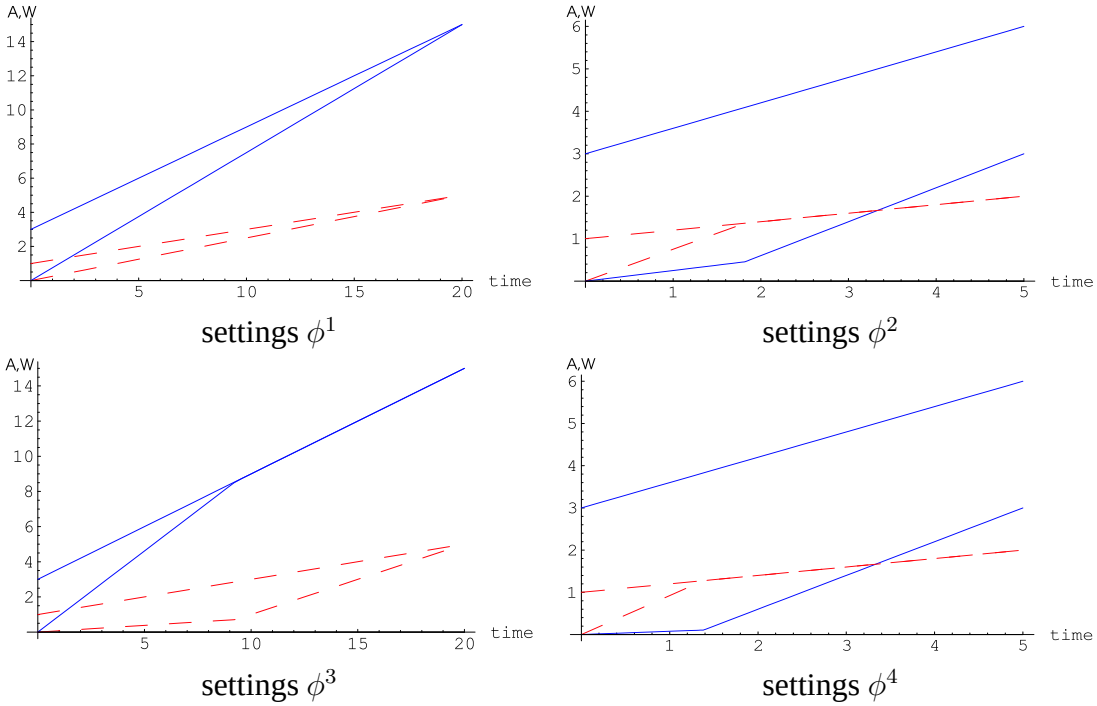


Figure 4.2: Arrival and service functions in time

To better understand the behavior of the GPS node, we present results on the arrival and service functions in Figure 4.2. Note that same sessions are drawn with same line styles and service functions start from zero while arrival functions start with bursts. Figure 4.2

[1]Note that the lowest delays can be achieved only by a degenerated system, where one weight is set to zero

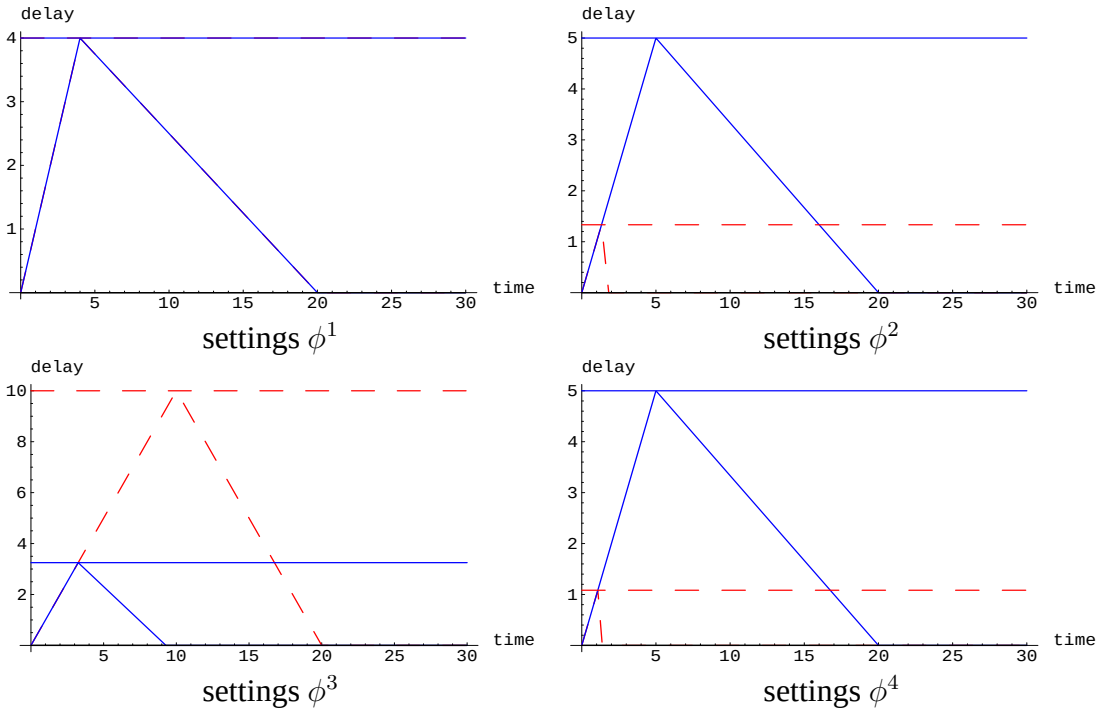


Figure 4.3: Maximal delays and delay functions in time

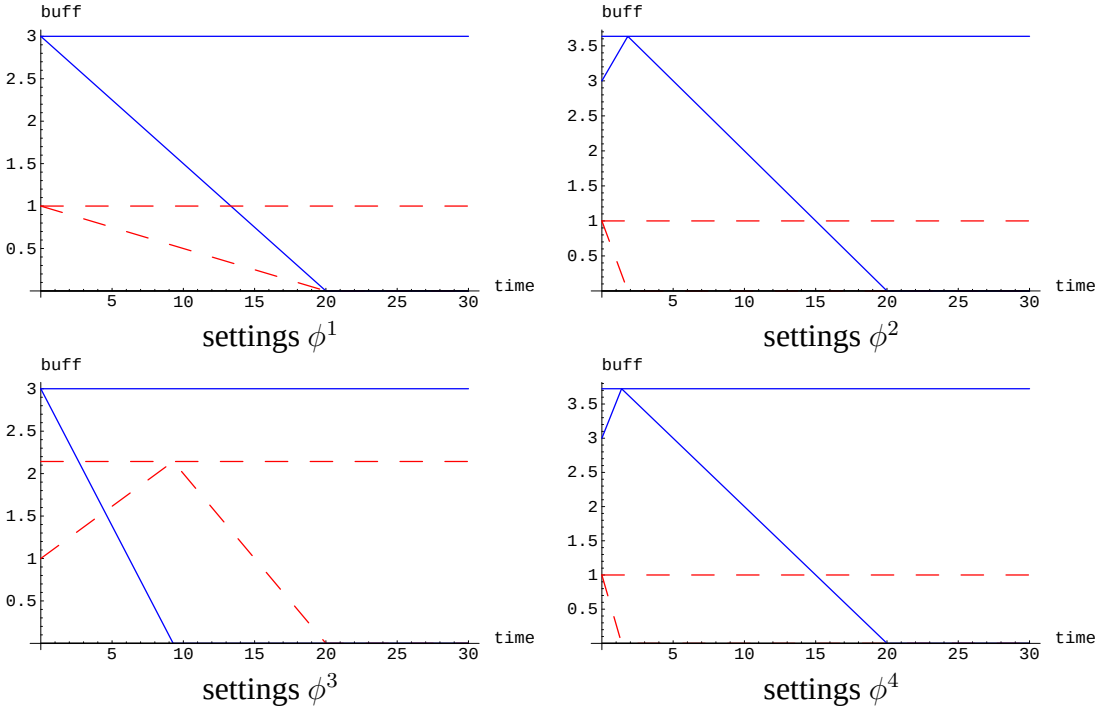


Figure 4.4: Maximal backlogs and backlog functions in time

nicely shows how service rates are increasing at backlog emptying times. The fact that one of the sessions sooner or later empties its backlog and further consumes only as much resources (bandwidth) as its sustained rate (ρ) allows the other session to clear its backlog faster, thus reducing its delay. The different weight settings determine whose backlog is emptied first.

Since our primary goal was to introduce tighter delay bound, let us also investigate the delay behavior of the sessions. Figure 4.3 shows the maximal and the experienced delays in the function of time. Again, same line style corresponds to one session's maximal delay and delay function in time. It is eye-catching that the delay curves for ϕ^1 coincide. However there may be situations when lower rate sessions have real-time delay requirements. In this case one has to de-couple bandwidth demands from the actual weights and set ϕ's according to the delay needs. For example, in setting ϕ^4 the small bandwidth session is favored against the high bandwidth one. In this case one may also consider that the increase in worst case delay for the high bandwidth traffic (from 4 to 5) may worth the significant decrease of the low rate traffic's delay (from 4 to 13/12). Further, one may carry out a system that is fair in a sense, that smoother traffic gets lower worst case delay performance than burst ones with indifference of their rates.

It is usually said that there are no gains without losses. The price one has to pay for playing with weights and setting them arbitrarily is the increase of necessary buffer capacity. Since the guaranteed service rates in the bandwidth-delay coupled system are greater than the corresponding arrival rates ($g_i \geq \rho_i$), session backlogs are determined by the burst sizes (see Figure 4.4 setting ϕ^1). On the other hand when setting weights de-coupled one session has to suffer higher delays (see Figure 4.3) hence requiring larger buffers (see Figure 4.4 settings: $\phi^2,\ \phi^3,\ \phi^4$).

4.3.2 A scenario where worst case delay is not achieved at σ

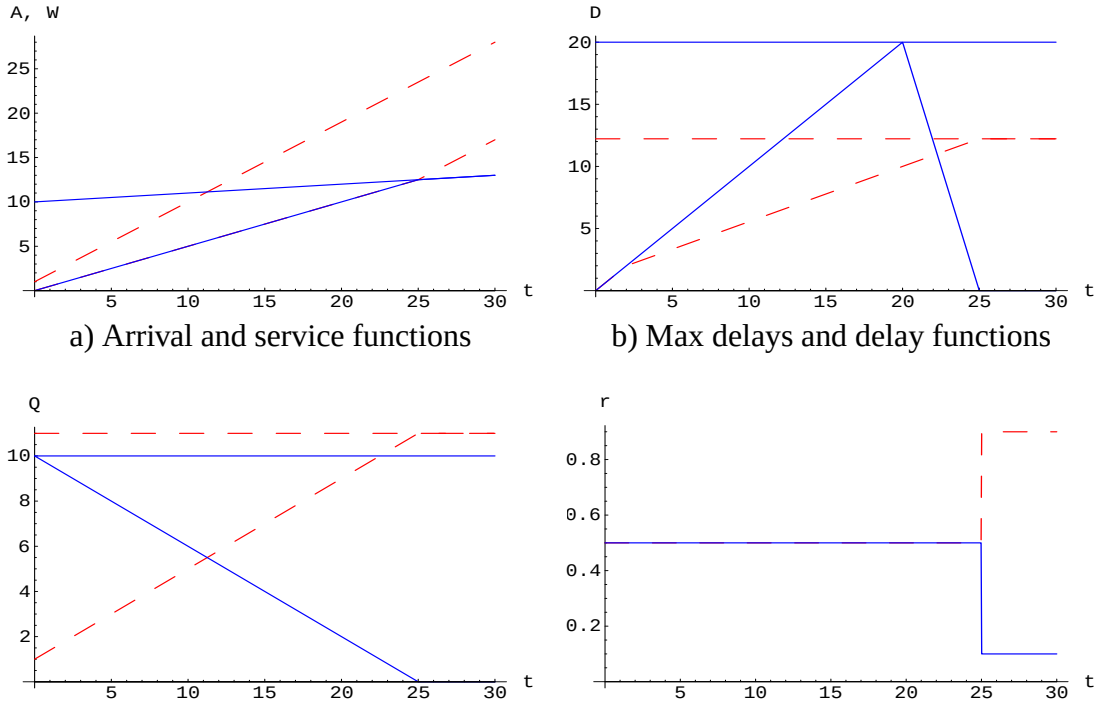
In this section, we illustrate a situation that corresponds to the (ii) part of Theorem 4.1.1, i.e., when the maximal delay of a session is not achieved at the time the service curve

meets the initial σ burst. We constructed a situation, where one of the sessions' service rate is lower at time τ - defined by (4.1.5) - than its arrival rate. The session descriptors and the weights are shown in Table 4.3.

Session parameters	σ	ρ	ϕ
Session 1 (red - dashed)	1	.9	1
Session 2 (blue - cont.)	10	.09999	1

Table 4.3: Session parameters when the worst case is not achieved at σ

Figure 4.5/a shows that the horizontal distance between the arrival and service functions spreads out up till the time session 2 clears its backlog. This way session 1 delay increases as shown in Figure 4.5/b. Figure 4.5/c outlines the backlog curves and the changing of rate functions is seen in Figure 4.5/d.



a) Arrival and service functions

b) Max delays and delay functions

c) Backlogs

d) Rate functions

Figure 4.5: Session characteristics when max delay is not achieved at σ

Another example when the maximum delay is achieved after burst flushing time, i.e., the time the accumulating phase turns into backlog emptying one, is later than the time when the corresponding σ burst is emptied. This scenario can not happen in the rate-proportional weighting case. Figure 4.6 shows the arrival functions and the service functions of a two session example ($\phi_1 = \phi_2 = 1$, $\sigma_1 = \sigma_2 = 1$, $\rho_1 = .25$ and $\rho_2 = .73$) where the maximal delay of session 2 is achieved when session 1 clears its backlog (t_1).

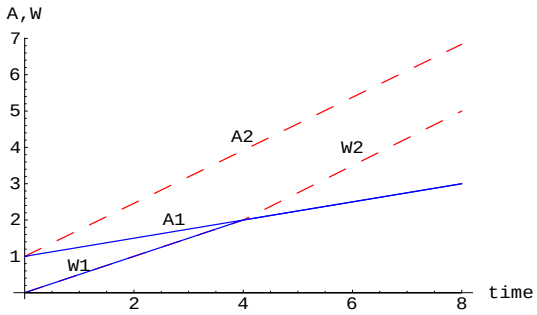


Figure 4.6: Arrival and service functions

The corresponding delay curves of Figure 4.6 are shown in Figure 4.7. This figure shows, that the delay of session 1 increases until its σ burst is emptied ($t = 2$) and decreases afterwards since its service rate is faster than its arrival rate (it only has backlog emptying phase). Further, as weights are set equally ($\phi_1 = \phi_2$) and burst sizes are the same, session 2 also gets service of its burst size up till $t = 2$, however from now on it only has to clear its incoming traffic with rate ρ_2 starting at time zero. The lag between the arrival rate and the service rate gives the new slope of the session 2 delay function. Since we are studying a case where $\rho_2 > g_2$ session 2 delay is still increasing after $t = 2$. Now session 2 achieves its maximal delay when its service rate passes its arrival rate, i.e., at the time when session 1 clears its backlog thus only requires service rate equal to its arrival rate. This is shown in both Figure 4.6 and 4.7 at $t = t_1 = 4$. One can see that the maximal delay of session 2 resulting from the non rate-proportional weighting is no longer bounded by (4.1.1).

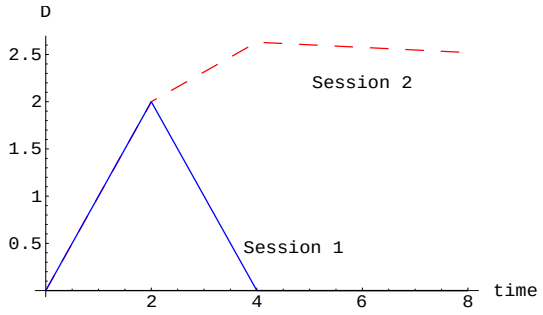


Figure 4.7: Queueing delays of leaving traffics

4.3.3 A 4-session scenario

Now turn our attention to the *rate-proportional weighting* once more. In Figure 4.8 we show the fourth session service curve of a 4 session scenario with rate-proportional weighting, where all σ's were 1 and $\{\rho_1, \rho_2, \rho_3, \rho_4\}$ were set to $\{.1, .03, .02, .01\}$ respectively. Figure 4.8 shows the four breaking points of session 4 service curve and its cross point with its σ burst line where it achieves its maximal delay. The figure also shows the guaranteed service curve whose cross point with the σ line gives the delay bound given by (4.1.1). Now it is clear that our worst case delay bound is *much* more tight than the one given in (4.1.1).

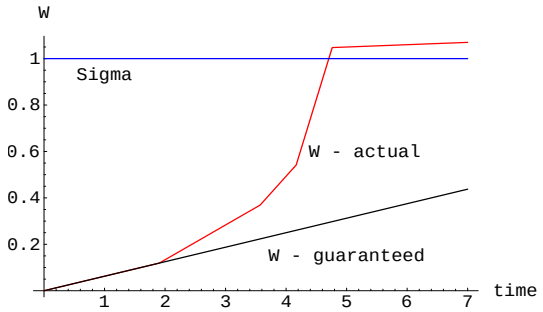
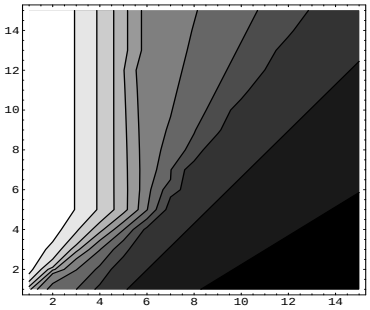


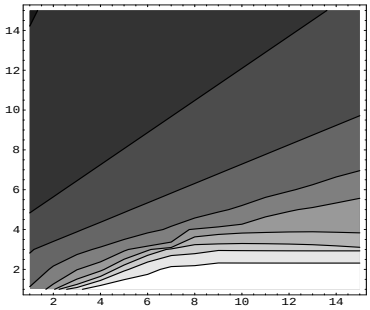
Figure 4.8: A service curve of a 4-session scenario

4.3.4 A 3-session scenario with 2 dimensional cuts

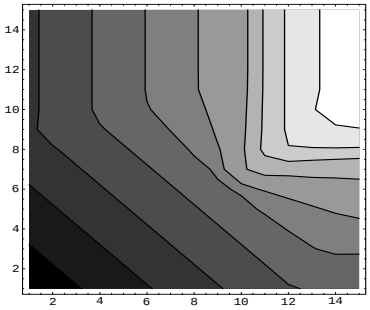
Our next scenario considers the case when three sessions share the server capacity. The corresponding traffic parameters and worst-case delay of each session can be seen in Figure 4.9. We study here the impact of weight changes of the first two sessions, while the third one is kept constant. Lighter areas denote higher delay. It is worth noting, that the delay of the third session is the smallest and unchanged in the area where the weight of other sessions relatively smaller (these are lines in the left bottom corner where the sum of the weights of the first two session is constant). Moving on one of the lines the delay of the third session remains constant, while one of the others decreases and the other one increases according to weight changes, see Figure 4.10. It implies that one can favor a delay sensitive session, while having the freedom in adjusting the other weights and thus the delay arbitrarily in that region. Note, that the third session's delay increases when both ϕ_1 and ϕ_2 increase because those sessions reserve more server capacity. We can observe another nice property if further fix one of the ϕ_1, ϕ_2. In this case the delay of the third session is unchanged again, independent from the one adjusted weight (see the horizontal and vertical lines in the right top corner of the third session delay picture). Notice, this property also follows from equation (3.2.11).



session 1



session 2



session 3

	σ	ρ
session 1	1	.4
session 2	1	.25
session 3	1	.25

parameters

Figure 4.9: Delays in the function of ϕ_1 (x-axis) and ϕ_2 (y-axis)

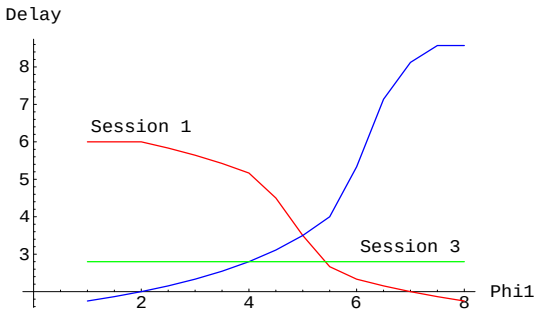


Figure 4.10: Delays when $\phi_1 + \phi_2$ is constant

Chapter 5

The Impact of Weight Changing on Delay Bounds

In this chapter to control session delays we introduce our approach to assign session weights and to make admission decisions. We consider a GPS node that serves a certain number of leaky bucket constrained sessions simultaneously. Further, by relaxing the rate-proportional weighting we lose the unambiguous assignment of ϕ_i's, i.e., we have to show a method for setting session weights to satisfy both the delay and bandwidth demands of sessions. However, the call admission control (CAC) for bandwidth is quite straightforward and is based on the preassured inequality that $\sum_{i=1}^{N} \rho_i < r(=1)$. On the other hand the appropriate setting of weights regarding the delay requirements is much more complex. In the following sections we first investigate the problem of ϕ assignment and introduce our theorem that CAC algorithms presented later in this section are built on.

5.1 Setting of session weights

In order to introduce the problem let us present a delay analysis of GPS systems. In the very simple case of three sessions with $\underline{\sigma} = \{1, 1, 0.9\}$, $\underline{\rho} = \{0.1, 0.1, 0.52\}$ and $\underline{\phi} = \{\phi_1, 10, 10\}$ Figure 5.1 shows how the worst case delay of sessions are changing if ϕ_1 is changing. One can make very interesting findings based purely on the simple results shown

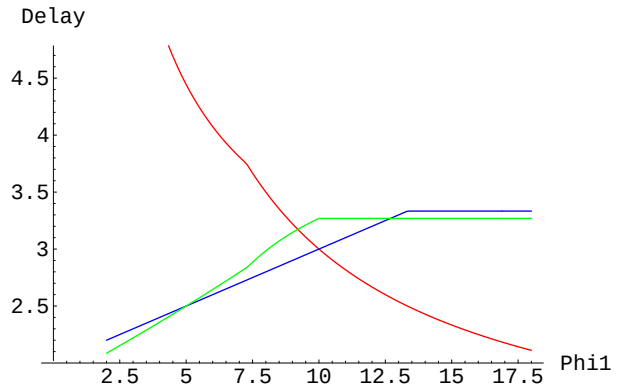


Figure 5.1: Delay of sessions for one varying ϕ

in the figure, namely:

- the worst case delay of the session, whose ϕ is changing is a monotone decreasing function while the others behave right the opposite way;
- even if the relation of ϕ's of the other two sessions are unchanged they cross each other twice;
- there are certain thresholds after which the worst case delays of the corresponding sessions are indifferent of the changing of ϕ;
- even in the region mentioned in previously, the delay of the session whose weight is increasing still decreasing, i.e., one can gain on a worst case delay bound without losing more at the others'.

Based purely on the former scenario it is clear, that the setting of weights is not straightforward at all, therefore we analyzed the delay behavior of GPS systems regarding weight changes. Through the following lemmas we introduce a theorem that states how a weight change affects the appropriate delay bound.

Since we are dealing with weights of GPS servers we deduced some very important attributes of weight changing. Further denote the altered system parameters by a $'$.

Our first lemma is based on the observation that the increment of session i's ϕ_i will result in an increment of the corresponding rate function r_k^i while the remaining rate functions of still backlogged sessions will slow down till session i backlog is cleared. In an exact way:

Lemma 5.1.1 ([C8, C6]). *Let's have a greedy GPS system with backlog clearing order of* $\{L(1), \ldots, L(N)\}$. *Now change* $\phi_i' = \phi_i + \delta$ *in a way that the backlog clearing order remains unchanged, i.e.,* $\{L(1), \ldots, L(N)\} = \{L'(1), \ldots, L'(N)\}$. *Then*

$$\exists I \in \{1, \ldots, N\} : i = L(I) = L'(I)$$

for which

$$\begin{aligned} r'^j_k &< r^j_k \quad \forall j \in \mathcal{B}(k) \setminus \{i\},\ k \in \{1, \ldots, I\} \\ r'^i_m &> r^i_m \quad \forall m \in \{1, \ldots, I\}, \end{aligned} \tag{5.1.2}$$

where $\mathcal{B}(k)$ *is the set of still backlogged sessions at step* k.

Proof. According to (3.1.7) in Theorem 3.1.1

$$\forall j \in \mathcal{B}(k) \setminus \{i\} \wedge k \in \{1, \ldots, I\} : \tag{5.1.3}$$

$$r'^j_k = \frac{(1 - \sum_{l=1}^{k-1} \rho_{L'(l)})\phi_j}{\sum_{i=1}^{N} \phi_i + \delta - \sum_{l=1}^{k-1} \phi_{L'(l)}} \tag{5.1.4}$$

and

$$r^j_k = \frac{(1 - \sum_{l=1}^{k-1} \rho_{L(l)})\phi_j}{\sum_{i=1}^{N} \phi_i - \sum_{l=1}^{k-1} \phi_{L(l)}}. \tag{5.1.5}$$

Since $L'(j) = L(j)$ from the assertion the only difference between (5.1.4) and (5.1.5) is the $\delta > 0$ in the denominator, that yields for $r'^j_k < r^j_k$. Further, as the server is work-conserving and the backlog clearing order is unchanged, i.e., already non-backlogged sessions consume the same amount of resources, the sum of session j service rates is equal in both system

$$\sum_{j \in \mathcal{B}(k) \setminus \{i\}} r'^j_k + r'^i_k = \sum_{j \in \mathcal{B}(k) \setminus \{i\}} r^j_k + r^i_k \tag{5.1.6}$$

thus by reordering

$$r'^{i}_{k} - r^{i}_{k} = \sum_{j \in \mathcal{B}(k) \setminus \{i\}} \{r^{j}_{k} - r'^{j}_{k}\}, \tag{5.1.7}$$

where each part of the right hand sum is greater than zero, which yields for $r'^{i}_{k} > r^{i}_{k}$. □

Since Lemma 5.1.1 only defines rate function relations at backlog clearing times we introduce the following lemma that puts the statement of Lemma 5.1.1 to any time instant within the system busy period.

Lemma 5.1.2 ([C8, C6]). *By taking the assertions of Lemma 5.1.1 and further assuming that* $t'_k > t_k \; \forall k \in \{1, \ldots, I-1\}$, *where* $i = L(I)$ *then*

$$\begin{aligned} r'^{j}(t) < r^{j}(t) \quad & \forall j \in \mathcal{B}(t) \setminus \{i\} \wedge t \in [0, t'_I) \\ r'^{i}(t) > r^{i}(t) \quad & \forall t \in [0, t'_I) \end{aligned} \tag{5.1.8}$$

where $r^{j}(t)$ *is the service rate of session* j *at time* t *as defined in Theorem 4.1.1.*

Note that this lemma states more than the previous one, since it defines the relation of the service rates at intermediate intervals of backlog clearing times.

Proof. The statement immediately follows from Lemma 5.1.1 if $t \in [0, t_1)$. Now, if $t \in [t_1, t'_I)$, then

$$\begin{gathered} \forall t \in [t_1, t'_I) \; \exists u, v : \\ u = \arg\max_u \{t_u | t_u < t\} \\ v = \arg\max_v \{t'_v | t'_v < t\}. \end{gathered} \tag{5.1.9}$$

In other words, u and v are the last backlog clearing events in the original and modified system respectively before time t. By the assertion of $t'_k > t_k$ it follows that $u \geq v$, which using Lemma 5.1.1 yields the proof. □

In order for us to go ahead towards our theorem, we have to prove that the assertion of Lemma 5.1.1 holds, i.e., if ϕ_i changes to $\phi'_i = \phi_i + \delta$ where $\delta > 0$ then session backlog clearing times are increasing till session i clears its backlog. Session i backlog clearing time on the other hand decreases.

Lemma 5.1.3 ([C8, C6]). *In a GPS system where ϕ_i changes to ${\phi'}_i = \phi_i + \delta$ where $\delta > 0$, and the backlog clearing order is intact, i.e., $\{L(1), \ldots, L(N)\} = \{L'(1), \ldots, L'(N)\}$, then*

$$\begin{aligned} \exists I : i = L(I) = L'(I) \\ {t'}_k \geq t_k \quad k \in \{1, \ldots, I-1\} \\ {t'}_I \leq t_I \text{ where } i = L(I). \end{aligned} \tag{5.1.10}$$

Proof. By assumption $\{L(1), \ldots, L(N)\} = \{L'(1), \ldots, L'(N)\}$. From Lemma 5.1.1 it follows[1] that the first session to clear its backlog starts with a modified rate lower than its original one, i.e., ${r'}_1^{L(1)} < r_1^{L(1)}$. Thus the time to clear its backlog (see Definition 3.1.2) ${t'}_1$ will increase. From Lemma 5.1.2 it follows up till ${t'}_1$ that

$$r'_j(t) < r_j(t) \quad \forall j \in \{1, \ldots, N\} \setminus \{i\} \wedge t \in [0, t'_1], \tag{5.1.11}$$

therefore as the service functions are

$$W'_j(t'_1) = \int_0^{t'_1} {r'}_j(\tau) \mathrm{d}\tau \tag{5.1.12}$$

and

$$W_j(t'_1) = \int_0^{t'_1} r_j(\tau) \mathrm{d}\tau \tag{5.1.13}$$

one gets that

$$W'_j(t'_1) < W_j(t'_1) \; \forall j \in \{1, \ldots, N\} \setminus \{i\}. \tag{5.1.14}$$

Now assume that for the k^{th} step the following holds:

$$\begin{aligned} t'_l \geq t_l \qquad & l \in \{1, \ldots, k\} \\ W'_j(t'_l) < W_j(t'_l) \quad & j \in \{L(k+1), \ldots, L(i)\} \wedge \\ & l \in \{1, \ldots, k\} \end{aligned} \tag{5.1.15}$$

Hence, if the $(k+1)^{\text{th}}$ step is proven we induce the lemma. Since from (5.1.15)

$$W'_{L(k+1)}(t'_k) < W_{L(k+1)}(t'_k) \tag{5.1.16}$$

[1] without using its assertions, that is under proof

and the time till the next backlog clear (further denote by $\Delta\zeta$) is formulated

$$\Delta\zeta_{k+1} = \frac{A_{L(k+1)}(t'_k) - W_{L(k+1)}(t'_k)}{\rho_{L(k+1)} - r_{k+1}^{L(k+1)}} \tag{5.1.17}$$

and

$$\Delta\zeta'_{k+1} = \frac{A_{L(k+1)}(t'_k) - W'_{L(k+1)}(t'_k)}{\rho_{L(k+1)} - r'^{L(k+1)}_{k+1}}. \tag{5.1.18}$$

Using (5.1.16) and Lemma 5.1.1 one gets

$$\Delta\zeta'_{k+1} > \Delta\zeta_{k+1} \tag{5.1.19}$$

thus

$$t'_{k+1} \geq t_{k+1}. \tag{5.1.20}$$

Moreover,

$$W'_j(t'_{k+1}) = \int_0^{t'_{k+1}} r'_j(\tau)\mathrm{d}\tau \tag{5.1.21}$$

and

$$W_j(t'_{k+1}) = \int_0^{t'_{k+1}} r_j(\tau)\mathrm{d}\tau. \tag{5.1.22}$$

By subtracting (5.1.21) from (5.1.22) and using Lemma 5.1.2 it yields

$$W'_j(t'_{k+1}) < W_j(t'_{k+1}) \;\; j \in \{L(k+2), \ldots, L(i)\}. \tag{5.1.23}$$

This proves that all the assertions and the inequality holds at the $(k+1)$th step, thus the lemma is induced. □

In the following we will relax the session backlog order intact assumption to get our generally applicable result.

Lemma 5.1.4. *If the session backlog clearing order changes due to a session weight change in a GPS system, then all the results of Lemma 5.1.1 - Lemma 5.1.3 still hold.*

Proof. Assume that the session backlog clearing times for sessions j and i are t_J and t_I, respectively. Further assume that

$$t_J = t_I, \tag{5.1.24}$$

i.e., the backlog clearing times of session j and i coincide. Now, session i weight either increases or decreases. Separate the former two scenarios as follows:

i) $\phi_i' = \phi_i - \delta$, where $\delta > 0$. Now define the backlog clearing time order as

$$\{L(1), \ldots, L(J), L(I), \ldots, L(N)\}, \tag{5.1.25}$$

which coincide with the altered system backlog clearing order using Lemma 5.1.3 for sufficiently small δs, i.e.,

$$\{L(1), \ldots, L(J), L(I), \ldots, L(N)\} = \{L'(1), \ldots, L'(J), L'(I), \ldots, L'(N)\}. \tag{5.1.26}$$

Note, that since $t_J = t_I$ the backlog clearing order could also be defined as

$$\{L(1), \ldots, L(I), L(J), \ldots, L(N)\}. \tag{5.1.27}$$

We also know from Lemma 5.1.3 that

$$t_J' < t_J \tag{5.1.28}$$

$$t_I' > t_I. \tag{5.1.29}$$

ii) $\phi_i' = \phi_i + \delta$, where $\delta > 0$. Now, however, change the backlog clearing order of sessions to

$$\{L(1), \ldots, L(I), L(J), \ldots, L(N)\}. \tag{5.1.30}$$

Results of Lemma 5.1.3 can similarly derived for this scenario.

From the above separation we know that for all $\delta > 0$

$$t_J(\phi_i + \delta) < t_I < t_I(\phi_i - \delta) \tag{5.1.31}$$

$$t_I(\phi_i + \delta) > t_J > t_J(\phi_i - \delta), \tag{5.1.32}$$

where the backlog clearing order is defined as

$$\{L(1), \ldots, L(J), L(I), \ldots, L(N)\}. \tag{5.1.33}$$

Now, if $\delta > 0$ and $\delta \to 0$ then both the left hand side and the right hand side go to the same value t_I and t_J respectively, thus allowing a continuous smooth transition of finishing times. This way Lemma 5.1.3 holds regardless the backlog clearing order is intact or not.

In the case if more than two session finishing times coincide, then one can similarly determine the backlog clearing orders for any sufficiently small $\delta > 0$. With the separation of $\phi_i + \delta$ and $\phi_i - \delta$ it can also be shown that the $t_{L^{-1}(i)}(\phi_i)$ function is continuous in ϕ_i and satisfies the statements of Lemma 5.1.3 even for backlog clearing time changes. □

Lemma 5.1.5 ([C8, C6]). *The service function of session i is a monotone* increasing *function of the session's weight. In other words, for any time t*

$$\frac{\Delta W_i(t, \underline{\phi}, \underline{\sigma}, \underline{\rho})}{\Delta \phi_i} \geq 0 \tag{5.1.34}$$

Proof. To prove this lemma let's consider the system of Lemma 5.1.2 and Lemma 5.1.3. Now if $t \in [0, t'_I)$ then

$$W'_i(t) = \int_0^t r'_i(\tau) \mathrm{d}\tau \tag{5.1.35}$$

and

$$W_i(t) = \int_0^t r_i(\tau) \mathrm{d}\tau. \tag{5.1.36}$$

By subtracting (5.1.36) from (5.1.35) and using Lemma 5.1.2 it yields the result.

If $t \in [t'_I, \infty]$, then the service function is equal to the arrival function, which also bounds the service function of the original system. Therefore

$$W'_i(t) = A_i(t) \geq W_i(t) \quad \forall t \in [t'_I, \infty). \tag{5.1.37}$$

Hence the lemma is proved for $\forall t \in [0, \infty)$. □

Corollary 5.1.6 ([C8, C6]). *From Lemma 5.1.5 it follows that $t'_i \leq t_i$ if $\phi'_i = \phi_i + \delta$ where $\delta > 0$.*

Proof. The corollary is proved by contradiction. Assume that $t_i < t'_i$, then from Lemma 5.1.5 and the fact that the arrival function bounds the service function

$$A_i(t_i) = W_i(t_i) \ \leq W'_i(t_i) \leq A_i(t_i). \tag{5.1.38}$$

Since the left and the right hand is equal, thus

$$W_i(t_i) \ = W'_i(t_i) = A_i(t_i) \tag{5.1.39}$$

which contradicts the definition of t'_i, i.e.,

$$t'_i = \min\{t | W'_i(t) = A_i(t)\} \tag{5.1.40}$$

since $t_i < t'_i$ also satisfies the above equation. □

Now we have enough lemmas to proceed with our main theorem later used by the proposed CAC algorithms.

Theorem 5.1.7 ([C8, C6]). *The worst case delay of session i is a monotone decreasing function of the corresponding weight. In other words,*

$$\frac{\Delta D_i}{\Delta \phi_i} \leq 0. \tag{5.1.41}$$

Proof. From Theorem 4.1.1 it follows that the worst case delay is either (i) experienced when $W_i(t) = \sigma_i$ or (ii) at a backlog clearing time when session i turns its accumulating phase to backlog clearing one (4.1.8).

In case of (i), using the Lemma 5.1.5 and the definitions (4.1.5) and (4.1.4) will yield for the result.

If the GPS system is stable, i.e., $\sum_{i=1}^{N} \rho_i < 1 (= r)$ then $\forall t_a > 0$

$$\begin{aligned} &\exists\, t_d(t_a), t'_d(t_a) \geq t_a : \\ &A_i(t_a) = W_i(t_d) = W'_i(t'_d). \end{aligned} \tag{5.1.42}$$

What Theorem 4.1.1 says in other words is that

$$\begin{aligned} D_i &= \max_{t_a}\{t_d(t_a) - t_a\} \\ D'_i &= \max_{t_a}\{t'_d(t_a) - t_a\}, \end{aligned} \tag{5.1.43}$$

i.e., the maximum horizontal difference between the arrival and the service function gives the worst case delay. In a similar way as it was proven in Lemma 5.1.5 from $W'_i(t) \geq W_i(t)$ it follows that

$$t_d(t_a) = t'_d(t_a) + \xi \;\text{ where } \xi \geq 0 \tag{5.1.44}$$

hence

$$D_i = \max_{t_a}\{t'_d(t_a) + \xi - t_a\} = \max_{t_a}\{\hat{D'}_i(t_a) + \xi\} \tag{5.1.45}$$

which yields for the result. □

All this efforts were made to control the worst-case delays of sessions. From Theorem 5.1.7 it is now known that to decrease session i worst case delay one has to increase its appropriate weight. Unfortunately the side effects are quite complex in case of the modification of more than one weight. Nevertheless this assumption is further used in our CAC approaches.

Chapter 6

Call Admission Control in GPS schedulers

6.1 Different approaches for the CAC funtion

In an ideal system the CAC function only says whether the system is able to satisfy all sessions demands or not. However in reality one can not be satisfied by the fact that there exists solution for the problem, it is also needed to support a possible realization. Thus we further on tie the problem of CAC with finding a weight assignment $\underline{\phi}$, if exists, for any given $(\underline{\sigma}, \underline{\rho}, \underline{D})$ initial vectors that $\underline{D} \geq \underline{d} = \text{GPS}(\underline{\sigma}, \underline{\rho}, \underline{\phi})$ where $\underline{D}, \underline{d}$ are the delay requirement and the actual delay vector of the GPS system respectively, and $(\underline{\sigma}, \underline{\rho})$ are the leaky bucket descriptors of sessions[**C8**, **C6**].

In the following subsections we provide three methods to handle the CAC problem. The first is the random search algorithm, which operates purely on the vector spaces while the other two use the result of Theorem 5.1.7.

6.1.1 Random search

This algorithm[**C8**, **C6**] was created to verify all those algorithms (introduced later) that take some assumptions on the system. The method we used to find a solution was based on

the genetic algorithm approach. We constructed a population of a size determined by the number of sessions. Then by each step we so to say mutated the elements of the population and also added some limited inheritance by identifying those sessions whose requirements were fulfilled. After each step we limited the population to the initial fix number (N) by dropping all settings except the top N regarding an objective function by which the goodness of a setting was described. We have used objective functions similar to the ones described in the following section for the Gradient method.

Evidently, if no solution is found we can not say anything about the feasibility of the system parameters, however the step count of successful searches can provide a reference for the other algorithms.

6.1.2 The gradient method

Our CAC algorithm here is based on an N dimensional vector-scalar function used as the objective of minimization. The objective function is constructed in a way that a delay bound exceed is punished with a penalty function for each session. The gradient method[**C8**, **C6**] is based on the findings of Theorem 5.1.7, since it gives an unambiguous answer for how to change a session weight to improve its delay bound. In our initial effort we considered the objective function of (further referred as ***flat***)

$$O_1(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) = \sum_{i=1}^{N} I_{(d_i - D_i)}(d_i - D_i)^2 \tag{6.1.1}$$

where $I_{(x)}$ is an indicator function defined by

$$\begin{aligned} I_{(x)} &= 0 \quad x < 0 \\ I_{(x)} &= 1 \quad x \geq 0. \end{aligned} \tag{6.1.2}$$

$O_1(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi})$ was minimized with a gradient search in the $\underline{\phi}$ space. However we find that because of the one side flattened penalty function, if a session gets much better service than required then it took too much time to move the $\underline{\phi}$ from its place. It was even worse if sessions outside their delay bounds were in their saturated region, where their delay is

almost indifferent to $\underline{\phi}$ changes. Further, as the objective function is flat in the feasible region, a random solution is picked even if a more fair one could be found. What we mean here is that it may happen that a few sessions' delay bounds are overly satisfied while some are at their tolerance level. This is against the approach of GPS, since excess resources are assumed to be shared according to the assigned weights. Now, since we dimension our system for delays, we defined the *delay-fairness (d-f)* of a weight assignment in the following way:

Definition 6.1.1 (delay-fairness[C8, C6]). The delay fairness (Γ) of a GPS system for a given $\{\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}\}$ is defined by the sum of quadratic variance of delays. In a more formal way

$$\Gamma(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) = \frac{1}{N}\sum_{i=1}^{N}(\frac{d_i}{D_i} - \frac{1}{N}\sum_{j=1}^{N}\frac{d_j}{D_j})^2. \tag{6.1.3}$$

To take into account the delay-fairness property, we constructed the following objective function:

$$O_2(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) = \Gamma(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) + O_1(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}). \tag{6.1.4}$$

Optimizing O_2 in the $\underline{\phi}$ space will yield, if exists, for a solution $(\underline{\phi^*})$ where the proportional improvements in each worst case delay is balanced or in other words

$$\frac{d_i^*}{D_i} = \frac{d_j^*}{D_j} \quad \forall i, j \in \{1, \ldots, N\}. \tag{6.1.5}$$

This not only provides a delay-fair solution, but also shows how much session delays can further be improved still maintaining feasibility. It can be said, that the GPS system defined by $\{\underline{\sigma}, \underline{\rho}, \underline{d^*}, \underline{\phi}\}$ is at its efficiency bound, i.e., if any of session demands was tighter then there would be no feasible solution to satisfy all session needs.

6.1.3 The incremental approach

The above two methods have the drawback that they must handle the whole GPS system as one unit, however CAC is based on reservation requests, when a new session wants to join the system. Quite evidently the random search approach could not utilize the fact that

$N-1$ sessions were successfully allocated in the system, so only the newcomer has to be accommodated in the system somehow. Note that it usually requires modification of session weights already in the system. The gradient method also bears the same shortcoming, since the choosing of initial ϕ_N for the newcomer can not be defined, so starting randomly the optimization process must be undertaken. It certainly affects the required computation complexity for a CAC decision.

An obvious engineering approach could be to take the requests one by one as they arrive and allocate weights in the following way[**C8**, **C6**]:

Define a *dummy* session considering as if it was always in the system with the following parameters: $\{\sigma_d = \infty, \rho_d = 0\}$. Without loosing generality let us use a normalized ϕ vector in order to avoid identical solutions, since only the relations of ϕ's are important. Further assume that our goal is to give as much service to the dummy session as possible while still maintaining the delay requirements of remaining sessions. Now if a session enters the system at the ith step it can allocate weight from the dummy session up to the amount the dummy disposes of. It is practical to take all the weight the dummy has. If session i delay requirement is satisfied by $\phi_i = \phi_d$ then the CAC can immediately returns a positive acknowledgement. On the other hand, if ϕ_d is not enough for session i, there is still chance to accommodate the newcomer. It is because the dummy session probably consumes more resource[1] than the newcomer requires, thus the dummy replaced with the newcomer can improve the performance of sessions already in the system. Thus, by re-optimizing these weights one may gain enough to accommodate the newcomer. If the server still fails to satisfy session i demands, it rejects the request. In the case of positive acknowledgement the system must be re-optimized in a way that the dummy session is inserted again at the $(i+1)$th place. Note that this step can be done during idle periods, while the CAC decision in most cases will require only one step unless the system is at its efficiency bound.

Re-optimization is based on Theorem 5.1.7, i.e., if session j worst case delay is less than allowed ($d_j < D_j$) then ϕ_j can be decreased. This step is done iteratively, since the changing of one ϕ affects not only the corresponding delay but also the others. If no

[1] note that $\sigma_d = \infty$

session can be modified during an iteration step, we consider the system to be optimized. The flowchart of the algorithm is shown in Figure 6.1.

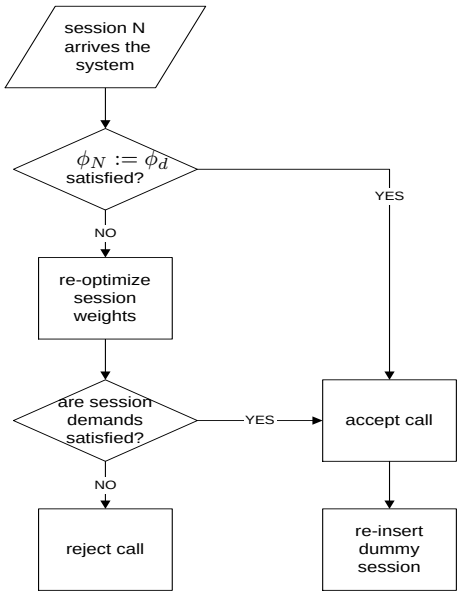


Figure 6.1: Flowchart of the incremental CAC algorithm

Further, this approach is said to be best-effort aware, since the weight of the dummy session can be assigned to the best-effort service class without putting at risks the guarantees of QoS sessions. This is because the dummy session is considered to be endless thus any best-effort stream will consume less resource than the dummy. Moreover, the normalized weight of the dummy session can also be considered as a load metric, since it expresses the amount of free resources that can be allocated to any other/new sessions unrestricted.

6.2 Numerical Results

This section presents numerical results of a 5 session GPS system. The input parameters for the CAC problem are shown in Table 6.1. Two extreme cases have been examined, namely (i) in the first case (referred as loose system) the delay requirements of sessions

(denoted by D_{loose}) can be satisfied easily, i.e., all five admitted sessions do not consume all system resources, while in (ii) referred as tight system, the delay requirements (denoted by D_{tight}) are so tight that the system bandwidth practically is fully utilized. Note that the delays of RPS weighting are also shown as a reference ($D_{RPSweighting}$).

Figure 6.2 reveals the 2-dimensional cuts of the objective functions of the loose system using different penalty functions. In these figures the feasible region in the ϕ_1, ϕ_2 space can be seen. At first there seems to be no difference between the results of the two objective functions O_1 and O_2 (Figure 6.2a and Figure 6.2c), but focusing on the edge of the feasible region (Figure 6.2b and Figure 6.2d) we find that the use of delay-fairness function results in an objective function which has an unique minimum. This ensures that the gradient search not only will find a solution (if exists) but it will provide the optimal solution where the relative delays ($\frac{d_i^*}{D_i}$) are balanced. Moreover, despite the fact that we have a loose system, in this way the server bandwidth is not wasted and the fairness property of GPS is maintained.

The objective functions of the tight system where the parameters were taken as d_i^*'s of the delay-fair system, is shown in Figure 6.3. In this case the server has no excess bandwidth and this is why the objective functions are very similar in the two cases O_1 and O_2 (compare Figure 6.3b and Figure 6.3d). It means that the acceptable weight region is so small that the optimization procedure must find the same solution even in case of different penalty functions.

Comparing the various CAC methods one can observe that in the loose system different searching algorithms results in different feasible weights (Table 6.2). This is because the feasible region is quite large and it permits the existence of different acceptable weight assignments. Note that the delay-fairness measure is close to zero in case of minimizing the O_2 function meaning that using the delay-fairness penalty function one can get an optimal solution in terms of fairness.

The incremental approach allows us to investigate the system reserve denoted by BE, i.e., the unused fraction of resources remaining after satisfying the admitted sessions' delay requirements. The other three approaches do not have this property because they only

Table 6.1: The Input Parameters of a 5 Session GPS System

	s_1	s_2	s_3	s_4	s_5
σ	1	2	3	2	1
ρ	0.2	0.25	0.2	0.15	0.05
D_{loose}	14	12	20	25	14
D_{tight}	8	6.8	11.4	14.2	8
$D_{RPSweighting}$	5.88	9.41	17.64	15.68	23.52

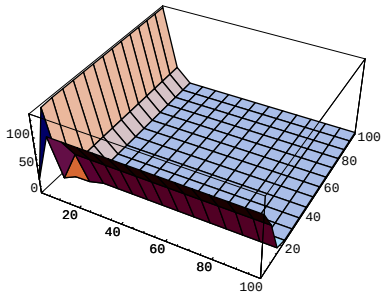
give a feasible solution, while in the fourth case the *dummy* session involves the unused bandwidth.

By definition the BE measure in the tight system is close to zero (Table 6.3). As we have a small feasible region, all approaches provide practically the same results. Also note, that the fairness measure has an acceptable range in all cases. The random search algorithm was not able to find solution, since the feasible region were so tight that without deeper insight of the system the probability of finding a good solution was small.

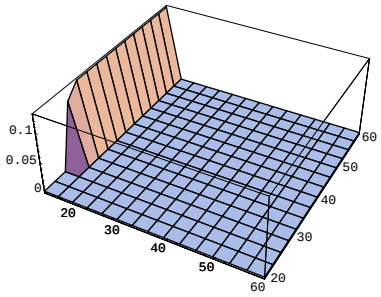
By concluding the former experiment, it is the gradient approach with delay-fairness metric that one should select if no additional traffic is imposed to the system. This is because it equalizes the improvements in delay among all competing sessions. On the other hand, if best-effort services are also handled by the system it is best to reserve only the necessary bandwidth required to satisfy connection demands and let the surplus resources used by the best-effort class. This way, even a minimum throughput can be guaranteed to the b-e class without degrading the qos of higher priority traffics.

6.2.1 A Voice/Video Scenario

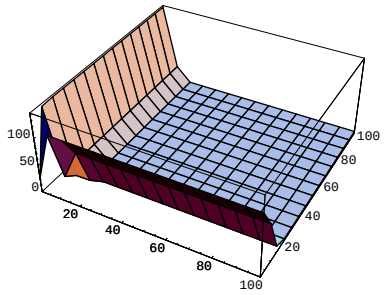
Here we have investigated a scenario where several voice and video calls were competing for the shared resource of a single GPS node. Both video and audio (Voice over IP call) trace files were analysed for the respective traffic descriptors. We compared the performance of the traditional rate-proportional weighting and our bandwidth-delay decoupled system. As in a realistic system we assumed quite tight performance requirements for the



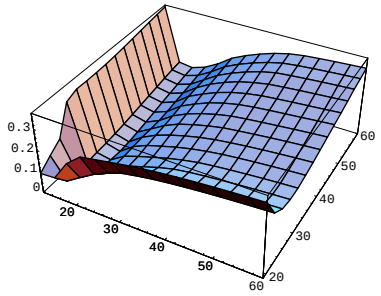
$a)$, O_1 Objective function



$b)$, O_1, zoomed into the edge of the feasible region

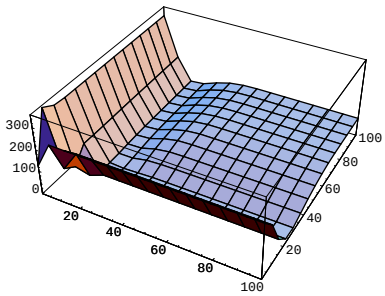


$c)$, O_2 Objective function

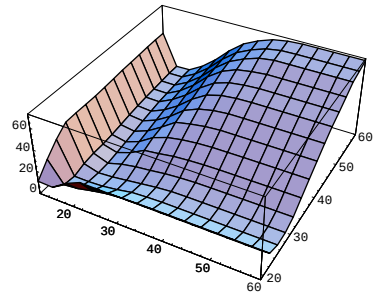


$d)$, O_2, zoomed into the edge of the feasible region

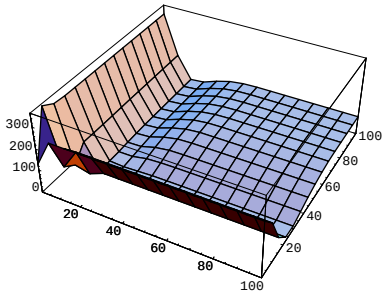
Figure 6.2: 2 dimensional cuts of the objective function of a 5-session *loose* system



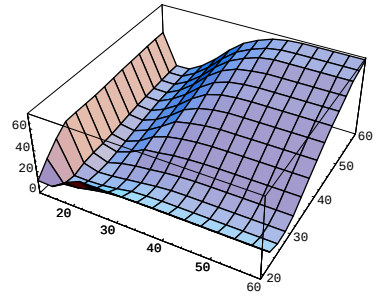
$a)$, O_1 Objective function



$b)$, O_1, zoomed into the edge of the feasible region



$c)$, O_2 Objective function



$d)$, O_2, zoomed into the edge of the feasible region

Figure 6.3: 2 dimensional cuts of the objective function of a 5-session *tight* system

Table 6.2: Session Weights & Fairness in the *Loose* System

	random	grad(flat)	grad(d-f)	incremental
ϕ_1	15.1	12.2	17.3	23.0
ϕ_2	25.8	23.3	29.5	28.6
ϕ_3	20.8	14.0	26.4	22.9
ϕ_4	12.7	33.3	14.1	17.2
ϕ_5	25.6	17.2	12.7	8.3
Γ	0.0145	0.0474	1.95e-06	0.0329
BE				14.7%

Table 6.3: Session Weights & Fairness in the *Tight* System

	grad(flat)	grad(d-f)	incremental
ϕ_1	17.5	17.3	17.4
ϕ_2	29.5	29.6	29.5
ϕ_3	26.4	26.5	26.4
ϕ_4	14.0	14.1	14.1
ϕ_5	12.6	12.6	12.6
Γ	2.430e-04	1.17e-07	5.22e-05
BE			0.4%

low bandwidth VoIP calls like 10 msec per one hop, i.e., a GPS node in our case. The calculated leaky bucket descriptors of the video trace files found in the [mpe] are shown in Table 6.4. As traffic inputs we also used measurements of video conference calls used by Cornell CU-SeeMe software. Here, different codings and quality levels were adjusted to get different traffic characteristics (coding: CU-SeeMe Gray, WhitePine Color, WhitePine M-JPEG Code) (see Table 6.5). VoIP descriptors are shown in Table 6.6.

A 2-session scenario

Here we investigated a low bandwidth link imposed with a VoIP and a video conference call. The idea here is that with rate-proportional weighting (RPS) one has to increase the allocated rate for the voice session to meet its very tight delay requirement. This, however

Table 6.4: Leaky Bucket descriptors of MPEG videos

	σ (Mb)	ρ (Mbps)
Mr. Bean	44.58	.424
Asterix	55.31	.536
ATP	23.31	.525
Goldfinger	67.82	.583
Jurassic Park	37.89	.314
Silence of the Lambs	29.31	.175
Movie Review	24.90	.343

Table 6.5: Leaky Bucket descriptor of CU-SeeMe video conference software

	σ (kb)	ρ (kbps)
CU-SeeMe Gray	109.6	272.2
WhitePine Color 1	16	194
2	184	889
WhitePine M-JPEG Codec 1	200	168
2	325	296

consumes too much resource that the high bandwidth call can not be satisfied nevertheless its quite loose demand in delay. On the other hand, using our proposed bandwidth-delay decoupled (B-D) system, where the tighter delay bounds are used for CAC, both session demands are met (see Table 6.7). This is because after the backlog clearing time of the voice session its bandwidth consumption is as low as its long term sustained rate (9.58 kbps), which is much lower than the allocated rate needed to draw out its initial burst (144 kbps). Rate-proportional service does not take into account the fact that the service rate of the voice session slows down, thus incapable of accommodating both sessions.

Table 6.6: Leaky Bucket descriptor of VoIP calls

	σ (kb)	ρ (kbps)
VoIP call	1.450	9.58

Table 6.7: A 2-session scenario

	σ [kb]	ρ [kbps]	Delays [sec]	RPS	B-D
VoIP call	1.450	9.58	.01	✓	✓
Conference call	325	296	1	-	✓

A realistic scenario

In this experiment we investigated a more realistic scenario corresponding to an ATM link of 25 Mbps server capacity, where three session types were competing for the shared resource (voice (v), mpeg video (m) and video conference (c)). 25 sessions were trying to access the link simultaneously with random shares of $2/5$ voice, $2/5$ video conference and $1/5$ mpeg video. The CAC results are shown in Table 6.8 where *session pattern* means the arrival order of different sessions, which are considered for CAC one by one. *(m, v, c ($\sum$))* are the number of accepted flows from the mpeg, voice, video conference and total number of calls respectively. It can be seen from the results that our proposed CAC outperforms the traditional rate-proportional weighting in all cases, though its efficiency depends on the actual arrival pattern.

Throughout these representative patterns it is obvious that our approach has reasons to investigate. More insight analysis of realistic scenarios like the one presented herein will be needed.

Table 6.8: Some 25-session scenarios with 25 Mbps link

Session pattern	RPS (m, v, c ($\sum$))	B-D (m, v, c ($\sum$))
cmcmccvvmccccccvcvcvvmvmcv	2, 2, 7 (**11**)	3, 8, 12 (**23**)
vcvmccccccccmvvcvccmmcvvc	3, 5, 11 (**19**)	4, 7, 13 (**24**)
vmvmvvvvvvvvmcccvvvcvcmcv	3, 15, 5 (**23**)	4, 15, 5 (**24**)
ccccvvvvmcmmmvvccccmcvccv	2, 6, 5 (**13**)	3, 8, 12 (**23**)
cmmcmcccvccccvccvcccccccvm	3, 2, 10 (**15**)	4, 4, 16 (**24**)
ccccmcccvvvcmvvvvcvcvmmcv	2, 10, 11 (**23**)	3, 10, 11 (**24**)
vvcvvvccmcvcvvcvccmmccvvv	2, 12, 9 (**23**)	3, 12, 9 (**24**)
vmcvcccmmvvmvvccvvcvmvmcv	4, 6, 6 (**16**)	4, 11, 8 (**23**)

6.3 Computation Complexity of Algorithms

The calculation of delay bounds of an N sessions GPS system consists of N steps. In the ith step the system state is computed at t_i using the state at t_{i-1}. There are some $o(N)$ complex tasks like calculating the state variable r_i^j for $\forall j$. Using these variables, the session which clears its backlog in the ith step has to be found. We do this in $o(N)$ time: computing state variables and choosing the smallest one. (With more complex data structures the algorithm might be improved.) So the complexity of the ith step is $o(N)$, therefore the complexity of a delay calculation for a given $\underline{\sigma}, \underline{\rho}, \underline{\phi}$ in the GPS system is $o(N^2)$.

In the random and the gradient methods one step is built up from proportional number of GPS simulation to N. In the former algorithm there is no guarantee for finding a solution, in the later one the speed of the optimization might be too slow so we bounded the iteration steps for both algorithms. If any of the algorithms exceeds this bound it stops with CAC refusal. So the overall complexity is $o(N)$ GPS calls, that is $o(N^3)$ elementary steps[**C8**, **C6**].

In case of the incremental approach one has two possibilities[**C8**, **C6**]: (i) newcomer's demand can be satisfied by the weight of the dummy session, which requires 1 GPS call resulting in $o(N^2)$ overall complexity; (ii) sessions already in the system have to be re-optimized in order for the newcomer to be admitted, which takes $o(N)$ GPS calls leading to an $o(N^3)$ complexity like the gradient method. In either case the re-optimization of the system after successful admission has to be undertaken (requiring $o(N)$ GPS calls), but this can be done during server idle periods.

6.4 Conclusions

To shortly summarize the proposed CAC algorithms, it is trivial that their computational complexity is higher than of traditional CAC based on the guaranteed rate approach. On the other hand, with the proposed methods one can achieve

- better utilization of network resources,

- bandwidth and delay de-coupling [**C7**],
- more flexible weight assignment and
- in the case of incremental CAC best-effort awareness [**C8**].

Since complexity of the proposed CAC algorithms have increased questions related to their applicability to make on-the-fly decision arise. Therefore either

- methods supporting prompt admission or rejection of flows are desired - with further optimization of flows in idle periods -

  or

- the algorithmic complexity must be kept low enough to be applicable in nowadays hardware/software technology.

The proposed incremental CAC algorithm aims to control CAC in the former way, utilizing idle periods of the server to re-optimize weight of sessions already in the system. On the other hand, as the computational power of future equipment ever increases, algorithms with higher complexity may also become or even have become feasible like the proposed gradient approach.

Also note, that in these cases not only the admittance / rejection of flows are handled but also proper weight assignments considering both bandwidth and delay requirements are managed.

Chapter 7

Application Possibilities of New Results

Throughout the dissertation I investigated ideal fluid models of fair schedulers. These models can not be realized in real application, however the transformation of achieved results is straightforward following the transformation of GPS to Packetized GPS.

Further, the currently accepted approach for the design of fair schedulers is based on the deterministic model derived by Parekh et al. This as I have also shown is overly conservative and leads to limitations on capacity. Thus, the introduction of delay bounds even with higher computational complexity can help to utilize networks more efficiently without wasting resources. On the other hand, we must note that deterministic description of a system is in disharmony with statistical multiplexing, though it is usually applied for applications with real time requirements.

An other issue I have proposed alternatives is to *relax* the bandwidth and delay coupling of GPS based schedulers. This is important in the case of future low bandwidth real time applications like voice over ATM/IP. Here, with the traditional weight assignment low delay can only be guaranteed through over-provisioning. What I propose is to set session weights taking into account delay requirements. I have also proposed algorithms that can perform the above task.

I must note that the *incremental CAC* algorithm also suits nowadays best-effort like traffic environment, since resources allocated to the *dummy* session can be used to serve the

best-effort service class without any side effects of destroying QoS guarantees of priority classes.

Chapter 8

Conclusions

Call admission control in GPS schedulers has not been widely analyzed by the research community due to its complexity. Initial CAC efforts based on deterministic bounds had limited applicability since these bounds were quite loose but simple. Further, the general rate-proportional weighting of sessions (like WFQ, WF^{2}Q, etc...) lacks the ability to handle both delay and bandwidth requirement in an efficient way. To overcome these limitations, we introduced a bandwidth and delay de-coupled GPS system, where weights are set to satisfy delay requirements while long term sustained rate is guaranteed 'on the fly'. Since the simple worst case bounds of rate-proportional weighting are no longer hold for our system, we developed a computationally effective method to compute the corresponding worst case delay bounds. Our method not only provides solution in a more general case, for arbitrary weighting of session, but also gives much tighter worst case delay bound than those widely used. However, all these efforts would be of limited use if no methods were provided to handle weight assignment and perform CAC functionality. Hence, we developed and analyzed several of our CAC approaches all handling both delay and bandwidth demands. For numerical results a real scenario of a 25 Mbps link was analysed with three kinds of QoS sensitive applications (IP telephony, mpeg video and video conference). In this representative scenario our proposed CAC with bandwidth delay decoupling outperformed the traditional rate proportional system in means of call blocking. Further, complexity analysis of different CAC methods was also presented.

On the whole, we propose a novel approach to admission and flow control in GPS systems where delay and bandwidth demands are handled independently of each other.

Appendix A

Notation List

$A_i(t_1, t_2)$	arrival function for session i in the interval $[t_1, t_2)$
C	link capacity
D_i	session i delay
D_i^*	session i worst-case delay bound
F	finishing time of a packet in PGPS
$L(i)$	the index of session clearing its backlog at the i^{th} order
N	number of sessions in a GPS server
Q_i	session i backlog
$R(u)$	session rate in time
$W_i(t_1, t_2)$	session i traffic served in the interval $[t_1, t_2)$
$\mathcal{B}(t)$	set of sessions backlogged at time t
$\mathcal{F}(t)$	set of sessions already cleared their backlogs at time t
ϕ_i	session i weight in the GPS system
ρ	long term sustained rate; leaky bucket parameter
σ	maximum burst size; leaky bucket parameter
g_i	guaranteed service rate of session i in a GPS server
r	service rate of the GPS server
$r_i(t)$	the service rate of session i at time t
r_k^i	the service rate of session i prior to the k^{th} backlog clear
t_i	the i^{th} finishing time or backlog clear in the GPS system
$t_{i,k}$	the finishing time of session k still backlogged after the first $i-1$ backlog clears

Index

Bibliography

[Bou96] J.-Y. Le Boudec. Network calculus made easy. Technical Report EPFL/DI 96/218, Networking and Communication Lab, EPFL, Lausanne, Switzerland, December 1996.

[Bou98] Jean-Yves Le Boudec. Selected lecture notes. Technical report, ICA Ecole Polytechnique Federale de Lausanne, October 1998.

[BZ94] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM'94*, pages 120–128, San Francisco, March 1994.

[BZ96] J.C.R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *Proceedings ACM SIGCOMM'96*, pages 143–156, August 1996.

[Cha92] Cheng-Shang Chang. Stability, queue length and delay, part II: stochastic queueing networks. Research Report RC 17709 (#77963), IBM Research Division, Yorktown Heights, New York, February 1992.

[Cru91a] Rene Leonardo Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Trans. Inform. Theory*, 37(1):114–131, January 1991.

[Cru91b] Rene Leonardo Cruz. A calculus for network delay, part II: Network analysis. *IEEE Trans. Inform. Theory*, 37(1):132–141, January 1991.

[Cru98] R. L. Cruz. SCED+: efficient management of quality of service guarantees. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, page 625, San Francisco, California, March/April 1998.

[CSZ92] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architetcture and mechanism. In *Proceedings of ACM SIGCOMM'92*, pages 14–26, Baltimore, Maryland, August 1992.

[DKS89] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 1–12, Austin, Texas, September 1989. ACM. also in Computer Communications Review, 19 (4), Sept. 1989.

[DKS90] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26, January 1990.

[DLS98] N. G. Duffield, T. V. Lakshman, and D. Stiliadis. On adaptive bandwidth sharing with rate guarantees. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, page 1122, San Francisco, California, March/April 1998.

[GGP97] Leonidias Georgiadis, Roch Guérin, and Abhay Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. *IEEE Transactions on Information Theory*, 43(5):1518–1535, September 1997.

[GGPS96] L. Georgiadis, Roch Guérin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[Gol90] S. Golestani. A stop-and-go queueing framework for congestion management. In *Proceedings of ACM SIGCOMM'90*, pages 8–18, Philadelphia, PA, September 1990.

[Gol94] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, CA, June 1994.

[GV97] P. Goyal and H.M. Vin. Generalized guaranteed rate scheduling algorithms: A framework. *IEEE/ACM Trans. On Networking*, 5(4):561–571, August 1997.

[GVC97] P. Goyal, H.M. Vin, and H. Cheng. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Trans. On Networking*, 5(5):690–703, October 1997.

[Kle75] L. Kleinrock. *Queueing Systems*. John Wiley and Sons, 1975.

[Kle76] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*. Wiley, 1976.

[Kur92] James F. Kurose. On computing per-session performance bounds in high-speed multi-hop computer networks. In *Sigmetrics 1992*, pages 128–139, New Port, Rhode Island, June 1992. ACM.

[mpe] Mpeg video statistics and bandwidth traces. Internet. http://www2.ncsu.edu/-eos/service/ece/project/rtcomm/ewfulp/www/sources/sourcestats.html.

[Nag85] John Nagle. On packet switches with infinite storage. Request for Comments 970, Internet Engineering Task Force, December 1985.

[PG92] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks — the single node case. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, volume 2, pages 915–924 (7A.3), Florence, Italy, May 1992. IEEE.

[PG93a] 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 Transactions on Networking*, 1(3):344–357, June 1993.

[PG93b] A.K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. In *Proceedings of the INFOCOM'93*, pages 521–530, San Francisco, CA, March 1993.

[PG94] 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 Transactions on Networking*, 2(2):137–150, April 1994.

[RKJ95] S. Rajagopal, V.G. Kulkarni, and S. Stidham Jr. Optimal flow control of a stochastic fluid-flow system. *IEEE Journal on Selected Areas in Communications*, 13(7), September 1995.

[Rob90] John Robinson. Congestion control in BBN packet-switched networks. *ACM SIGCOMM Computer Communication Review*, 20(1):76–90, January 1990.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Request for Comments (Proposed Standard) 2212, Internet Engineering Task Force, September 1997.

[SS99] David Starobinski and Moshe Sidi. Stochastically bounded burstiness for communication networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, March 1999.

[SSZ98] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *Proceedings of INFOCOM'99*, 1998.

[Sti96] Dimitrios Stiliadis. *Traffic Scheduling in Packet-Switched Networks: Analysis, Design, and Implementation*. PhD thesis, University of California, Santa Cruz, June 1996.

[SV96] Dimitrios Stiliadis and Anujan Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. In *Proceedings of the Conference*

on Computer Communications (IEEE Infocom), pages 111–119, San Fransisco, California, March 1996.

[SV97] D. Stiliadis and A. Varma. A general methodology for designing efficient traffic scheduling and shaping algorithms. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pages 326–335, Kobe, Japan, April 1997.

[SV98] D. Stiliadis and A. Varma. Rate-proportional servers: A design methodology for fair queueing algorithms. *IEEE/ACM Trans. On Networking*, 6(2):164–174, April 1998.

[SW97] S. Shenker and J. Wroclawski. General characterization parameters for integrated service network elements. Request for Comments (Proposed Standard) 2215, Internet Engineering Task Force, September 1997.

[Wor94] T. Worster. Modelling deterministic queues: The leaky bucket as an arrival process. In *International Teletraffic Congress, ITC-14*, volume 1a, pages 581–590. ITC, June 1994. Antibes, France.

[Wro97] J. Wroclawski. Specification of the controlled-load network element service. Request for Comments (Proposed Standard) 2211, Internet Engineering Task Force, September 1997.

[YS93] Opher Yaron and Moshe Sidi. Performance and stability of communication networks via robust exponential bounds. volume 1, pages 372–385, June 1993.

[YS94] Opher Yaron and Moshe Sidi. Generalized processor sharing networks with exponentially bounded burstiness arrivals. In *Proceedings of INFOCOM*, Toronto, Canada, June 1994.

[YSS92] N. Yamanaka, Y. Sato, and K. Sato. Performance limitation of leaky bucket algorithms for usage parameter control and bandwidth allocation methods. *IEICE Transactions on Communications*, E75-B(2):82–86, 1992.

[ZF94] Hui Zhang and Domenico Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, February 1994.

[Zha95] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. In *Proceedings of the IEEE*, 83(10), October 1995.

[ZK91] H. Zhang and S. Keshav. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM'91*, pages 113–122, Zurich, Switzerland, September 1991.

[ZLKT97] Zhi Li Zhang, Zhen Liu, Jim Kurose, and Don Towsley. Call admission control schemes under generalized processor sharing scheduling. *Telecommunications Systems*, 7(1-3):125–152, July 1997.

[ZTK94] Z.-L. Zhang, D. Towsley, and J. Kurose. Statistical analysis of generalized processor sharing scheduling discipline. In *Proceedings of ACM SIGCOMM'94*, pages 68–77, London, UK, September 1994.

Publications of new results

[J] **JOURNALS**

[**J1**] I. Cselényi, **Róbert Szabó**, I. Szabó, A. Latour-Henner, C. Gisglrd, and N. Bjõrkman. Experimental platform for telecommunication resource management. *Computer Communications*, 21:1624–1640, 1998.

[**J2**] G. Fodor, T. Henk, T. Marosits, **Róbert Szabó** and L. Westberg. Simulative analysis of optimal routing and link allocation strategies in B-ISDN networks. *Periodica Polytechnica*, 42(3):275–197, 1998.

[**J3**] P. Barta, Cs. Lukovszki, Ádám Marquentatn, G. Fodor, and **Róbert Szabó**. Optimized resource utilization on call and cell level in atm networks. *Magyar Távközlés: Selected Papers from the Hungarian Telcommunication Periodicals*, pages 2–6, 1999.

[**J4**] I. Cselényi and **Róbert Szabó**. Service specific information based resource allocation for multimedia applications. *Design Issues of Gigabit Networking of INFORMATICA JOURNAL*, 23(3):317–325, 1999. ISSN 0350-5596.

[**J5**] D. Hoványi, G. Kováts, S. Daróczy, and **Róbert Szabó**. Voip lehetõségei. *Magyar Távközlés*, 11:10–14, Nov. 1999. in Hungarian.

[J6] C. András, A. Takács, and R. Szabó. VoIP szolgálatok minõségbiztosítása - Quality management of VoIP services. *Magyar Távközlés*, 5:14–17, May 2000. in Hungarian.

[J7] Cs. Lukovszki, **Róbert Szabó**, and T. Henk. Performance evaluation of a hybrid atm switch architecture by parallel discrete event simulation. *INFORMATICA JOURNAL*, 2000. to appear at the second volume in 2000

[C] **CONFERENCES**

[C1] G. Fodor, T. Henk, T. Marosits, and **Róbert Szabó**. On the call and cell level resource allocation in atm networks. In *Proceedings of European Simulation Symposium 1995*, pages 475–484, Erlangen-Nuremberg, Germany, Oct. 1995. SCS, The Society for Computer Simulations.

[C2] G. Fodor, T. Marosits, and **Róbert Szabó**. Comparison of conservative parallel simulation techniques for multistage interconnection networks. In *Proceedings of European Simulation Multiconference 1996*, pages 513–517, Budapest, Hungary, June 1996. SCS, The Society for Computer Simulation International.

[C3] **Róbert Szabó** and Cs. Lukovszki. Parallel cell scale simulation of a hybrid atm switch architecture. In *Proceedings of SPECTS '98 - 1998 Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 53–57, Reno, Nevada, July 1998. SCS.

[C4] I. Cselényi and **Róbert Szabó**. Performance evaluation of an intelligent resource allocation scheme for multimedia applications. In *Proceedings of SPECTS '98 - 1998 Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 302–306, Reno, Nevada, July 1998. SCS, The Society for Computer Simulation International, SCS.

[C5] Cs. Lukovszki, **Róbert Szabó**, and T. Henk. Parallel simulation model of a novel hybrid min architecture. In *Proceedings of European Simulation Symposium, ESS'98*, pages 725–729, Oct. 1998.

[C6] **Róbert Szabó**, P. Barta, F. Németh, and J. Bíró. A novel approach to admission and flow control in generalized processor sharing (GPS) schedulers. In *4th International Conference on Applied Informatics*, Aug. 1999. Eger-Noszvaj, Hungary.

[**C7**] **Róbert Szabó**, P. Barta, F. Németh, J. Bíró, and C.-G. Perntz. Non rate proportional weighting of generalized processor sharing schedulers. In *Proceedings of GLOBECOM'99*, volume 2, pages 1334–1339, Rio de Janerio, Brazil, Dec. 1999.

[**C8**] **Róbert Szabó**, P. Barta, F. Németh, J. Bíró, and C.-G. Perntz. Call admission control in generalized processor sharing (GPS) schedulers using non-rate proportional weighting of sessions. In *Proceedings of INFOCOM 2000*, volume 3, pages 1243–1252, Tel-Aviv, Israel, Mar. 2000.

[**C9**] **Róbert Szabó**, P. Barta, F. Németh, and J. Bíró. Worst-case deterministic delay bounds for arbitrary weighted generalized processor sharing schedulers. In *Proceedings of Networking 2000*, pages 727–739, Paris, France, May 2000.

[C10] P. Barta, F. Németh, **Róbert Szabó**, and J. Bíró. Network qos provisioning through per node traffic characterization. submitted to Globecomm2000.

[W] **WORKSHOPS AND OTHERS**

[**W1**] J. Bíró, T. Henk, **Róbert Szabó**, P. Barta, and F. Németh. Issues on packet service disciplines for guaranteed quality of service networks. In *COST 257 MC Meeting*, Jan. 1999. Faro, Portugal.

[**W2**] J. Bíró, T. Henk, **Róbert Szabó**, and P. Barta. Non rate proportional generalized processor sharing schedulers. In *IFIP WG 6.3 Workshop*, Aug. 1999. Rethymnon, Crete, Greece.

[**W3**] G. Rétvári and **Róbert Szabó**. Qos-based routing and ip multicasting: A framework. In *Fifth EUNICE Open European Summer School*, Sept. 1999. Barcelona, Spain.