ADMISSION AND FLOW CONTROL IN GENERALIZED PROCESSOR SHARING SCHEDULERS

Collection of 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
2000

1 Introduction

Offering a wide-range of Quality of Service (QoS) guarantees is a central issue in modern high-speed multi-service telecommunication networks. Several mechanisms have been introduced and studied under the aegis of QoS networks. One important family of such mechanisms is traffic scheduling algorithms within individual network elements (switches, routers) [BZ94, BZ96, CSZ92, Gol90, Gol94, Zha95]. I only considered individual scheduling elements of telecommunication networks like an output port of an ATM switch or a router throughout the dissertation.

Traffic scheduling algorithms usually operate on packet/cell level and assume there are different *session* traffic to be scheduled in a certain way. The task of service disciplines at switching nodes is to control the order in which incoming packets are served in order to provide (end-to-end) *performance bounds*. Many scheduling algorithms have been proposed in the literature. One of the most significant ones is the Generalized Processor Sharing (GPS) [PG92, PG93a, PG93b, PG94], which is a generalized version of Uniform processor Sharing [Kle76]. The packet-by-packet version of GPS (PGPS) is essentially the same as the Weighted Fair Queuing (WFQ) [DKS90], however, they were independently developed. Parekh and Gallager introduced a notation for describing a GPS system in [PG92] that I recall now:

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 served in the interval $[t_1, t_2]$, then a GPS server is defined as one for which

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j}, \quad j = 1, 2, \ldots, N, \tag{1.1}$$

for any session i that is continuously backlogged[1] in the interval $[t_1, t_2]$. The immediate consequence of this definition is that every session has a minimum *guaranteed service rate* that is $g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r$ [PG92].

In GPS each input traffic can be shaped by a *leaky bucket* (see Fig. 1) that is characterized by a token pool depth (σ) and a token generation rate (ρ) [Wor94, YSS92]. An important advantage of using leaky buckets is that it 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 [PG92].

Further, the amount of incoming traffic arriving from a leaky bucket in the interval $[t_1, t_2]$ from the active source i assuming infinite capacity links can be characterized by the function $A_i(t_1, t_2)$. If $A_i(t) = A_i(0, t) = \sigma_i + \rho_i t$, then by definition session i starts *greedy*,

[1] the session buffer is not empty

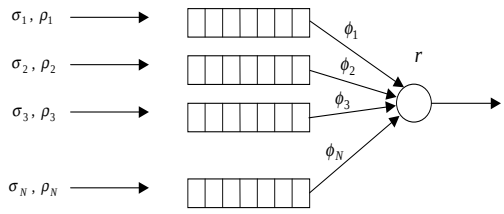

Figure 1: GPS server with leaky bucket inputs

i.e., it 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*. The main result of [PG92] is revisited here supporting my work:

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[2]*. Furthermore, assuming that for each session i $g_i \geq \rho_i$, then*

$$Q_i^* \leq \sigma_i \quad \textit{and} \quad D_i^* \leq \frac{\sigma_i}{g_i}. \tag{1.2}$$

The significance of this result is that for worst case behavior one has to analyze a greedy system 'only', which makes the analysis more tractable compared to any arbitrary arrival pattern imposed to the system. Therefore, hereafter without losing generality I considered only greedy GPS systems, where all sessions start greedy at time zero, the beginning of a system busy period. The above result is also valid when the internal link capacities are infinite ($C_i = \infty$). Thus, I performed my analysis for the infinite capacity case, however, the extension to finite capacities is straightforward [Bou96, Bou98].

The GPS service discipline is an ideally fair 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 GPS system can not be accomplished in practice, there are several schedulers emulating it at the background to determine packet serving orders. Packet-by-packet versions of GPS were also analyzed establishing important relations between the fluid model and the packetized versions [PG92]. In most cases analysis of the GPS model is sufficient since results can be transformed to packetized versions in a straightforward manner [Gol94, PG92].

[2] *an interval in which the server is continuously working*

2 Objectives of the Dissertation

The main goal of the dissertation is to lay down the foundation of a novel approach to characterize Generalized Processor Sharing (GPS) schedulers in a more efficient way. Earlier deterministic approaches describing GPS schedulers were overly conservative and led to limitations on capacity. In my dissertation

- I have derived a new iterative approach that allows the characterization of any arbitrary weighted GPS fluid server;
- I have introduced a bandwidth and delay de-coupled system, where weights are not set in proportion to session bandwidth demands but assigned based on the delay requirements;
- based on the above characterization I have given an algorithmic method to calculate worst case delay bounds for GPS servers with the following attributes:
  - in case of rate-proportional weighting the derived worst case delay is tighter than the conservative one defined by Parekh et al. [PG93b, PG92],
  - for locally stable sessions (where $g_i \geq \rho_i$) the delay bound given is also tighter than the one calculated by the guaranteed rate g_i,
  - it works with any arbitrary weighting in contrast with the one given by Parekh et al. [PG93b, PG92].

However, all above efforts would be of limited use if additional methods for call admission control (CAC) and weight assignments were not given. Thus, I integrated the task of CAC and weight assignment and developed several algorithms to solve the problem. Weight assignment in a non rate-proportional GPS system is highly non-trivial task as shown in the dissertation.

Throughout the dissertation I tried to contribute to the area of quality of service schedulers. I paid special attention to analytically describe the corresponding ideal fluid GPS system. In parallel with the analytical approaches I performed numerical evaluation of the proposed algorithms for performance evaluation.

3 Methodology of Research

Throughout the dissertation I have investigated *deterministic fluid models* of fair schedulers. I have used mathematical analysis for deriving important attributes of such systems while I have applied numerical evaluation (Mathematica) where the complexity of the task did not allow the closed form formalization of results.

4 New Results

4.1 Arbitrary Weighting of GPS Servers [W1, W2, C6, C7, C8, C9]

A large subclass of GPS-based schedulers is the rate-proportional servers (RPS), in which weights are set according to session bandwidth demands.[3] This, however, introduces coupling between bandwidth and delay, i.e., the delay bound is inversely proportional to the long term allocated bandwidth (ρ). The drawback with this weighting is that if one wants to decrease the delay bound of a session then the corresponding bandwidth should be increased in order to maintain rate-proportional bandwidth allocation. Clearly, this can result in an inflexible resource allocation and may lead to waste of network resources.

As opposed to the rate-proportional weighting I consider cases where the weight ϕ_i is set independently of the parameter ρ_i. This is further on referred to as *arbitrary weighting* of sessions. Thus, for some sessions the minimum guaranteed service rate can be smaller than the corresponding token generation rate ρ, nevertheless, the system remains stable as long as the sum of all sustained arrival rates (ρ) remains strictly less than the service rate of the GPS server [PG92, PG93b], [**C7**].

Before proceeding on I first introduce a general notation system relevant to further understanding:

Since the GPS server is *work-conserving* and I analyzed a *greedy* (1.2) system it follows that each session i starts with a burst σ_i at time zero, the beginning of the *system busy period*, and continues to transmit with its ρ_i long term sustained rate. Additionally, only *globally stable* systems are analyzed where $\sum_{j=1}^{N} \rho_j < r$; hence the GPS server sooner or later empties all session buffers and will only transmit the incoming sustained rates. Note, that I talk about a fluid system, where session demands are served concurrently.

The session whose buffer is not empty is called *backlogged* and its buffered traffic is called its *backlog*, denoted by $Q_i(t)$ at time $t > 0$.

Now let $\mathcal{L} = \{L(J) | \, J = 1, \ldots, N\}$ be an ordered set of indices where $L(I)$ means that the session $i = L(I)$ backlog is cleared as I^{th} in order. Further, let's define $I = L^{-1}(i)$ and then t_I denotes the time instant when the backlog of session i becomes zero. By definition, let $t_0 = 0$ the start of the system busy period with all greedy sessions. Further, let the service rate of session i be $r_i(t)$ at time t.

THESIS 1 ([W1, W2, C6, C7]): *I introduced a novel iterative algorithmic approach to handle and characterize arbitrary weighted GPS servers.*

THESIS 1.1 ([C6, C7, C9]): *I have explicitly derived that the service rate of still backlogged sessions in a GPS server is a step function and can be formulated as the weighted share of the reduced service rate for the still backlogged sessions.*

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

Theorem 1.1.1 (Discrete Rate Function [C6, C7, C9]). *In the time interval* $[t_{k-1}, t_k),\ k = 1, \ldots, N$ *within the system busy period the backlogged session* j *is served at a rate*

$$r_j(t) = r_1^j = g_j = \frac{\phi_j}{\sum_{l=1}^{N} \phi_l} \qquad j \in \{1, \ldots, N\},\ k = 1$$

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

$$k = 2, \ldots, N. \tag{4.1}$$

Note, that Parekh and Gallager in [PG93b] have used a similar formula without formally proving it. In my case, the construction of my algorithmic approach demanded the formal proof of the above theorem.

A corollary stating the monotone increasing attribute of the former theorem immediately follows:

Corollary 1.1.2 ([C6, C7, C9]). r_k^j *is a monotone increasing function of* k *for every backlogged session* j *for which* $\phi_j > 0$.

THESIS 1.2 ([C6, C7, C9]): *I have shown* where *and* when *the maximal backlog of each session is experienced in a greedy GPS system throughout the following two lemmas and the consecutive theorem.*

Note the differences between my work and the one presented by Parekh and Gallager in [PG93b]. While they introduced a theoretically method to determine the maximum backlog, here, I determine the time instances where each session experiences its maximum backlog depending on its weight assignment.

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 1.2.1 ([C6, C9]). *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.*

Lemma 1.2.2 ([C6, C9]). *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.*

Further I have shown that,

Theorem 1.2.3 (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).*

The importance of the above theorem is that in order to determine the maximum backlog one has to trace backlogs only at times sessions clear their backlogs.

Since all the way along the backlog clearing times played a very important role,

THESIS 1.3 ([C6, C7, C9]): *I have constructed an iterative algorithm for the calculation of backlog clearing time instances.*

Theorem 1.3.1 ([C6, C7, C9]). *The backlog clearing times at the k^{th} step (t_k) of an arbitrary weighted GPS system can be calculated by the following iterative formula:*

$$t_k = \min\{t_{i,k} \mid t_{i,k} > 0;\ i \in \{1,\dots,N\} \setminus \bigcup_{j=1}^{k-1} L(j)\} \tag{4.2}$$

where

$$t_{i,k} = \frac{S_{i,k}}{r_k^i - \rho_i} \tag{4.3}$$

and

$$S_{i,k} = \sigma_i + \sum_{j=1}^{k-1} r_j^i\ (t_{j-1} - t_j) + r_k^i\ t_{k-1}. \tag{4.4}$$

Note, that $t_{i,k}$ can be interpreted as the candidate finishing times of the k^{th} step from which the smallest positive one is picked.

Figure 2 shows the flowchart of the proposed algorithm.

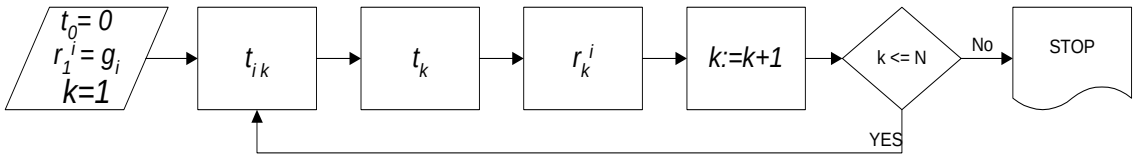

Figure 2: Flowchart of the iterative algorithm

Also note the recursion

$$S_{i,1} = \sigma_i \tag{4.5}$$

$$S_{i,k+1} = S_{i,k} + (r_{k+1}^i - r_k^i)\ t_k, \tag{4.6}$$

and that

$$L(k) = \arg\min_i \{t_{i,k} \mid t_{i,k} > 0;\ i \in \{1,\dots,N\} \setminus \bigcup_{j=1}^{k-1} L(j)\}. \tag{4.7}$$

4.2 Delay Bounds of Arbitrary Weighted GPS Servers

THESIS 2 ([C6, C7, C9]): *I have proven that the general and widely used delay bound given by Parekh et al. is quite loose; therefore I have derived a novel formula for delay calculation giving tighter bounds for locally stable systems ($g_i \geq \rho_i$). The proposed formula also works with arbitrary weighted scenarios unlike the ones existing in the literature.*

THESIS 2.1 ([C6, C7, C9]): *I have proven that (i) for sessions clearing their initial burst with higher service rate than their corresponding arrival rate, their maximum experienced delay is equal to the time needed to serve the last amount of their initial burst; while (ii) for sessions failing the above criteria the maximum delay is experienced at the rate-changing time instant where their service rate* first *exceeds or equals their arrival rate.*

Theorem 2.1.1 (Delay Bound [C6, C7, C9]).
(i) If $r_i(t)$ as defined in (4.1) satisfies

$$r_i(\tau_i) \geq \rho_i \tag{4.8}$$

then

$$D_i = \tau_i \tag{4.9}$$

where τ_i is defined as

$$W_i(0, \tau_i) = \int_0^{\tau_i} r_i(t)\, \mathrm{d}t = \sigma_i. \tag{4.10}$$

(ii) If (4.8) 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

$$\forall t < t_j : \; r_i(t) < \rho_i \tag{4.11}$$

and

$$\forall t \geq t_j : \; r_i(t) \geq \rho_i \tag{4.12}$$

then

$$D_i = t_j - \frac{W_i(0, t_j) - \sigma_i}{\rho_i} \tag{4.13}$$

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

Note that (ii) corresponds to scenarios where there are sessions locally un-stable. On the other hand, if local stability is ensured then delay bounds calculated by (4.10) is at least as tight as the one given by Parekh et al. by (1.2). More formally:

THESIS 2.2 ([C6, C7, C9]): *I have proven that in the case of* locally stable system *the delay bound given by Theorem 2.1.1 is at least as good as the one given by Parekh et al., i.e.,*

Theorem 2.2.1 (Tighter Delay Bound [C6, C7, C9]). *For all locally stable sessions ($g_i \geq \rho_i$), where results of Parekh et al. holds, the following relation can be derived*

$$D_i \leq D_i^*, \tag{4.14}$$

where D_i is defined by Theorem 2.1.1 and D_i^* is defined by (1.2).

The statement of Thesis 2 for locally stable systems follows from Theorem 2.2.1 while its extension to arbitrary weighting is defined in Theorem 2.1.1. All the theorems used were proved analytically.

Figure 3 illustrates a four-session scenario, where the worst case delay is shown for both calculation methods. The red (thin black) line shows how the delay of session i is calculated by using the guaranteed rate approach. Blue (thick black) line with piecewise linear sections shows how the session i service curve takes shape if backlog clears, i.e., service rate updates are also considered. D_i shows the maximum delay calculated with my proposed method, t_i is the session i backlog clearing time while D_i^* is the delay calculated with the guaranteed rate approach. At each joint point of the piecewise linear sections a session empties its backlog, hence further on demanding less service than it has previously allocated. Note the greedy system.

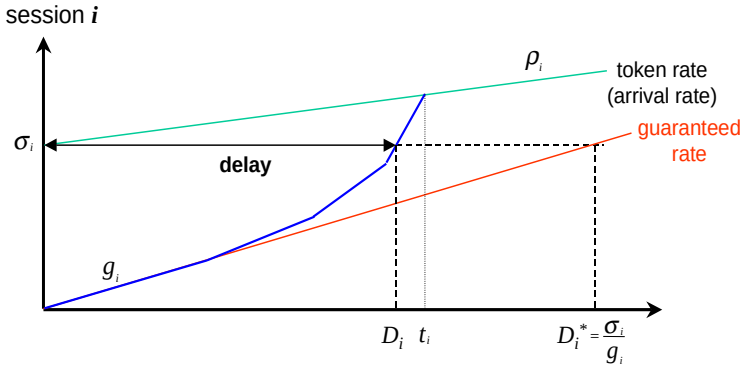

Figure 3: Service and arrival functions of a 4 session GPS system

Possible weighting scenarios are shown in Figure 4. However, if the weight space is normalized then the rate proportional servers (RPS) set will shrink to a single point in the possible parameter space.

4.3 The Impact of Weight Changing on Delay Bounds [C8, C6]

My call admission control algorithms are based on the previously outlined approach with arbitrary (non rate-proportional) weighting of sessions and consider the introduced tighter

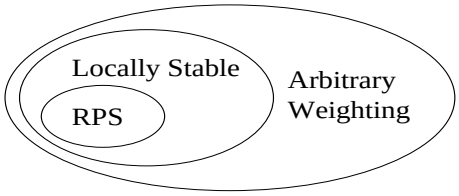

Figure 4: Weighting possibilities for sessions

delay bound. Here I also considered a GPS node that serves a certain number of leaky bucket constrained sessions simultaneously. By relaxing the rate-proportional weighting I lost the unambiguous assignment of ϕ_i's, i.e., I had to show a method for setting session weights to satisfy both their delay and bandwidth demands. However, the CAC for bandwidth is quite straightforward and is based on the well known stability inequality $\sum_{i=1}^{N} \rho_i < r$. On the other hand, the appropriate setting of weights regarding the delay requirements is highly nontrivial as shown later.

THESIS 3 ([C8, C6]): *I have analytically deduced the relation of a single session's weight change and its experienced worst case delay for arbitrary weighted GPS systems.*

Let a **'** further denote the altered system parameters.

My first lemma shows that increasing a session's weight results in the increment of the corresponding rate function, while the rate functions of other, still backlogged sessions will slow down till session i backlog is cleared.

Lemma 3.1 ([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*

$$\exists I : i = L(I) = L'(I)$$

for which

$$\begin{aligned} {r'}_k^j &< r_k^j \quad \forall j \in \mathcal{B}(k) \setminus \{i\},\ k \in \{1, \ldots, I\} \\ {r'}_m^i &> r_m^i \quad \forall m \in \{1, \ldots, I\}, \end{aligned} \tag{4.16}$$

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

Since Lemma 3.1 only defines rate function relations at backlog clearing times I introduced the following lemma that puts the statement of Lemma 3.1 into continuous time.

Lemma 3.2 ([C8, C6]). *By taking the assertions of Lemma 3.1 and further assuming that* ${t'}_k > t_k \, \forall k \in \{1, \ldots, I-1\}$ *then*

$$\begin{aligned} {r'}_j(t) &< r_j(t) \quad \forall j \in \mathcal{B}(t) \setminus \{i\},\ t \in [0, {t'}_I] \\ {r'}_i(t) &> r_i(t) \qquad\qquad \forall t \in [0, {t'}_I] \end{aligned} \tag{4.17}$$

where $r_j(t)$ is the service rate of session j at time t as defined in (4.1).

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.

Next, I step-by-step relaxed the assertions of Lemma 3.2. First I have proven how the finishing times shift if a session weight is changed. In the ensuing step I relaxed the restriction of unvarying session finishing time order.

Lemma 3.3 ([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*

$$\exists I : i = L(I) = L'(I)$$

for which

$$\begin{aligned} &t'_k \geq t_k \quad k \in \{1, \ldots, I-1\} \\ &t'_I \leq t_I \quad \textit{where } i = L(I). \end{aligned} \tag{4.19}$$

Lemma 3.4. *If the session backlog clearing order changes due to a session weight change in a GPS system, then all the results of Lemma 3.1 - Lemma 3.3 will continuously hold.*

From the above lemmas, I have deduced another important lemma dealing with the service function:

Lemma 3.5 ([C8, C6]). *The service function of session i ($W_i(t, \underline{\phi}, \underline{\sigma}, \underline{\rho}) = W_i(0, t)$ with the given input parameter vectors of $\{\underline{\phi}, \underline{\sigma}, \underline{\rho}\}$) is a monotone* increasing *function of the session's weight. In other words, for any time t*

$$\frac{\Delta W_i(t, \underline{\phi}, \underline{\sigma}, \underline{\rho})}{\Delta \phi_i} \geq 0. \tag{4.20}$$

From the above lemma it follows that by increasing a session weight one can cut down the corresponding backlog clearing time, i.e.,

Corollary 3.6 ([C8, C6]). *From Lemma 3.5 it follows that $t'_i \leq t_i$ if $\phi'_i = \phi_i + \delta$ where $\delta > 0$.*

My main theorem that summarizes all the results of this thesis defines the relationship between the worst case delay bound and the session weight. I have proven the forthcoming theorem by the former lemmas.

Theorem 3.7 ([C8, C6]). *The worst case delay of session i is a monotone decreasing function of the corresponding weight. In other words,*

$$\frac{\Delta D_i(\underline{\phi}, \underline{\sigma}, \underline{\rho})}{\Delta \phi_i} \leq 0. \tag{4.21}$$

All these efforts were made to control the worst-case delay of sessions. From Theorem 3.7 it is now known that in order 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 my CAC approaches.

4.4 CAC Algorithms for Arbitrary Weighted GPS Servers [C8, C6]

In an ideal scenario the call admission control (CAC) function only says whether the system is able to satisfy all session 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 feasible realization. Thus further on I tie the problem of CAC with finding a solution of the feasible set $\mathcal{F}$, 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 $\underline{\phi} \in \mathcal{F}$; $\underline{D}, \underline{d}(\underline{\phi})$ 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.

THESIS 4 ([C8, C6]): *I have constructed a novel approach to handle call admission control in GPS system and compared it's performance to other CAC algorithms I have adopted from other scientific fields. I have also introduced an additional performance metric called* delay-fairness*.*

Algorithm 4.1 *(Random search [**C8**, **C6**]).* Solution is searched in a random manner with limited inheritance and frequent mutations. This algorithm is a descendant of genetic algorithms.

THESIS 4.1 ([C8, C6]): *Throughout the adaptation of the gradient method to the addressed problem I have constructed a new performance metric* (delay-fairness) *characterizing the relative delay handling of sessions in any GPS server. I have derived an objective function that optimizes the above performance metric through the weight assignment.*

Algorithm 4.2 *(Gradient Method [**C8**, **C6**]).* The algorithm is based on an N dimensional vector-scalar function used as the objective of minimization (O), where delay bound exceed is punished with a penalty function for each session. The way the feasible set is approached is given by Theorem 3.7, giving an unambiguous answer for how to change a session weight to improve its delay bound.

There were two objective functions used in the gradient method, where the first is further referred to as *flat*, since only delay exceeds are punished while over-satisfaction of session demands are not taken into account:

$$O_1(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) = \sum_{i=1}^{N} I_{(d_i - D_i)}(d_i - D_i)^2, \tag{4.22}$$

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

$$I_{(x)} = 0 \quad x < 0$$
$$I_{(x)} = 1 \quad x \geq 0. \tag{4.23}$$

This objective function has certain drawbacks just to mention the unfair favor of some sessions whose delays are low below their delay requirements. Besides, because of the flat objective function the algorithm may be too slow to release excess resources of sessions located in the flattened region of the objective function as there will be no immediate effect of the parameter change. Nevertheless, as the parameter space is normalized in each step, sessions outside the feasible region will lead to a solution. To overcome the former drawbacks I constructed an objective function that ensure faster convergence and achieves a *delay-fair* feasible solution if exists.

I have defined the following performance metric for a GPS system:

Definition 4.3 (delay-fairness). The delay fairness (Γ) of a GSP system for a given $\{\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}\}$ is defined by the sum of quadratic variance of delays or in a more formal way

$$\Gamma(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) = \frac{1}{N}\sum_{i=1}^{N}(\frac{d_i}{D_i} - \frac{1}{N}\sum_{j=1}^{N}\frac{d_j}{D_j})^2. \tag{4.24}$$

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

$$O_2(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) = \Gamma(\underline{\sigma}, \underline{\rho}, \underline{D}, \underline{\phi}) + O_1. \tag{4.25}$$

Note that different weighting factors may be used to improve convergence speed, though in sight investigation of this problem is out of the scope of the recent dissertation.

Further it is trivial to show that 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

$$\frac{d_i^*}{D_i} = \frac{d_j^*}{D_j} \quad \forall i, j \in \{1, \ldots, N\}. \tag{4.26}$$

Furthermore, I have also shown that ϕ^* not only provides a delay-fair solution, but also shows how much session delays can further be tightened still maintaining feasibility. It can be said, that the GPS system defined by $\{\underline{\sigma}, \underline{\rho}, \underline{\mathbf{d}}^*, \underline{\phi}\}$ is at its *efficiency bound*, i.e., if any of session demands was tighter while the rest remained the same then there would be no feasible solution to satisfy all sessions' requirement.

THESIS 4.2 (Incremental CAC [C8, C6]): *I have constructed a CAC decision algorithm with the following attributes: CAC is considered to take place each time a new session enters the system of $N-1$ sessions served in a GPS system in a way that:*

- *There is an additional* dummy *session with $\sigma_d = \infty$ in the system taking all excess resources;*

- *weights are set that each session experiences it's allowed maximum delay when considering the dummy session too.*

Now the weight for the newcomer is taken from the dummy session where

- *In the case the weight of the dummy session allocated to the newcomer satisfies its delay demand; CAC function can immediately return acceptance confirmation with a feasible weight*

- *If the weight of the dummy session is not enough then re-optimization of sessions already in the system is needed. Since the newcomer will consume less resources than the dummy, thus allowing other sessions to release some of their allocated weights. If the server fails to satisfy newcomer's demand after re-optimization, CAC function rejects the call.*

In the case of positive acknowledgement the system must be re-optimized based on Theorem 3.7 in a way that the dummy session is inserted again at the $(N+1)^{th}$ place. However, 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 or near of its efficiency bound.

The flowchart of the algorithm is shown in Figure 5.

THESIS 4.3 ([C8, C6]): *I have analyzed the computational complexity of the GPS delay calculation and both the gradient and the incremental CAC approaches. I found that all algorithms bear polynomial complexity.*

The computational complexity of the proposed CAC algorithms can be summarized as follows [**C8**, **C6**]: a single GPS delay calculation for a given $\{\underline{\sigma}, \underline{\rho}, \underline{\phi}\}$ bears $o(N^2)$ complexity; the gradient method adds another $o(N)$ complexity to the GPS calculation resulting in $o(N^3)$ overall numerical complexity. However, the incremental CAC is able to make decisions in 1 step, i.e., a single GPS calculation of $o(N^2)$ complexity, if the share allocated to the dummy session satisfies the newcomer's need. Otherwise, it also fall back to $o(N^3)$ complexity.

Though the computational complexity of these algorithms are very similar, they are designed for different purposes. Incremental CAC can handle an additional best-effort service class throughout the dummy session, while the gradient approach provides delay-fair solution that extends the fairness of GPS like schedulers.

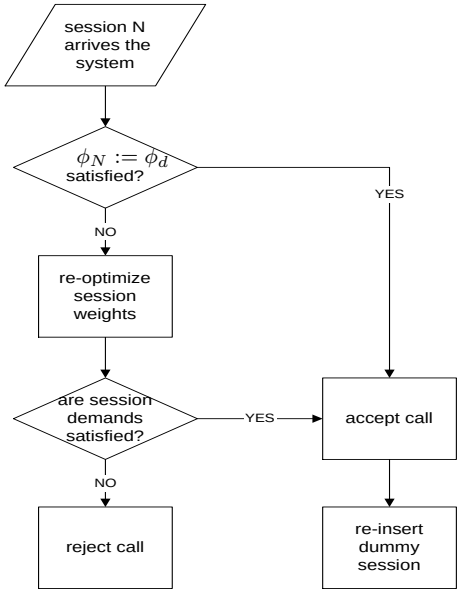

Figure 5: Flowchart of the incremental CAC algorithm

4.4.1 Concluding remarks

To shortly summarize the proposed CAC algorithms, it is trivial that their computational complexity is higher than that 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.

My 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.

5 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.

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.

An other issue I have proposed 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 proposed is to set session weights taking into account delay requirements. I have developed algorithms performing the previously mentioned 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.

The model I have presented here rather bears theoretical results just like its ancestor the GPS system. Nevertheless, packetized implementations of GPS systems are also exist regardless its computational complexity. Physical implementation of the proposed bandwidth-delay de-coupled system and the developed CAC algorithms are under work.

Acknowledgements

I owe special thanks to the team I worked with in the scheduling topic, namely József Bíró, Péter Barta and Felicián Németh.

The research work presented herein was supported by Ericsson Applied Research - Switch Lab, Stockholm and the High Speed Networks Laboratory of the Department of Telecommunications and Telematics, Budapest University of Technology and Economics. I would like to acknowledge Carl-Gunnar Perntz, Miklós Boda and Tamás Henk for their support.

References

[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.

[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.

[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.

[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.

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

[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 multiple node case. In *Proceedings of the INFOCOM'93*, pages 521–530, San Francisco, CA, March 1993.

[PG93b] 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.

[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.

[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.

[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.

[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.

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

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.