S^2GPS: Slow-Start Generalized Processor Sharing

Extended Abstract

Anastasios Stamoulis *Jörg Liebeherr*

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
Email: {as4r | jorg}@cs.Virginia.EDU

Abstract

Packet scheduling methods that approximate Generalized Processor Sharing (GPS) are currently the focus of much research in the area of Quality-of-Service (QoS) networks. The ability of GPS schedulers to provide rate guarantees as well as delay guarantees meets the demand of many network applications. This paper addresses a shortcoming of GPS which has been given little attention, however, which can have significant impact on the service provided by GPS. Since, with GPS, the service rate received by a session is proportional to the number of backlogged sessions in the system, the service rate of a session may change abruptly if some other session becomes active. This may result in abrupt increases of delay of consecutive packets. In this paper we propose a new scheduler, called *Slow-Start GPS* (S^2GPS), which alleviates the problem of abrupt delay increases when new sessions start transmitting. S^2GPS is a modification of GPS where sessions do not receive their guaranteed service rates immediately after they become active. Instead, the service rates of such sessions are gradually increased. This We will show that this prevents an abrupt delay increase of the other sessions. We derive delay bounds for sessions constrained by leaky buckets and we express quantitatively the advantages of the S^2GPS discipline.

1 Introduction

The *Generalized Processor Sharing (GPS)* scheduling method is known for providing support for *isolation* and *sharing* in a Quality-of-Service (QoS) network [4, 7, 8]. Even though GPS can provide rate guarantees to the session it services, a session that has been active for a long period of time can experience a sudden reduction in its service rate when some other, previously idle, session becomes active. The decrease of the service rates can be quite large, resulting in a possibly significant increase of the delay of consecutive packets of an active session. To alleviate this problem, we propose a modification to GPS, called *Slow-Start Generalized Processor Sharing (S^2GPS)* that prevents abrupt rate and delay increases by gracefully degrading the service rate of active sessions. This is accomplished by the following modification to the original GPS scheduling method. Whenever a session becomes active and starts sending packets, this session is not assigned the full bandwidth at once, but gradually. The name *"slow-start"* should indicate that the service rate of a newly active session is increased slowly when the session starts transmitting. As a result, all sessions that have been active see their service rates decreased smoothly.

The remainder of this extended abstract is structured as follows. In Section 2 we review GPS and demonstrate the problem of drastic delay increases in GPS. In Section 3 we present the new Slow-Start GPS scheduler that alleviates this problem. In Section 3 we present an analytical result on the worst-case delays in S^2GPS. In Section 4 we define the packetized version of S^2GPS and show how it can be implemented using the concept of virtual time. We present conclusions in Section 5.

2 The Problem: Abrupt Delay Increases in GPS

A GPS scheduler [7] is a work-conserving scheduler that serves all incoming arrivals at a fixed rate r. Each session i is characterized by a weight ϕ_i. Let $S_i(\tau, t)$ be the processing time given to session i in the interval $(\tau, t]$. Then, a GPS scheduler is defined as one for which:

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{\phi_i}{\phi_j}$$

for any pair of sessions i and j that are active throughout the interval $(\tau, t]$. If $B(t)$ is the set of active sessions at time t, then every session $i \in B(t)$ is served at the instantaneous service rate of:

$$r_i(t) = \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} r \tag{1}$$

We refer to $r_i(t)$ as the *fair share* of session i at time t. A session i is guaranteed a minimum service rate of

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r \tag{2}$$

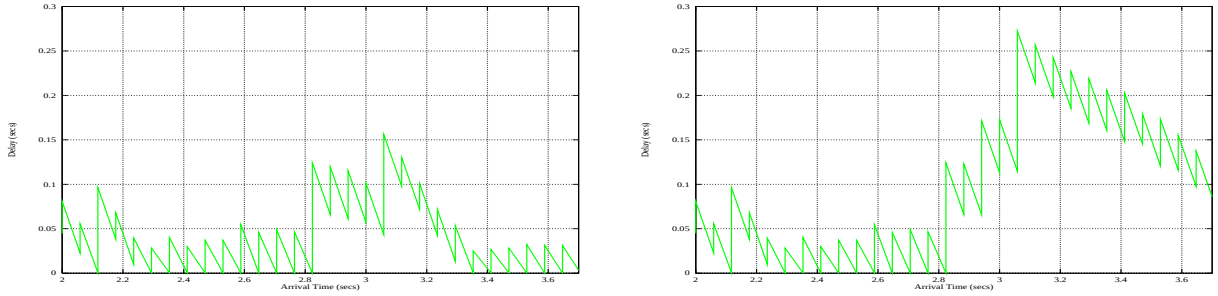
where N is the maximum number of sessions that are being serviced by the GPS scheduler.

GPS is an idealized scheduler in that it works under the assumption that all workload is infinitely divisible and that all backlogged sessions can be served simultaneously. Since a packet scheduler can transmit only one packet at a time, approximation techniques are needed that emulate the idealized GPS system. Several approximations of GPS have been proposed recently, e.g., [1, 3, 4, 5, 7, 12, 9, 10, 6, 13, 14]. An advantage of scheduling method derived from GPS is that (within certain restrictions) end-to-end delay bounds can be calculated with relative ease [8].

A problem with GPS: The following example will demonstrate how the service rate of a session can decrease under GPS due to the activity of other sessions, and how, as a result, the delay can increase abruptly. Suppose that we have a video transmission system over an ATM Permanent Virtual Connection (PVC) with a bandwidth of 1Mbps. On this PVC, three MPEG movie transmissions are multiplexed using PGPS scheduling, a packet-by-packet version of the GPS scheduling method [7]. The movie transmissions result from two software-compressed MPEG traces [**?**] (Sessions 1 and 2 send identical streams). Sessions 1 and 2 have a worst-case delay bound of 400 ms and session 3 has a worst-case delay of 200 ms. Sessions 1 and 2 start transmitting at time $t = 0$ sec.

Figure 1(a) depicts the delay of Sessions 1 and 2 without transmission from Session 2. In Figure 1(b) we assume that Session 2 starts transmitting at time $t = 3$ sec. Note in Figure 1(a) that even without Session 2 the burstiness of the MPEG traces results in abrupt delay increases; For example, the delay increase at time $t = 2.8$ *sec* is only due to burstiness of the MPEG traces.

However, the delay increase caused by the bursty nature of MPEG is much smaller than the delay caused by the arrival of Session 3 at time . From Figure 1(b) we can see that when session 3 starts transmitting at $t = 2.8\ sec$. Here, the delay of Sessions 1 and 2 is increased to over 250 ms, corresponding to a 66% increase of delay. This example shows that even with very bursty types of traffic (like MPEG), the abrupt delay increase when new sessions start transmitting can be dramatic. This problem is resolved by the S^{2}GPS scheduler presented in the next section.



(a) Only Session 1 and 2 transmit. (b) All sessions (1, 2, and 3) transmit.

Figure 1: Abrupt Increase of Delay

3 Slow-Start GPS (S^2GPS)

In this section we propose a modification to the GPS scheduling method that prevents abrupt delay increases such as the one shown in the previous section. The delay increases are a direct result of Equation (1) for calculating the service rate $r_k(t)$ for a session k. Whenever the set of active sessions, $B(t)$, changes, the service rate $r_k(t)$ of a session k is changed also. Therefore, when a new session becomes active, i.e., the set $B(t)$ grows, the service rate of all active sessions is immediately reduced.

As a solution to mitigate this effect, we present the *Slow-Start GPS* (S^{2}GPS) scheduling method. In S^{2}GPS, whenever a session becomes active, its service rate is increased linearly over an interval of length T. Thus, when a 'new' session becomes active, the service rate of the 'old' sessions decrease linearly.

For each session k, $T > 0$ specifies the amount of time that has to pass before session k is assigned its fair share of the bandwidth.

If a session k becomes active at τ_k, the S^{2}GPS scheduler increases its service rate linearly in the interval $[\tau_k, \tau_k + T]$. At time $\tau_k + T$, the session is assigned the same share of bandwidth that is allocated by the GPS scheduler (and given in Equation (1)). If we use $\hat{r}_k(t)$ to denote the service rate for session k that is allocated by the S^{2}GPS scheduler at time t, we have:

$$\hat{r}_k(t) = \min\{r_k(t), \frac{t - \tau_k}{T} r_k(t)\} \qquad : t \geq \tau_k \tag{3}$$

where $r_k(t)$ is the service rate allocated to session k under GPS.

The rate increase of session k in the time interval $[\tau_k, \tau_k + T]$ can be viewed as a *slow-start phase* of this session. At time $t = T + \tau_k$, the slow-start phase for session k is over, and session k will be assigned its full fair share of the bandwidth.

In Figure 2 we illustrate the difference between the rate allocation of GPS and S^{2}GPS. The figure depicts the service rate of a session k as a function of time. We assume that session k

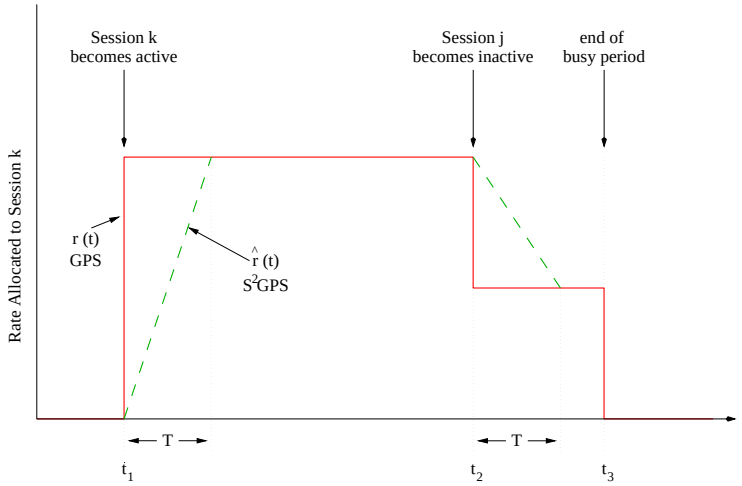


Figure 2: Service rate of a session as a function of time in GPS and S^{2}GPS.

becomes active at time t_1 and that some previously active session $j \neq k$ becomes inactive at time t_2. Under GPS, the service rate $r_i(t)$ is changed immediately whenever the set of active sessions is changed. Under S^{2}GPS, the service rate adapts slowly with a linear slope. Note that the slope of the rate increase after time τ_1 is completely determined by T and $r_k(t)$; However, the slope of the rate decrease after time t_2 additionally depends on $B(t)$, the set of active sessions.

4 Analysis of S^{2}GPS

The slow-start phase introduced by S^{2}GPS can increase the worst-case delays of a sessions, if compared to a GPS scheduler. To gain insights into the delay performance of the S^{2}GPS we have derived analytical worst-case delay bounds. Similar to [7], we assume that the traffic of a session is constrained by a leaky bucket. That is, for each session k we have a pair of parameters (σ_k, ρ_k) and the maximum traffic that the session can submit to the scheduler is constrained by $A_k^*(t) = \sigma_k + \rho_k t$.

The derivation of the worst-case delay bounds too complex to be presented here. A complete derivation is available in [11]. Here, we only present delay bounds in the following theorem.

Theorem 1 *In a S^{2}GPS scheduler the worst-case delay δ_k for a leaky bucket constrained session k with parameters (σ_k, ρ_k) and with guaranteed rate g_k is given by*

$$\delta_k = \begin{cases} \sqrt{\frac{2T\sigma_k}{g_k}} & : \quad \sqrt{\frac{2T\sigma_k}{g_k}} \leq T \quad and \quad \frac{\rho_k T}{2g_k} - \frac{\sigma_k}{\rho_k} \leq 0 \\ \sqrt{\frac{2T(\sigma_k + \rho_k \frac{1}{\rho_k}(\frac{T^3}{2g_k} - \sigma_k))}{g_k}} - \frac{1}{\rho_k}(\frac{T^3}{2g_k} - \sigma_k) & : \quad \frac{\rho_k T}{2g_k} - \frac{\sigma_k}{\rho_k} \geq T \quad and \quad 0 \leq \frac{1}{\rho_k}(\frac{T^3}{2g_k} - \sigma_k) \leq T \\ \sqrt{\frac{2T(\sigma_k + \rho_k \frac{\rho_k T}{2g_k} - \frac{\sigma_k}{\rho_k})}{g_k}} - \frac{\rho_k T}{2g_k} - \frac{\sigma_k}{\rho_k} & : \quad 0 \leq \frac{1}{\rho_k}(\frac{T^3}{2g_k} - \sigma_k) \leq T \quad and \quad \sqrt{\frac{2T(\sigma_k + \rho_k \frac{\rho_k T}{2g_k} - \frac{\sigma_k}{\rho_k})}{g_k}} \leq T \\ \frac{1}{g_k}(\sigma_k + \frac{g_k T}{2}) & : \quad \frac{1}{g_k}(\sigma_k + \frac{g_k T}{2}) \geq T \\ \frac{\rho_k - g_k}{g_k}\frac{1}{\rho_k}(\frac{g_k T}{2} - \sigma_k) + \frac{1}{g_k}(\sigma_k + \frac{g_k T}{2}) & : \quad \frac{1}{g_k}(\sigma_k + \frac{g_k T}{2}) < T \quad and \quad 0 \leq \frac{1}{\rho_k}(\frac{g_k T}{2} - \sigma_k) \leq T \\ \frac{\rho_k - g_k}{g_k} T + \frac{1}{g_k}(\sigma_k + \frac{g_k T}{2}) & : \quad \frac{\rho_k - g_k}{g_k} T + \frac{1}{g_k}(\sigma_k + \frac{g_k T}{2}) \geq 0 \\ 0 & : \quad otherwise \end{cases} \quad (4)$$

5 S^{2}PGPS: A Packet-by-packet Version of S^{2}GPS

Similar to GPS, S^{2}GPS assumes a fluid model where traffic is infinitely divisible. In real life, however, a scheduler can only serve one packet at a time. Analogous to the work in [7], we define a packet approximation of S^{2}GPS, called *Slow-Start Packetized GPS* or S^{2}PGPS. We can show that S^{2}GPS can closely approximate the fluid model. In fact we have shown that a S^{2}PGPS system cannot fall behind the corresponding S^{2}GPS system by more than the transmission time of a packet with maximum size.

We have devised an algorithm that implements S^{2}GPS using the concept of 'virtual time', a concept that was proposed in [4, 7] for implementing GPS systems. In [4, 7], the virtual time $V(t) = \int_{t_1}^{t_2} \frac{d\tau}{\sum_{i \in B(t)} \phi_i}$ is used as a measure of progress in the system in the time interval $[\tau_1, \tau_2)$. When a packet arrives, the scheduler assigns it a *'virtual finishing time'* and then serves packets in increasing order of the virtual finishing times. For the p-th packet of the k-th session, the virtual start time $S_i^{(p)}$ and virtual finish time $F_i^{(p)}$ are defined as [7]:

$$S_i^{(p)} = \max\{F_i^{(p-1)}, V(t_i^{(p)})\} \tag{5}$$

$$F_i^{(p)} = S_i^{(p)} + \frac{L_i^{(p)}}{\phi_i} \tag{6}$$

where $t_i^{(p)}$ is the arrival time of packet p and $L_i^{(p)}$ is the length of packet p.

The difficulty of finding a packet-by-packet version of S^{2}GPS, as compared to the original GPS and PGPS systems, is that the weight ϕ_k of a session k is not constant if session k is in the slow-start phase. We have solved this problem by assigning to sessions that are in a slow-start phase an "effective" weight $\phi_{eff}^{(p)}$ and set the virtual finishing time equal to $S_k^{(p)} + \frac{L_k^{(p)}}{\phi_{eff}^{(p)}}$. Then we can use Equation (6) to calculate the virtual finishing times. The effective weight $\phi_{eff}^{(p)}$ can be interpreted as the average weight of the pth packet, averaged over the transmission time of the pth packet.

If we set $\phi_{eff}^{(p)}$ to:

$$\phi_{eff}^{(p)} = \frac{L_k^{(p)} \sum_{i=1, i \neq k}^{N} \phi_i}{r(w_p - w_{p-1}) - L_k^{(p)}} \tag{7}$$

where w_p is an auxiliary variable given iteratively by (with $w_{-1} = 0$):

$$w_p = \sqrt{\frac{2L_k^{(p)} \sum_{i=1}^{N} \phi_i}{r\phi_k} + (w_{p-1})^2} \tag{8}$$

we can prove the following theorem [11] which is the analogue to the result in [7] for the relation on GPS and PGPS.

Theorem 2 *Let $d_{k,S^2PGPS}^{(p)}$ and $d_{k,S^2GPS}^{(p)}$ denote the departure times of the pth packet of session k under S^{2}GPS and S^{2}PGPS. Then we have*

$$d_{k,S^2PGPS}^{(p)} - d_{k,S^2GPS}^{(p)} \leq \frac{L_{max}}{r} \qquad \forall p, k \tag{9}$$

In other words, a S^2PGPS system cannot fall behind a S^2GPS system by more than one maximum packet size.

We have performed simulation experiments [11] that demonstrate how well S^2GPS reduces the rate at which delays can change in a GPS system. Due to space limitations, the results of the simulations are not included in this abstract. If accepted, we will show the simulation results at the workshop.

References

[1] J.C.R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *ACM Sigcomm '96*, August 1996.

[2] J.C.R. Bennett and H. Zhang. WF2Q: Worst-case Fair Weighted Fair Queueing. In *Proc. IEEE Infocom '96*, March 1996.

[3] J. Davin and A. Heybey. A simulation study of fair queueing and policy enforcement. *Computer Communication Review*, 20(5):23–29, October 1990.

[4] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proc. Sigcomm '89*, pages 1–12, 1989.

[5] S. J. Golestani. A self-clocked fair queueing scheme for broandband applications. In *Proc. IEEE Infocom '94*, pages 636–646, 1994.

[6] P. Goyal, H. M. Vin, and H. Cheng. Start-time Fair Queueing: A Scheduling Algorithm for Intergrated Services Packet Switching Networks. In *ACM Sigcomm'96*, pages 157–168, 1996.

[7] 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.

[8] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Intergrated Services Networks: The Multiple Node Case. *IEEE/ACM Transanctions on Networking*, 2:137–150, April 1994.

[9] D. Saha, S. Mukherjee, and S. H. Tripahi. Carry-Over Round Robin: A Simle Cell Scheduling Mechanism for ATM Networks. In *IEEE Infocom'96*, pages 630–637, 1996.

[10] S. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. *IEEE Transanctions on Networking*, 4(3):375–385, June 1996.

[11] A. Stamoulis and J. Liebeherr. S^2GPS: Slow-Start Generalized Processor Sharing. Technical Report TR96-xx, University of Virginia, 1996. In preparation.

[12] D. Stiliadis and A. Varma. Design and Analysis of Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks. In *ACM Sigmetrics '96*, pages 104–115, May 1996.

[13] O. Yaron and M. Sidi. Generalized Processor Sharing Networks with Exponentially Bounded Burstiness Arrivals. *Journal of High Speed Networks*, 3:375–387, 1994.

[14] Z.L. Zhang, D. Towsley, and J. Kurose. Statistical Analysis of Generalized Processor Sharing Scheduling Discipline. In *ACM Sigcomm'94*, pages 68–77, August 1994.