Technical Report No. CS-97-03, Computer Science Department, University of Virginia, February 1997.

S^2GPS: Slow-Start Generalized Processor Sharing [*]

Anastasios Stamoulis *Jörg Liebeherr*

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
{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 can have significant impact on the service provided by GPS, however, which has been given little attention. 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 decreases of service rates when new sessions start transmitting. S^2GPS is a modification of GPS where a session does not receive its guaranteed service rate immediately after it becomes active. Instead, the service rate of a session is gradually increased. We 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 scheduling discipline.

[*]This work is supported in part by the National Science Foundation under Grant No. NCR-9309224 and NCR-9624106.

1 Introduction

The statistical multiplexing nature of packet-switched networks with Quality-of-Service (QoS) guarantees sheds light on the importance of the following two desirable properties of networks: *isolation* and *sharing*. *Isolation* is directly related to the ability of the network to provide to sessions the performance guarantees that were made based on a QoS contract and, at the same time, protecting them from faulty or malicious sessions that do not conform to this contract. *Sharing* is related to the ability of the network to distribute its resources (mainly bandwidth and buffer space) to active sessions in such a way that the percentage of unused resources is minimal. The property of sharing is one of the fundamental advantages of packet-switched networks over circuit-switched networks, since it enables switches to achieve a high utilization of link capacities.

The *Generalized Processor Sharing* (GPS) scheduling method is known to support isolation and sharing in a QoS network [8, 16, 17]. GPS can provide rate guarantees to the sessions it services. However, with GPS, a session that has been active for a long period of time can experience dramatic decreases 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. Such a delay increase can have a negative impact on the jitter[1] and delay rate of the sessions.

In this study we show how to alleviate the problem of abrupt decrease of service rates with GPS. We propose a modification to GPS, called Slow-Start GPS (S^2GPS), that prevents abrupt rate changes 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"* was elected to indicate that the service rate of a newly active session is slowly increased when the session starts transmitting. As a result, the service rates of previously active sessions decrease smoothly.

In more detail, we define as *fair share* of a session the service rate that a session receives under GPS. Under S^2GPS, when a session starts transmitting, it undergoes a *slow-start phase* during which the service rate of the session is increased from zero to the fair share. During the slow-start phase, the service rates of all previously active sessions are decreased. As a result, the delay increases and delay variations of the old sessions will be reduced. We refer to the slow adaptation of active sessions as *graceful degradation*. Our definition of graceful degradation refers to the way in which the sessions experience the decrease of the service rates and the bandwidth that is allocated to them.

The advantage of S^2GPS can be exploited for several network services and applications. We give two examples that discuss how S^2GPS can be used to support congestion control schemes and adaptive applications.

<u>Congestion Control:</u> In networks that employ GPS scheduling, a new session can immediately transmit at the maximum allowed transmission rate. This abrupt change of the total transmission may cause buffer overflows and result in congestion conditions. Under S^2GPS, the extra period of time given by

[1] Jitter is defined as the delay difference between two consecutive packets.

the length of the slow-start phase allows all sessions to adjust to their new shares of the transmission rates, and hence, avoid congestion. Note that the name of our scheduler borrows from the well-known concept of "slow-start" in TCP congestion control [12]. Our mechanism of slow-start is different from slow-start in TCP [12], but similar in spirit; in both cases the goal is to avoid congestion by preventing abrupt changes in the transmission rate of a session. Note, however, that our proposed slow-start technique is used at the scheduler level and not at the transport layer as in TCP.

Adaptive Applications: It is generally accepted that adaptive applications, which dynamically adjust their performance parameters based on bandwidth availability or delay of incoming packets, will play an indispensable role in the integrated services network infrastructure of the future [1, 5, 4]. The difficulty of supporting adaptive applications relates to sudden changes in the bandwidth availability, which cause large fluctuations in the network delay. An adaptive application must handle this situation either by buffering or by adjusting properly its playback rate. If the delay fluctuations are too large, temporary breaks in the service as it is perceived by the client may occur. S^2GPS helps to alleviate this problem by smoothing out delay fluctuations, hence, improving the service of the network.

The remainder of this paper is structured as follows. In Section 2 we review related work on GPS and in Section 3 we discuss GPS and its packetized version, PGPS. In Section 4 we study a class of scheduling disciplines that alleviate the problem of abrupt degradation of service and we present the novel S^2GPS scheduler. In Section 5 we analyze the worst-case delays with S^2GPS. In Section 6 we define the packetized version of S^2GPS and show how it can be implemented using the concept of virtual time. In Section 7 we test our scheduler in a set of simulation experiments. We present conclusions in Section 8.

2 Related Work

GPS and its variations are widely considered well-suited scheduling methods for integrated services networks. The basic reasons for the popularity of GPS are the *isolation* (or "protection") and *fairness* that GPS provides. The isolation property refers to the guarantee that sessions which transmit at rates greater than their service rates will not be served at the expense of other sessions; hence, sessions appear to be isolated. The fairness property refers to the guarantee that a session will always be served at a minimum service rate.

GPS is an idealized server in that it works under the assumption that all workload is infinitely divisible and that all backlogged sessions can be served simultaneously. Since in real networks, packets have a finite size, and only one packet can be transmitted at a time, approximations of GPS are needed that emulate the idealized GPS discipline. Weighted Fair Queueing (WFQ) [8] and Packetized GPS (PGPS) [16] have approximated GPS using the concept of virtual time to measure the progress in the fluid model. Each packet is tagged with a virtual time deadline upon its arrival, and packets are served in increasing order of their deadlines. In [16, 17], worst-case delay bounds are provided for single node and multiple node networks. In [23, 24], GPS is studied from a probabilistic point of view and statistical bounds on backlog and delay are derived. In [3], it is shown that PGPS is not the best packet approximation of GPS because a session in PGPS can be far ahead of GPS in terms of served packets.

A new approximation of GPS called Worst-case Weighted Fair Queuing (WF^{2}Q) is proposed which mitigates this problem. Under WF^{2}Q, the absolute difference between the amount of service that a session receives between GPS and WF^{2}Q cannot be more than one maximum length packet size.

One of the problems associated with PGPS is that keeping track of virtual time can be computationally expensive which makes it difficult to implement PGPS at high speeds. Self-Clocked Fair Queuing (SCFQ) [9] and Start-time Fair Queuing (SFQ) [11] address this issue by providing a different way of calculating the virtual time in GPS. Under these schemes, the scheduler does not keep track of the virtual time using a computationally expensive function. Instead, the virtual time is considered to be the virtual time of the packet under transmission: SCFQ transmits packets in increasing order of their virtual finishing times, whereas SFQ uses virtual start times.

One other approach to approximating GPS is followed by Frame-based Fair Queuing (FFQ) [21], Deficit Round Robin (DRR) [20], and Carry-over Round Robin (CRR) [19]. FFQ, DRR, and CRR provide more efficient implementations than PGPS. Central in these schedulers is the notion that during a frame (or round) the scheduler attempts to approximate GPS. When a session is not served at the rate specified by GPS during a frame, the scheduler attempts to compensate for this during the next frame. However, this comes at the expense of fairness and the increase of worst-case delays because of the error that the approximation of GPS introduces. Similar in spirit is the Fair Queuing Fixed Quota (FQFQ) scheduling discipline [7], where each session is allocated a fixed amount of buffer space that is available in the switch; if a session occupies all its buffer space, then all arriving packets of this session are discarded. In [14], a probabilistic variant of fairness queuing called Stochastic Fairness Queuing is proposed; this is based on the idea that a hash function will be used to map every session into a fixed set of queues.

In [2, 15], hierarchical implementations of GPS are proposed. In [15], the implementation is based on a GPS scheduler, where the weights of the sessions are dynamically changed. In [2], Hierarchical Worst-Case Weighted Fair Queuing is proposed (H-WF^{2}Q+); this scheduling discipline is a hierarchical expansion of multiple one-level GPS servers.

Finally, given this proliferation of GPS variants, in [10, 22], a general framework that attempts to encompass these variants is proposed. In [10], this framework takes the form of a generalized version of GPS, under which variable service rates can be allocated to the packets of a session. In [22], the concept of a Latency-Rate scheduler is introduced. A latency-rate server provides rate guarantees to a session after some time from the time that the session starts transmitting; it is shown how GPS and its variants can be modeled as latency-rate schedulers.

3 GPS/PGPS Scheduler

In this section we briefly outline the basic mechanisms and results for the GPS scheduling discipline and we discuss the rate and delay guarantees that GPS provides. We also illustrate the problem of abrupt decrease of service rates when new sessions start transmitting.

3.1 GPS Server

A Generalized Processor Sharing (GPS) scheduler is a work-conserving[2] scheduler that serves the incoming traffic at a fixed rate r. Each Session i that is served by the scheduler is characterized by a weight ϕ_i. Let $S_i(\tau,t)$ be the amount of Session i traffic that is served in the interval $(\tau,t]$. Define a session as *backlogged* whenever the queue Q_i of this session is not zero. Then, a GPS scheduler is defined as one for which:

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j} \tag{1}$$

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

$$r_i(t) = \frac{\phi_i}{\sum\limits_{j\in B(t)} \phi_j} r \tag{2}$$

Therefore, a Session i is guaranteed a minimum service rate of g_i for any time interval that it is backlogged:

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r \tag{3}$$

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

Note here that GPS is an idealized scheduler that does not transmit packets as individual entities. It works under the assumption that traffic is infinitely divisible; hence, it can serve all backlogged sessions simultaneously. However, in reality, only one session can receive service at a time, and a packet has to be fully transmitted before another packet starts being served. Perhaps the most popular method to approximate GPS in a packet system is Packet-By-Packet Generalized Processor Sharing (PGPS) [16] which is defined as follows. Let $d_{i,gps}^{(p)}$ be the departure time of a packet p from Session i under GPS. Then, PGPS is the service discipline that transmits packets in increasing order of $d_{gps}^{(p)}$'s.

In [16], it is proved that:

$$d_{i,PGPS}^{(k)} - d_{i,GPS}^{(k)} \leq \frac{L_{max}}{r} \qquad \forall i,k \tag{4}$$

$$S_{i,GPS}(0,t) - S_{i,PGPS}(0,t) \leq L_{max} \qquad \forall i,t \tag{5}$$

where $d_{i,PGPS}^{(k)}$, $d_{i,GPS}^{(k)}$ are the departure times of the k-th packet of Session i under GPS and PGPS, and $S_{i,GPS}(0,t)$, $S_{i,PGPS}(0,t)$ are the total amounts of service received by Session i in time interval $(0,t]$ under GPS and PGPS, respectively. In other words, a PGPS system cannot fall behind a GPS system by more than one maximum packet size.

In [16], it was proved that for leaky bucket constrained sessions, GPS guarantees deterministic worst-case delays. Specifically, let $A_i(\tau,t)$ be the amount of traffic of Session i that arrives in the time interval $(\tau,t]$. A *leaky bucket* [6] is defined as a pair (σ,ρ) where σ is a burst factor and ρ is a rate factor. We say that a Session i is (σ_i,ρ_i)-constrained if: $A_i(\tau,t) \leq \sigma_i + \rho_i(t-\tau)$, for all $t \geq \tau$. A session is called *greedy* if it always transmits at its maximum allowable data rate. If Session i is leaky-bucket

[2]A scheduler is work-conserving if it is not idle if there is incoming traffic to be transmitted. Otherwise, it is non-work-conserving.

constrained and greedy, then we have: $A_i(\tau, t) = \sigma_i + \rho_i(t - \tau)$. Under GPS, one easily derived delay bound for a (σ_i, ρ_i)-constrained Session i is $\frac{\sigma_i}{g_i}$ provided that $g_i \geq \rho_i$. The isolation property of GPS makes it possible to derive worst-case delay bounds for a session without taking into consideration the behavior of other sessions. If the behavior of other sessions is taken into consideration, the exact worst-case delay bound can be calculated as shown in [16]. The delays depend on the traffic characterization of all sessions and the assignment of ϕ's. Note that the exact worst-case delay bound can be quite smaller than $\frac{\sigma_i}{g_i}$. However, the delay bounds that are calculated using the guaranteed service rate g_i are independent of the traffic characterizations and arrival patterns of the other sessions. We will take advantage of this property of GPS, when we calculate the worst-case delay for S^{2}GPS schedulers.

3.2 GPS and abrupt decrease of service rates

When new sessions start transmitting packets, the service rates of the previously active sessions decrease abruptly. This abrupt decrease, which in some cases can be dramatic, can result in abrupt increases of delay and jitter. The decrease of service rates is a direct result of the fact that, under GPS, the service rate that a session receives is dependent on the number of backlogged sessions (as (2) indicates). However, GPS is considered to provide isolation to the sessions that it services. As sessions are considered to be isolated, it would be expected that the behavior of other sessions should not dramatically affect the service rate of a session. Contrary to this expectation we show that essentially the principle of isolation is violated in GPS; a session can experience an abrupt decrease of its service rate because of the behavior of the other sessions.

Let us present an example that illustrates how the service rate of a session under GPS can decrease rapidly. Suppose that we have a switch that operates at 45 Mbps. Further suppose that the switch serves five sessions. All sessions have the same weights, i.e., $\phi_0 = \phi_1 = \phi_2 = \phi_3 = \phi_4$. The guaranteed rate for every session is 9 Mbps. Let us assume that Session 0 becomes active at time $t = 0$ sec, Session 1 at $t = 1$ sec, …, Session 4 at time 4sec.

In Figure 1 we plot the bandwidth that is available to Session 1 in such a scenario.[3] As the figure indicates, when Session 1 starts transmitting at time $t = 1$ sec, it immediately obtains its fair share of the bandwidth, which is 22.5 Mbps. When Sessions 2, 3, and 4 start transmitting, the available bandwidth for Session 1 is decreased rapidly to 15 Mbps, 11.25 Mbps and 9 Mbps, respectively. The figure clearly shows that GPS abruptly changes the service rates of a session whenever a new session becomes active.[4]

4 Slow-Start GPS Schedulers (S^2GPS)

The problem of abrupt changes of the service rate available to sessions is a direct result of the dependence of the service rate on the number of backlogged sessions and their weights, ϕ's. In order to alleviate

[3]The figure depicts the results for PGPS, a packetized version of GPS (see Section 6). Note that the fluctuations of the service rate are caused by the fact that PGPS is an approximation of GPS. As a result, the total available bandwidth oscillates.

[4]In Figure 7 of Section 7 we show how our slow-start mechanism for *GPS* smoothes the changes of the service rate.

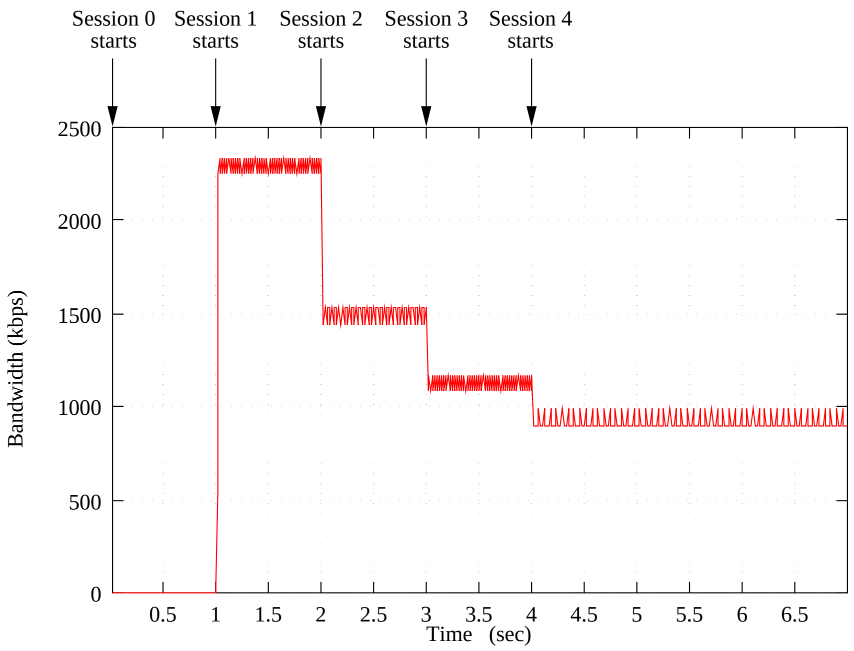


Figure 1: Available bandwidth for Session 1 under GPS.

this problem we propose a modification to the GPS servicing discipline that aims to provide smoother, i.e., graceful, decrease of the service rates of all backlogged sessions. In our scheme the service rate of a session cannot decrease abruptly. It is important to note that our proposal modification will provide long-term behavior equivalent to a GPS system, i.e., the long term service rates that a session receives under GPS and a modified-GPS should be similar.

In GPS, changes in the service rates occur when the set of the backlogged sessions changes. This happens when either "new" sessions become active or some sessions cease to be backlogged. When "new" sessions become backlogged, they demand their share of the bandwidth. This results in the decrease of the service rates of the "old" sessions and a potential increase of delays. On the other hand, when some sessions are no longer backlogged, the service rates of all other sessions have to be increased abruptly. In this paper we are only interested in the first case. Handling of the second case can be done straightforwadly by making the scheduler non-work-conserving. Specifically, if service rate increases are to be avoided we simply delay packets on purpose. The scheduler will be non-work-conserving, since no packet is transmitted when all packets in the queue have to be delayed.

Our approach is based on the following basic idea: when a new Session k with ϕ_k becomes active at time instant t_k (We use t_k to denote the time instant when Session k becomes active after an idle period), then the weight of this session will be gradually increased from an initial value of $\phi = 0$ to its final value, ϕ_k. Since the new session will not receive at once its fair, we anticipate that the service rates of all other sessions will not drop dramatically. In other words, a slow-start scheduler does not assign to a new session its fair share of the bandwidth as soon as it becomes active. Instead, the service rate is increased gradually until it reaches the service rate which is given by (2). Correspondingly, the service rates of the previously active sessions decrease in a smooth fashion.

In a slow-start GPS scheduler, for each new Session k, we use $T_k > 0$ to specify the length of the slow-start period, that is, the amount of time that has to pass before Session k is assigned its fair share of the bandwidth. If Session k becomes active at t_k, its service rate is increased in the interval $[t_k, t_k + T_k]$, and at time $t_k + T_k$, the session has obtained its fair share of the bandwidth. Let us denote the instantaneous service rates as $\hat{r}_k(t)$. A slow-start GPS scheduler is a work-conserving scheduler that maintains two sets of sessions, $B(t)$ and $B_{new}(t)$. $B(t)$ is the set of active sessions at time t as defined in Subsection 3.1, and $B_{new}(t) = \{k | \hat{r}_k(t) < r_k(t)\}$ is the set of all newly active sessions that have not yet acquired their fair share of the bandwidth at time t.

The slow-start GPS scheduler is characterized by the following properties:

1. The service rate of a newly active session in the slow-start phase is an increasing function in time. Thus,
$$0 \leq \hat{r}_k(t) \leq \hat{r}_k(t + \Delta t) \leq r_i(t + T_k) \qquad : \ t_k \leq t < t + \Delta t \leq t_k + T_k$$

2. If Session k is backlogged throughout the interval $[t_k, t_k + T_k]$, then after time $t_k + T_k$, Session k is served at a rate at least as large as the service rate under GPS. Note that $\hat{r}_k(t)$ can be greater than $r_k(t)$ for $t \geq t_k + T_k$. This will happen when some session (other than k) is in the slow-start phase. Thus,
$$\hat{r}_k(t) \geq r_k(t) \qquad : \ t \geq t_k + T_k$$

3. For any two connections i and j in $B(t) - B_{new}(t)$, we have that:
$$\frac{\hat{r}_i(t)}{\hat{r}_j(t)} = \frac{\phi_i}{\phi_j}, \qquad : \ \forall t$$

In this paper we investigate a slow-start GPS scheduler where the increase of service rates is carried out linearly with respect to time. Also, we assume that the length of the slow start period is identical for all sessions, that is, $T_k = T$, $\forall k$. Then the service rate of a Session $k \in B_{new}(t)$ that becomes active at time t_k and is continuously backlogged in the interval $[t_k, t_k + T]$ is given by:
$$\hat{r}_k(t) = \frac{t - t_k}{T} r(t) \qquad : \ t_k \leq t \leq T + t_k \tag{6}$$
At time $t = T + t_k$, k is removed from $B_{new}(t)$ because it will have been assigned its fair share of the bandwidth. We refer to this scheduler as Slow-Start GPS (S^{2}GPS).

In Figure 2 we illustrate the difference between GPS and S^{2}GPS. This figure depicts the service rate of Session k as a function of time. The figure shows three events: At time t_k, Session k becomes active, at time t_j, Session j becomes active and at time t_x Session k becomes idle. Under GPS, the service rate function $r_k(t)$ for a Session k consists of linear horizontal segments. Under S^{2}GPS the service rate function does not change abruptly at the points in time where a session becomes active or a session leaves the system. At these points, the service rate changes in a linear fashion as Figure 2 illustrates. Under GPS, when a session becomes active, it immediately receives service at a rate given by (2); when a new session arrives, the service rate of the "old" sessions drop abruptly. Under S^{2}GPS, when a session becomes active, its service rate is increased linearly in an interval of length T. Thus, when a "new" session becomes active, the service rate of the "old" sessions decrease linearly.

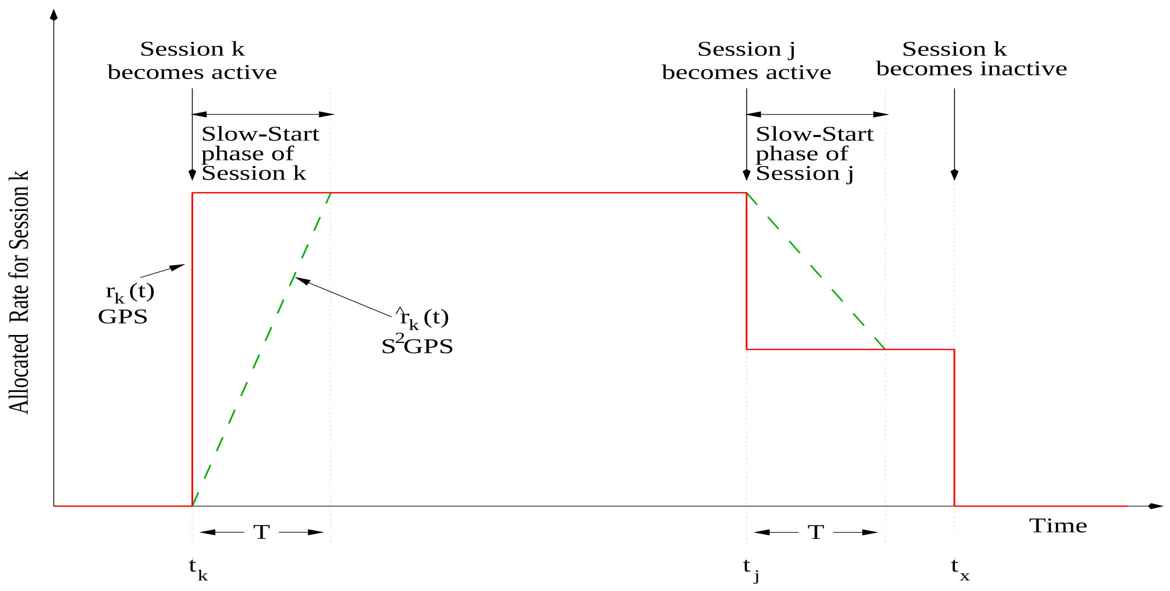


Figure 2: Service rate of Session k as a function of time in GPS and S^{2}GPS.

If $B(t)$ and $B_{new}(t)$ are constant in $[t_k, T + t_k]$, then a Session $k \in B_{new}(t)$ is served at the instantaneous rate of:

$$\hat{r}_k(t) = \frac{t - t_k}{T} \frac{\phi_k}{\sum\limits_{i \in B(t)} \phi_i} r \qquad : t_k \leq t \leq T + t_k \tag{7}$$

and a Session $j \in B(t) - B_{new}(t)$ is served at the instantaneous rate of:

$$\hat{r}_j(t) = (r - \sum_{k \in B_{new}(t)} \hat{r}_k(t)) \frac{\phi_j}{\sum\limits_{i \in B(t) - B_{new}(t)} \phi_i} \qquad : t_k \leq t \leq T + t_k \tag{8}$$

Equations (7) and (8) illustrate the fact that when a session is in the slow-start phase, the bandwidth, which corresponds to the difference between the fair share and the actual service rate, is distributed among the “old” sessions.

One can make the following observations about S^{2}GPS. First, equation (1) still holds for any two sessions that belong to $B(t) - B_{new}(t)$. Second, in the long term, when all sessions are active and their service rates have assumed their steady-state values, a S^{2}GPS system behaves exactly in the same way as a GPS system. Third, if the duration T of the slow-start phase is small relatively to the time that a session is active, then S^{2}GPS will not necessarily cause a dramatic increase to the average delay of the session. However, depending on the value of T, we expect the worst-case delay in a slow-start GPS scheduler to be larger than that in GPS. Fourth, the isolation property of GPS is further enhanced in S^{2}GPS. In GPS a session that does not transmit packets continuously (for example, a bursty source with long idle periods) can cause abrupt changes to the service rates of all the other sessions; though all sessions are guaranteed a minimum service rate, the actual service rates depend heavily on the behavior of all other sessions. Thus, the sessions are not perfectly isolated from each other. A slow-start GPS scheduler mitigates this problem and enhances the property of isolation.

5 Analysis of S^{2}GPS

The linear change of service rate in S^{2}GPS has primarily two effects. First, it increases the worst-case delay of new sessions that arrive in the system. Second, it decreases the delay rate for previously active sessions. In the following we derive bounds for the worst-case delay and the rate of delay changes for leaky bucket constrained sessions.

5.1 Worst Case Delay

In S^{2}GPS, when a previously idle session becomes active, it is assigned its fair share of the bandwidth only gradually. As a result, we expect the worst-case delay in a S^{2}GPS system to be larger than the worst-case delay in the corresponding GPS system. For sessions that have been assigned their fair share of the scheduler bandwidth, the worst-case delay in S^{2}GPS system is expected to be the same as in a GPS system, as the session will receive service at a minimum rate of g_i. Therefore, the difference between the worst-case delays in S^{2}GPS and GPS system for a Session i is obtained by evaluating the difference of service that the session receives until it is assigned its fair share of the bandwidth. In the following we will derive the worst-case delay bound for sessions that are constrained by leaky buckets.

In our analysis we take advantage of the so-called isolation property of GPS (which is also retained in S^{2}GPS). For the calculation of the worst-case delay, we consider a S^{2}GPS system where a maximum of N sessions can be admitted. Without loss of generality, we will calculate the worst-case delay for the k-th session that starts transmitting at time 0. As in GPS, in S^{2}GPS, Session k experiences its worst-case delay when:

1. All the other sessions in the system are continuously backlogged, **and**
2. Session k is greedy, i.e., $A_k(0,t) = \sigma_k + \rho_k t$.

The first condition keeps the maximum service rate attained by Session k to the minimum rate g_k. The second condition is required because the worst-case scenario will occur when a session transmits at its maximum allowable rate.

Let $Q_k(t)$ be the queue size (or backlog) of Session k at time instant t. Then we have:

$$Q_k(t) = A_k(0,t) - S_k(0,t) \tag{9}$$

Let $\delta_k(t)$ be the delay of each arrival at time t. Then we have:

$$Q_k(t) = S_k(t, t+\delta_k(t)) \tag{10}$$

For ease of notation, we will use $A_k(t)$ for $A_k(0,t)$ and $S_k(t)$ for $S_k(0,t)$. Figure 3 depicts $A_k(t)$ and $S_k(t)$. As Figure 3 suggests, $\delta_k(t)$ is the horizontal distance between $A_k(t)$ and $S_k(t)$; $Q_k(t)$ is the vertical distance between $A_k(t)$ and $S_k(t)$.

Under S^{2}GPS, the service rate of Session k is given by:

$$r_k(t) = \begin{cases} \frac{t}{T} g_k & : \quad t \leq T \\ g_k & : \quad t > T \end{cases} \tag{11}$$

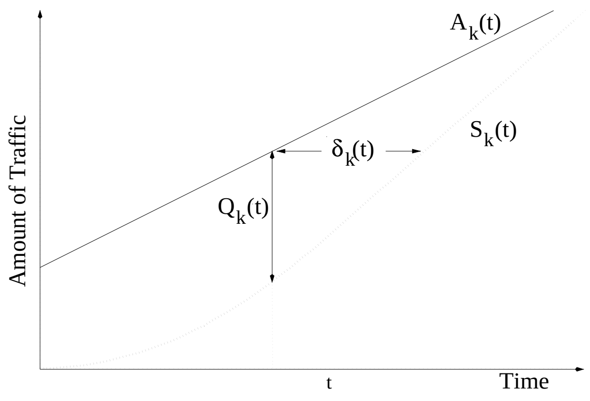


Figure 3: Relation of $\delta_k(t)$, $Q_k(t)$, $A_k(t)$, and $S_k(t)$.

The amount of traffic $S_k(t_1, t_2)$ that is served in the interval $[t_1, t_2]$ and the instantaneous service rate are associated through the equation:

$$S_k(t_1, t_2) = \int_{t_1}^{t_2} r_k(t)dt \tag{12}$$

From (10) and (12) we have:

$$Q_k(t) = \int_{t}^{t+\delta_k(t)} r_k(\tau)\, d\tau \tag{13}$$

From (9) and (13) we obtain:

$$S_k(0,t) + \int_{t}^{t+\delta_k(t)} r_k(\tau)d\tau = A_k(0,t)$$

As $S_k(0,t) = \int_0^t r_k(\tau)d\tau$:

$$\sigma_k + \rho_k t = \int_{0}^{t+\delta_k(t)} r_k(\tau)\, d\tau \tag{14}$$

Equation (14) will be used to evaluate the delay $\delta(t)$ as a function of time t. Hence, we are able to calculate the maximum delay $\delta_{\max}$, which is given in the following theorem.

Theorem 1 *The worst-case delay δ_k^{max} of the (σ_k, ρ_k)-constrained Session k is:*

$$\delta_k^{max} = \begin{cases} \frac{T}{2} + \frac{\sigma_k}{g_k} & : \quad T < \frac{2\sigma_k}{g_k} \\ \sqrt{\frac{2T\sigma_k}{g_k}} & : \quad \frac{2\sigma_k}{g_k} \leq T < \frac{2\sigma_k g_k}{\rho_k^2} \\ \frac{\sigma_k}{\rho_k} + \frac{\rho_k T}{2g_k} & : \quad \frac{2\sigma_k g_k}{\rho_k^2} \leq T \end{cases} \tag{15}$$

provided that $g_k \geq \rho_k$.

Proof: Without loss of generality, we assume that Session k starts transmitting at time 0. We assume that the worst-case delay of Session k is experienced by an arrival at time t. We use $\delta_k(t)$ to denote the delay of the traffic from Session k that arrives at time t. From (11) we have that the service rate $r_k(t)$ is time dependent. Hence, in order to use (14), we have to examine the three cases described in Table 1. For ease of notation, in the following we will drop the index k.

First, we proceed to examine the worst-case delay that corresponds to Case 1.

Case	arrival time	departure time
Case 1	$t < T$	$t + \delta(t) < T$
Case 2	$t < T$	$t + \delta(t) \geq T$
Case 3	$t \geq T$	$t + \delta(t) \geq T$

Table 1: Cases for the worst-case delay

Case 1: We have that $r(t) = \frac{t}{T} g$. From (14) we obtain:

$$\begin{aligned} \sigma + \rho t &= \int_0^{t+\delta(t)} \frac{\tau}{T} g \, d\tau \\ &= \frac{g}{T} \frac{(t+\delta(t))^2}{2} \end{aligned} \tag{16}$$

which gives:

$$\delta(t) = \sqrt{\frac{2T(\sigma + \rho t)}{g}} - t \tag{17}$$

As the departure must occur before time T, we need that $t + \delta(t) \leq T$ must hold. From (17) we obtain:

$$\sqrt{\frac{2T(\sigma + \rho t)}{g}} \leq T \tag{18}$$

As the left hand part of (18) is an increasing function of time t, we need to check what happens when $t = 0$. By setting $t = 0$ in (18), we obtain:

$$T \geq \frac{2\sigma}{g} \tag{19}$$

Hence, Case 1 is invalid if $\frac{2\sigma}{g} > T$.

In order to find the maximum value of $\delta(t)$ we will use the derivative $\delta'(t)$ which is given by:

$$\delta'(t) = \sqrt{\frac{T\rho^2}{2g(\sigma + \rho t)}} - 1 \tag{20}$$

Let us denote as t^* the time instant where $\delta'(t) = 0$. We find that:

$$t^* = \frac{\rho T}{2g} - \frac{\sigma}{\rho} \tag{21}$$

Depending on the values of T, σ, ρ and g, t^* can be on the "left", or "inside" of the interval $[0, T]$ (recall that $\rho \leq g$). Therefore we have to check the following cases:

(a) $t^* \leq 0$: As $\delta(t)$ is monotonously decreasing for $t \geq t^*$, we conclude that the maximum delay occurs at $t = 0$. Hence, in this case the maximum delay is given by:

$$\delta(0) = \sqrt{\frac{2T\sigma}{g}} \tag{22}$$

(b) $0 \leq t^* \leq T$**:** First, we need to check when $t^* \geq 0$ holds. From (21), we have:

$$\frac{\rho T}{2g} - \frac{\sigma}{\rho} \geq 0$$

which yields:

$$T \geq \frac{2\sigma g}{\rho^2} \tag{23}$$

In this case the maximum delay $\delta(t^*)$ is calculated from (17) and (21):

$$\delta(t^*) = \frac{\sigma}{\rho} + \frac{\rho T}{2g} \tag{24}$$

We still need to verify if this maximum delay corresponds to a departure that occurs before time T, i.e., we must show that $t^* + \delta(t^*) \leq T$. The proof that this inequality holds is technical and pushed in the Appendix in Lemma 1.

Combining (19), (22), (23), and (24) we obtain:

$$\delta_{max} = \begin{cases} \sqrt{\frac{2T\sigma}{g}} & : \quad \frac{2\sigma}{g} \leq T < \frac{2\sigma g}{\rho^2} \\ \frac{\sigma}{\rho} + \frac{\rho T}{2g} & : \quad T \geq \frac{2\sigma g}{\rho^2} \end{cases} \tag{25}$$

Case 2: From (11) and (14) we obtain:

$$\begin{aligned} \sigma + \rho t &= \int_0^{t+\delta(t)} r(\tau)\, d\tau \\ &= \int_0^T \frac{\tau}{T} g\, d\tau + \int_T^{t+\delta(t)} g\, d\tau \\ &= \frac{\frac{g}{T}T^2}{2} + (t + \delta(t) - T)g \end{aligned}$$

which yields:

$$\delta(t) = (\frac{T}{2} + \frac{\sigma}{g}) + \frac{\rho - g}{g} t \tag{26}$$

As $g \geq \rho$, $\delta(t)$ is a monotonously decreasing function of time and the maximum occurs at $t = 0$. Hence:

$$\delta_{max} = \delta(0) = \frac{T}{2} + \frac{\sigma}{g} \tag{27}$$

However, Case 2 assumes that the departure occurs at a time after time T. Therefore, we must have that $\delta(0) > T$. We can easily verify that this holds if and only if $T < \frac{2\sigma}{g}$. Hence the maximum delay for Case 2 is:

$$\delta_{max} = \frac{T}{2} + \frac{\sigma}{g}, \quad \text{if} \quad T < \frac{2\sigma}{g} \tag{28}$$

Case 3 In this case we look at arrivals that occur after time T. As after time T the session is served at a rate $g \geq \rho$, intuitively, all arrivals that occur after this time will have delays that are smaller than the worst-case delay. However, this needs to be proved. The proof is in the Appendix and it is given by Lemma 2.

To conclude the proof of the theorem, we note that from (25) and (28) we obtain (15). □

5.2 Graceful Degradation

S^{2}GPS offers graceful degradation of service to all previously active sessions in the sense that there is a smooth increase of delay when a new session starts transmitting. The smooth increase of delay is a direct result of the linear decrease of service rates. The delay increase can be evaluated by using the rate $\dot{\delta}(t)$ at which the delay changes over time when a new session arrives at the system. This rate is given by

$$\dot{\delta}_i(t) = \frac{\delta_i(t+\Delta t) - \delta_i(t)}{\Delta t}$$

If $\dot{\delta}_i(t)$ is bounded, then graceful degradation is provided. Under S^{2}GPS, we expect that $\dot{\delta}_i(t)$ will be bounded and that the bound will depend on T. We also expect that the "smoothness" of the delay increase should increase for larger values of T. In other words, if T is large, then $\dot{\delta}(t)$ should be small. Next we evaluate $\dot{\delta}_i(t)$ and we show that indeed the bound for $\dot{\delta}_i(t)$ is a decreasing function of T.

Let us suppose that a Session i which has been active for a long period of time before time $t = 0$ when a new Session N becomes active. Let $N-1$ be the number of the active sessions before $t = 0$. We assume that the number of active sessions has been constant and that after the arrival of the Session N, it will remain constant in the time interval $[0, T]$. We look at a "bit" of the Session i that arrives at time $(-t)$ and is transmitted at time $\tilde{t}$, such that $0 \le \tilde{t} \le T$. Then we have:

$$Q_i(-t) = \int_{-t}^{\tilde{t}} \hat{r}_i(\tau)\, d\tau \tag{29}$$

We proceed to calculate $\tilde{t}$. The service rate of the Session i is given by:

$$r_i(t) = \begin{cases} \frac{\phi_i}{\sum_{j=1}^{N-1}\phi_j} r(1 - \frac{t}{T}\frac{\phi_N}{\sum_{j=1}^{N}}\phi_j) & : \quad 0 \le t \le T \\ \frac{\phi_i}{\sum_{j=1}^{N-1}\phi_j} r & : \quad \tau \le 0 \end{cases} \tag{30}$$

From (29) and (30) we have:

$$\begin{aligned} Q_i(-t) &= \int_{-t}^{\tilde{t}} r_i(\tau)\, d\tau \\ &= \int_{-t}^{0} r_i(\tau)\, d\tau + \int_{0}^{\tilde{t}} r_i(\tau) d\tau \\ &= \int_{-t}^{0} \frac{\phi_i r}{\sum_{j=1}^{N-1}\phi_j}\, d\tau + \int_{0}^{\tilde{t}} \frac{\phi_i r}{\sum_{j=1}^{N-1}\phi_j}(1 - \frac{\tau}{T}\frac{\phi_N}{\sum_{j=1}^{N}\phi_j})\, d\tau \\ &= \frac{\phi_i r}{\sum_{j=1}^{N-1}\phi_j} t + \frac{\phi_i r}{\sum_{j=1}^{N-1}\phi_j}(\tilde{t} - \frac{(\tilde{t})^2}{2T}\frac{\phi_N}{\sum_{j=1}^{N}\phi_j}) \end{aligned}$$

which implies:

$$\frac{\phi_N}{2T\sum_{j=1}^{N}\phi_j}(\tilde{t})^2 - \tilde{t} + Q_i(-t)\frac{\sum_{j=1}^{N-1}\phi_j}{\phi_i r} = 0 \tag{31}$$

From (31) we obtain

$$\tilde{t} = \frac{T\sum_{j=1}^{N}\phi_j}{\phi_N}\left(1 - \sqrt{1 - 2\frac{\phi_N}{T\sum_{j=1}^{N-1}\phi_j}(\frac{\sum_{j=1}^{N}\phi_j}{\phi_i}Q_i(-t) - t)}\right) \tag{32}$$

In the same way we can find that a "bit" that arrives at $(-t)+\Delta t$ will have a delay of:

$$\delta_i(-t+\Delta t) = t - \Delta t + \tilde{\tilde{t}} \tag{33}$$

where $\tilde{\tilde{t}}$ denotes the transmission time, which is given by:

$$\tilde{\tilde{t}} = \frac{T\sum_{j=1}^{N}\phi_j}{\phi_N}\left(1-\sqrt{1-2\frac{\phi_n}{T\sum_{j=1}^{N}\phi_j}\left(\frac{\sum_{j=1}^{N}\phi_j}{\phi_i}Q_i(-t+\Delta t)-(t-\Delta t)\right)}\right) \tag{34}$$

Also, we have that:

$$Q_i(-t+\Delta t) = Q_i(t) - r_i\Delta t + A_i(-t,-t+\Delta t) \tag{35}$$

Using Equations (32), (34), and (35) we can calculate $\dot{\delta}(t)$ as a function of time t, the queue size $Q_i(t)$, and the amount of traffic $A_i(-t,-t+\Delta t)$:

$$\dot{\delta}_i(t) = \frac{T\sum_{j=1}^{N}\phi_j}{\phi_N\Delta t}\left(\sqrt{1-2\frac{\phi_n}{T\sum_{j=1}^{N}\phi_j}\left(\frac{\sum_{j=1}^{N}\phi_j}{\phi_i}Q_i(-t)-r_i\Delta t+A_i(-t,-t+\Delta t)-(t-\Delta t)\right)}\right.$$
$$\left.-\sqrt{1-2\frac{\phi_N}{T\sum_{j=1}^{N-1}\phi_j}\left(\frac{\sum_{j=1}^{N}\phi_j}{\phi_i}Q_i(-t)-t\right)}\right) \tag{36}$$

It follows from (36) that the delay rate is bounded if $Q_i(t)$ is bounded. However, $Q_i(t)$ is bounded by $Q_i(t) \le \sigma_i$, if Session i is constrained by a leaky bucket (σ_i,ρ_i) and $g_i \ge \rho_i$: Hence, S^{2}GPS provides a bounded delay rate and, as a result, graceful degradation of service. Also, it can be easily proved that $\dot{\delta}_i(t)$ is a decreasing function of T. This confirms our intuition that for larger values of T, S^{2}GPS provides smoother delay increases because the bound for the delay rate will be smaller.

6 A Packet-by-packet version of S^{2}GPS

S^{2}GPS assumes a fluid model where traffic is infinitely divisible. In reality, however, a scheduler can serve one packet at a time. In this section we define the packet approximation of S^{2}GPS, called Slow-Start Packetized Generalized Processor Sharing or S^{2}PGPS. and we show that S^{2}PGPS closely approximates S^{2}GPS. Also, we show how S^{2}GPS can be implemented using the concept of virtual time.

6.1 S^{2}PGPS

In this Subsection, we define the packet approximation of S^{2}GPS, called S^{2}PGPS. S^{2}PGPS is the scheduling discipline that transmits packets in increasing order of their finishing times under the S^{2}GPS system. S^{2}GPS attempts to approximate the fluid model as closely as possible. Note that S^{2}PGPS is derived from S^{2}GPS in the same way that PGPS is derived from GPS [16]. The question that arises is whether S^{2}PGPS is a good approximation of a S^{2}GPS system. We will prove that this is indeed the case. Specifically, we will show that a S^{2}PGPS system cannot fall behind from the corresponding S^{2}GPS system by more than one maximum packet size. We will take advantage of the following results that are available for GPS/PGPS [16]:

1. Let p and q be packets in a GPS system at time τ, and suppose that packet p completes service before packet q if there are no arrivals after time τ. Then, the packet p will also complete service before the packet q for any pattern of arrivals after time τ.

2. Let $\hat{F}_p$, F_p be the times that packet p departs from the PGPS and GPS systems, respectively. Then for all packets p,
$$\hat{F}_p - F_p \leq \frac{L_{\max}}{r} \tag{37}$$

3. Let $\hat{S}_i(0,\tau)$ be the amount of service that Session i receives under S^{2}GPS. Then, for all times τ and for each Session i:
$$S_i(0,\tau) - \hat{S}_i(0,\tau) \leq L_{\max} \tag{38}$$

It is not hard to show that the above properties also apply in S^{2}GPS. This is because the proofs of these properties for PGPS, which are given in [16], are not sensitive to time dependent service rates. Thus, a S^{2}PGPS system cannot fall behind from the corresponding S^{2}GPS system by more than one packet size. These properties facilitate the translation of delay bounds under a S^{2}GPS system to the corresponding S^{2}PGPS system.

6.2 Virtual Time Implementation of S^{2}PGPS

In this section we present an implementation of the S^{2}PGPS based on virtual times. The concept of virtual time as a means of implementing PGPS was proposed in [8, 16]; the virtual time $V(t)$ in [8, 16] is used as a measure of progress in the system. When packets arrive, the scheduler assigns virtual time deadlines to them and serves packets in increasing order of these deadlines. The virtual time is set to zero at the beginning of a busy period and increases at the marginal rate of $\frac{1}{\sum_{i \in B(t)} \phi_i}$. Thus, during any system busy period $[t_1, t_2]$, $V(t)$ evolves as follows:

$$V(t_1) = 0 \tag{39}$$
$$\frac{\partial V(t)}{\partial t} = \frac{1}{\sum\limits_{i \in B(t)} \phi_i} \qquad t_1 \leq t \leq t_2 \tag{40}$$

which yields:
$$V(t) = \int_{t_1}^{t_2} \frac{d\tau}{\sum\limits_{i \in B(t)} \phi_i} \tag{41}$$

Let us define as an *event* in the system the arrival or the departure of a packet. Let e_j be the time that the j-th event in the system occurs. By observing that $B(t)$ is constant in any time interval during which no events occur, we obtain:

$$V(e_j + \tau) = V(e_j) + \frac{\tau}{\sum\limits_{i \in B(e_j)} \phi_i} \qquad 0 \leq \tau \leq e_{j+1} - e_j \tag{42}$$

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:

$$S_i^{(p)} = \max\{F_i^{(p-1)}, V(t_i^{(p)})\} \tag{43}$$

$$F_i^{(p)} = S_i^{(p)} + \frac{L_i^{(p)}}{\phi_i} \tag{44}$$

where $t_i^{(p)}$ is the arrival time of packet p and $L_i^{(p)}$ is the length of packet p. The scheduler serves packets in increasing order of their virtual finishing times.

An implementation of S^{2}PGPS with virtual time is not straightforward and must address the following two problems:

1. From (42) we have that the virtual time $V(t)$ is calculated as a function of the ϕ's of the sessions. Note, however, that the weights of the sessions in the slow-start phase have to be modified to reflect the increasing service rates that these sessions receive, and the decreasing service rates of the other, i.e., the previously active sessions.

2. For a packet p that is transmitted during the slow-start phase of a S^2GPS of a Session k, the service rates of the session at the beginning and at the end of the transmission will be different. Correspondingly, the weight ϕ_k of a Session k will take different values during the transmission of a packet. As (44) suggests, the deadline of a packet depends on the ϕ_k of the session. If the ϕ_k of Session k is not constant over the transmission of a packet of the session, then it is not obvious how a deadline can be assigned to this packet.

We proceed to present solutions to these two problems. In Subsection 6.3 we show how the weights of sessions in the slow-start phase can be calculated. In Subsection 6.4 we show how the virtual finishing times are calculated in S^{2}GPS.

6.3 Definition of $\phi_k(t)$

Our goal is to define the ϕ_k of every Session k in $B_{new}(t)$ as a function of time, $\phi_k(t)$, such that:

$$r_k(t) = \frac{\phi_k(t)}{\sum\limits_{j\in B(t)-B_{new}(t)} \phi_j + \sum\limits_{j\in B_{new}(t)} \phi_j(t)} r \tag{45}$$

Using (7), (8), and (45), we can calculate $\phi_k(t)$ for every session in the slow-start phase. We show how this can be done in the case when (i) only one Session k is in the slow-start phase, and (ii) the set $B(t)$ of the backlogged sessions is constant in the time interval $[t_k, t_k+T]$. In Subsection 6.5 we discuss how this restriction can be relaxed.

To derive $\phi_k(t)$, recall that the service rate of Session k in time interval $[t_k, t_k+T]$ is given by:

$$r_k(t) = \frac{t-t_k}{T} \frac{\phi_k}{\sum\limits_{i\in B(t)} \phi_i} r \tag{46}$$

As only Session k is in the slow-start phase, (45) becomes:

$$r_k(t) = \frac{\phi_k(t)}{\sum\limits_{i \in B(t), i \neq k} \phi_i + \phi_k(t)} r \tag{47}$$

and (46) becomes:

$$r_k(t) = \frac{t - t_k}{T} \frac{\phi_k}{\sum\limits_{i \in B(t), i \neq k} \phi_i + \phi_k} r \tag{48}$$

From (47) and (48) we have:

$$\frac{\phi_k(t)}{\sum\limits_{i \in B(t), i \neq k} \phi_i + \phi_k(t)} = \frac{t - t_k}{T} \frac{\phi_k}{\sum\limits_{i \in B(t), i \neq k} \phi_i + \phi_k}$$

which implies:

$$\phi_k(t) = \frac{\frac{t-t_k}{T} \frac{\phi_k}{\sum\limits_{i \in B(t)} \phi_i} \sum\limits_{i \in B(t), i \neq k} \phi_i}{1 - \frac{t-t_k}{T} \frac{\phi_k}{\sum\limits_{i \in B(t)} \phi_i}} \tag{49}$$

Equation (49) clearly shows that $\phi_k(t)$ is an increasing function of time t. At time $t = t_k + T$, we have that $\phi_k(t_k + T) = \phi_k$ which implies that Session k will be assigned its fair share of the bandwidth.

6.4 Virtual Finishing Time in S^{2}PGPS

Let p be a packet of Session k that is transmitted during the slow-start phase of Session k. In this subsection we will show how the virtual finishing time of this packet can be computed upon the arrival of the packet. Note that the virtual finishing time is used as the deadline with which the packet p will be tagged upon its arrival.

As (49) suggests, the weight ϕ_k of a Session k in the slow-start phase changes during the transmission of a packet p of this session. We calculate an average value of the weight ϕ_k of the session over the transmission of the packet and we call it the "effective" value $\phi_{eff}^{(p)}$. Using $\phi_{eff}^{(p)}$, it is possible to calculate the virtual finishing time as $S_k^{(p)} + \frac{L_k^{(p)}}{\phi_{eff}^{(p)}}$. In other words, we are able to use (44) provided that we have calculated $\phi_{eff}^{(p)}$. We will show how $\phi_{eff}^{(p)}$ can be calculated for a Session k in the slow-start phase. We assume that only Session k is in the slow-start phase and that $B(t)$ is constant in $[t_k, t_k + T]$.

Let w_p denote the elapsed time between the end of the transmission of packet p and the arrival of the first packet of session k. Clearly, the transmission of packet p ends at time $t_k + w_p$. We can calculate w_p in terms of w_{p-1}. As the transmission of packet p will start at $t_k + w_{p-1}$ and end at $t_k + w_p$, we have:

$$\begin{aligned} L_k^{(p)} &= \int_{t_k + w_{p-1}}^{t_k + w_p} r_k(t)\, dt \\ &= \int_{t_k + w_{p-1}}^{t_k + w_p} \frac{t - t_k}{T} \frac{\phi_k}{\sum_{i=1}^{N} \phi_i} r\, dt \\ &= \int_{w_{p-1}}^{w_p} \frac{\tau}{T} \frac{\phi_k}{\sum_{i=1}^{N} \phi_i} r\, dt \end{aligned}$$

which yields:

$$w_p = \sqrt{\frac{2L_k^{(p)} \sum_{i=1}^{N} \phi_i}{r\phi_k} + (w_{p-1})^2} \tag{50}$$

However, packet p is served at a rate $\frac{r\phi_{eff}^{(p)}}{\sum\limits_{i=1, i\neq k}^{N} \phi_i + \phi_{eff}^{(p)}}$. As the transmission of the packet takes time $w_p - w_{p-1}$, we have:

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

which yields:

$$\phi_{eff}^{(p)} = \frac{L_k^{(p)} \sum\limits_{i=1, i\neq k}^{N} \phi_i}{r(w_p - w_{p-1}) - L_k^{(p)}} \tag{51}$$

Using (50) and (51), we can now devise a procedure for S^{2}GPS that assigns a deadline to an incoming packet of a new session. The deadlines of packets from sessions that are not in the slow-start phase can be directly calculated using (44). In the following, we present pseudo-code for the algorithm that assigns deadlines to packets from sessions that are in the slow-start phase:

```
procedure Assign_Deadline_To(Packet $p$, Session $k$)
1. if $p = 1$
2.     set $w_0 = 0$
3.     add Session $k$ to $B_{new}$
4. if $w_{p-1} < T$
5.     calculate $w_p$ using (50)
6.     calculate $\phi_{eff}^{(p)}$ using (51)
7.     assign a deadline to packet $p$ using (44)
8. else /* the slow-start phase of Session $k$ is over */
9.     remove $k$ from $B_{new}(t)$
```

When the first packet of Session k arrives, the Session k enters a slow-start phase. Thus it is inserted to $B_{new}(t)$ and w_0 is initialized to 0. Then w_1 is calculated from (50) and $\phi_{eff}^{(1)}$ is calculated from (51). A deadline to the first packet of the session packet is assigned using (44). When the p-th packet of Session k arrives, then the scheduler has to check if the session is still in the slow-start phase. This check can be carried out in the following way. If the $(p-1)$-th packet departs before the end of the slow-start phase, i.e., $w_{p-1} < T$, then packet p will also be transmitted during the slow-start phase of Session k. Thus, w_p is calculated using (50) and $\phi_{eff}^{(p)}$ is calculated using (51). However, if $w_{p-1} > T$, then the slow-start phase of Session k is over. Session k is removed from $B_{new}(t)$ and its ϕ is set to ϕ_k for all the other packets from Session k.

6.5 Relaxing the Assumption

So far we have shown how a S^{2}PGPS system can be implemented by assigning deadlines to packets of sessions in the slow-start phase provided that the following conditions hold:

Condition C_1: Only one session is in the slow-start phase

Condition C_2: B(t) is constant if a session is in the slow-start phase.

If this assumption holds, then S^{2}PGPS provides the following guarantees:

Guarantee G_1: The new session will experience a linear increase of its service rate.

Guarantee G_2: The worst-case delay bounds as given in Equation (15) will hold.

Guarantee G_3: Previously active sessions experience smooth decreases of their service rates.

We now relax this assumption and examine the impact on guarantees G_1, G_2, and G_3, if conditions C_1 or C_2 do not hold.

Let us suppose that C_1 does not hold. We will show that G_2 and G_3 still hold. When C_1 does not hold, there is a time period where $n > 1$ Sessions $k_1, k_2, \ldots, k_n$ are in the slow-start phase. Without loss of generality we can assume that Session k_1 first entered the slow-start phase, then Session k_2 and so on:

$$t_{k_1} \leq t_{k_2} \leq \ldots \leq t_{k_n}$$

In this scenario, the increase of the service rate of Session k_1 will be linear only until the time that Session k_2 enters the slow-start phase. In other words, in the time interval $[t_{k_1}, t_{k_2}]$ the service rate of Session k_1 increases linearly. After time t_{k_2} Session k_1 sees its service rate increase but not in a linear fashion. This is due to (51) not taking into account the ϕ 's of sessions $t_{k_2} \ldots t_{k_n}$. However, at time $t_{k_1} + T$, Session k_1 is guaranteed to be served at a minimum rate g_k as given by (3). Similarly, the service rates of Sessions k_2, ..., k_n also increase gradually but not linearly. Thus, when a new session enters the system, it is *always* served at a gradually increasing rate; the increase is linear only if assumptions C_1 and C_2 hold. As a result, G_3 holds.

Next we prove that G_2 holds even if C_1 does not hold. Recall that the delay bounds are computed by making worst-case assumptions, i.e., all $N-1$ sessions are continuously backlogged and only one Session k is in the slow-start phase. Let us denote the service rate of Session k as $r_k^{one}(t)$ when only Session k is in the slow-start phase and $r_k^{many}(t)$ when several sessions are in the slow-start phase. We have $r_k^{one}(t) \geq r_k^{many}(t)$ since S^{2}GPS is work-conserving and a session in the slow-start phase is always served at a rate smaller than the service rate the session gets when it is not in the slow-start phase. Hence, when C_1 does not hold, a session in the slow-start does not experience the worst-case delays, and G_2 holds.

Thus, relaxing C_1 does not have any detrimental effects on S^{2}PGPS. It turns out that this is also the case with relaxing assumption C_2; we will show that G_2 and G_3 still hold. When C_2 does not hold, then some previously active sessions are removed from the system while some "new" sessions are still in the slow-start phase. If an "old" session ceases to be backlogged, its service rate is distributed to all remaining sessions in the system. The sessions in $B_{new}(t)$ will experience a sudden increase in the service rates. As their service rates will increase, their delays will decrease; hence, G_2 will hold. However, this increase of service rates will not be at the expense of the remaining sessions in $B(t) - B_{new}(t)$, as these sessions will also take a significant part of the bandwidth that was made available. Thus the smooth decrease of service rates is guaranteed in this case, too, and G_3 will hold. Hence, the slow-start nature of S^{2}PGPS is preserved.

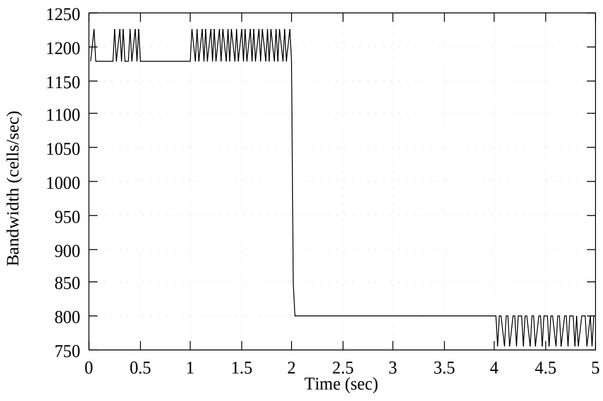
We have shown that even if our original assumption does not hold, the behavior of the packetized version of S^{2}GPS is not significantly altered. Previously active sessions still see their service rate decrease smoothly and the worst-case delay bounds for the sessions in the slow-start phase are guaranteed.

7 Evaluation

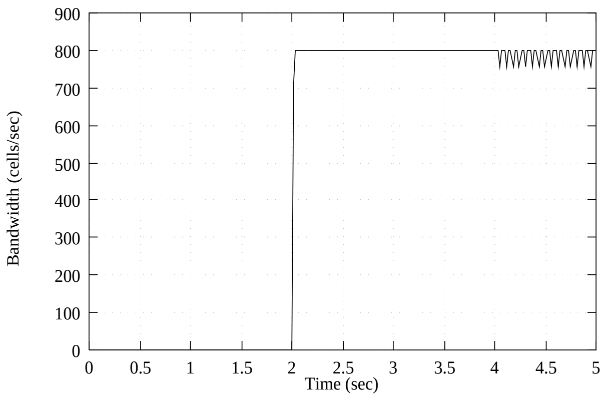
In this section we present three simulation experiments that evaluate the packetized version of our slow-start scheduler, S^{2}PGPS, with respect to the following properties. First, we will illustrate how S^{2}PGPS increases the bandwidth of new sessions and decreases the bandwidth of active sessions, for different selection of parameter T, the length of the slow-start phase. Second, we show that S^{2}PGPS prevents abrupt increases of delays when new sessions become active. In Experiments 1 and 2 we focus on the first property. In Experiment 3 we focus on the second property.

7.1 Experiment 1

Our goal in this experiment is to illustrate that S^{2}PGPS provides linear increase of service rates when only one session is in the slow-start phase. We consider an ATM Permanent Virtual Connection (PVC) with an available bandwidth of 1 Mbps. This bandwidth is distributed among three continuously backlogged sessions with $\phi_0 = \phi_1 = \phi_2 = 1$. We assume that all sessions transmit ATM 53-byte cells. Since each session being continuously backlogged results in unbounded delays, we only plot the bandwidth that is available to each session. Sessions 0 and 1 start transmitting at time $t = 0$ sec, and Session 2 starts transmitting at time $t = 2$ sec. Figure 4 shows the bandwidth for Sessions 0, 1 and 2 under PGPS. This figure clearly shows that under PGPS the bandwidth for Sessions 0 and 1 drops abruptly at time $t = 2$ sec, when Session 2 starts transmitting.

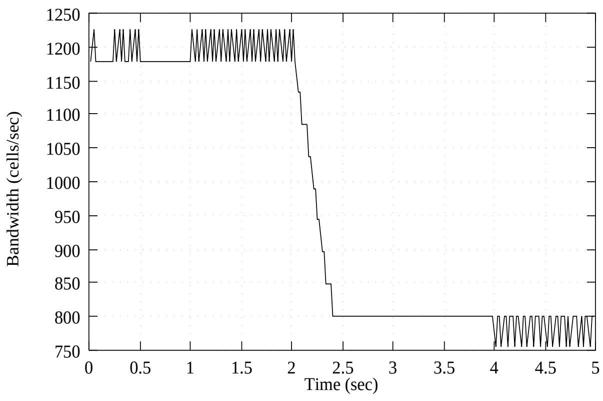


(a) Sessions 0 and 1

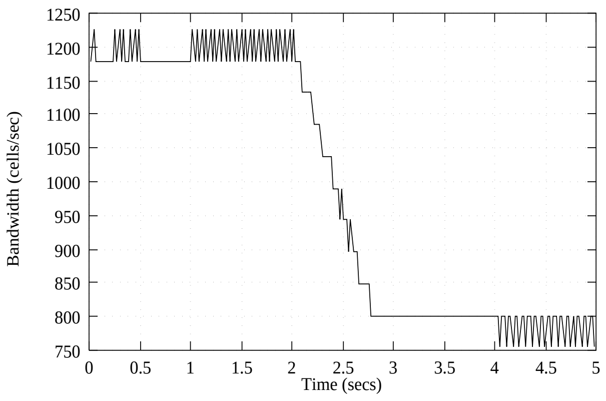


(a) Session 2

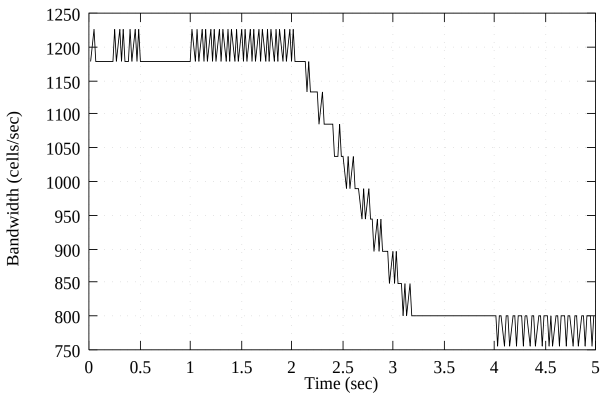
Figure 4: Available bandwidth under PGPS.



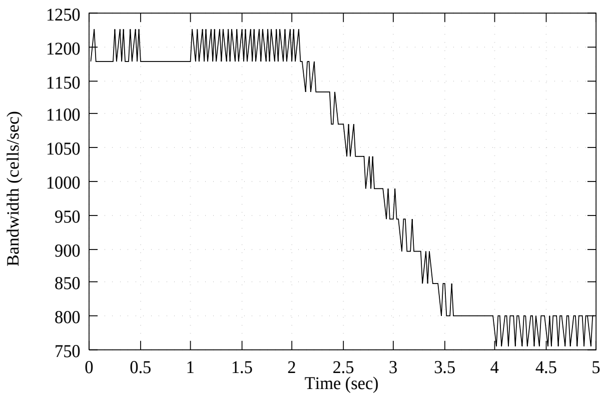
(a) S^2PGPS with T=0.4 sec



(b) S^2PGPS with T=0.8 sec

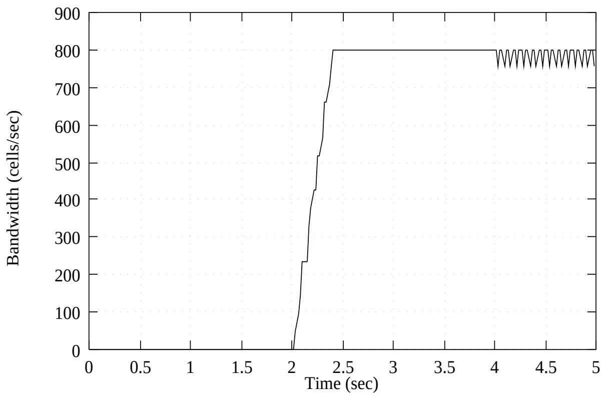


(c) S^2PGPS with T=1.2 sec

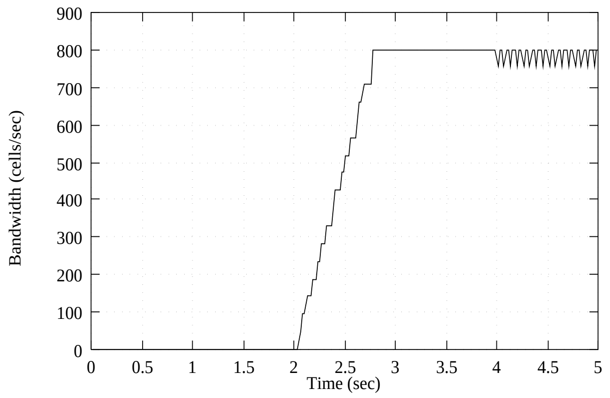


(d) S^2PGPS with T=1.6 sec

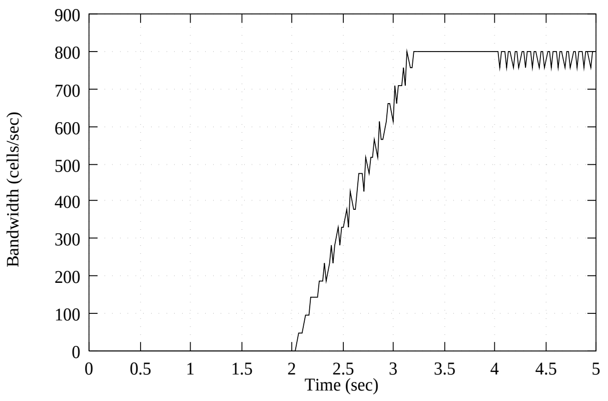
Figure 5: Available bandwidth of Session 0 under S^2PGPS (The results are almost identical for Session 1).



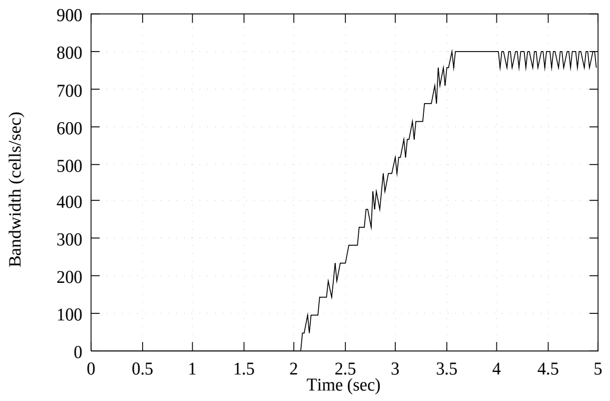
(a) S^2PGPS with T=0.4 sec



(b) S^2PGPS with T=0.8 sec



(c) S^2PGPS with T=1.2 sec



(d) S^2PGPS with T=1.6 sec

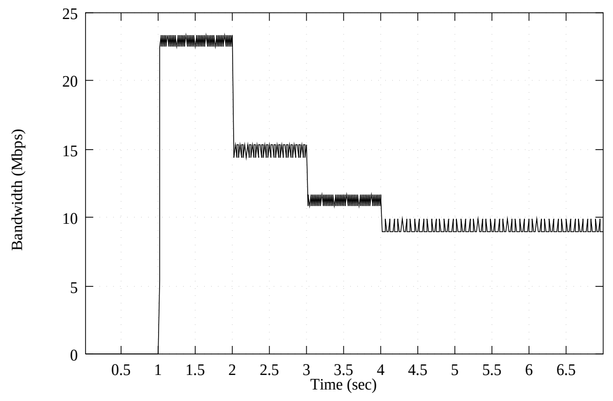
Figure 6: Available bandwidth of Session 2 under S^2PGPS.

Figure 5 depicts the bandwidth for Sessions 0 and 1 under S^2PGPS and Figure 6 depicts the bandwidth for Session 2 for various values of parameter T. Figure 5 clearly shows that the bandwidth of Sessions 0 and 1 is decreased smoothly in an interval of length T. Figure 6 shows that the bandwidth of Session 2 is increased almost linearly in an interval of length T. Note that bandwidth increase for Session 2 is not "strictly" linear. This is because of the approximation S^2PGPS introduces.

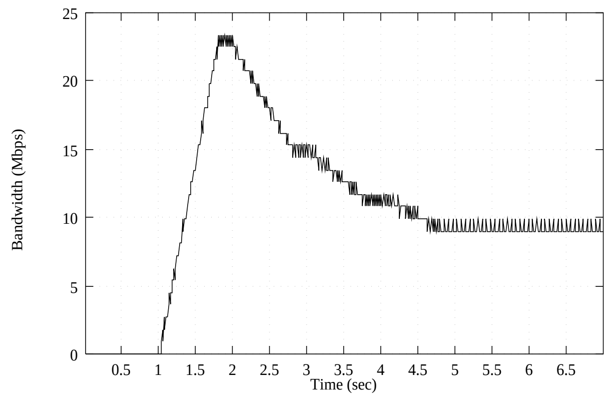
From this experiment we can clearly see that when only one session is in the slow-start phase and the set of sessions is constant, then S^2PGPS provides linear increase of the service rate of the "new" session. Correspondingly, the service rates of the previously active sessions are decreased linearly.

7.2 Experiment 2

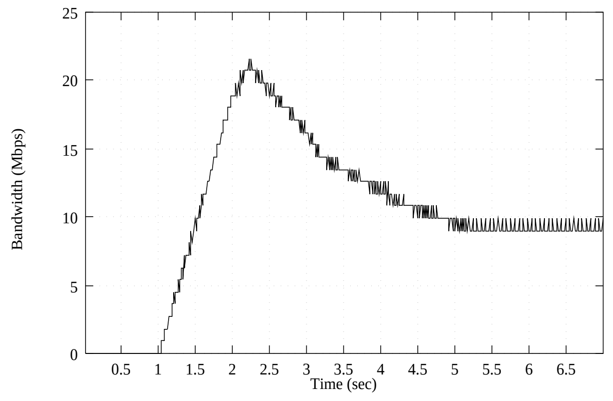
Our goal in this experiment is to examine the behavior of S^2PGPS when several sessions are in the slow-start phase. The parameters for this experiment are the same as in the example of Subsection 3.2. The available bandwidth is 45Mbps and is distributed among five sessions. All sessions have the same weights, $\phi_0 = \phi_1 = \phi_2 = \phi_3 = \phi_4$. Session 0 starts transmitting at $t = 0$ sec, Session 1 starts transmitting at time $t = 1$ sec and so on. As it was pointed out in Subsection 3.2, under PGPS, when a session starts transmitting, the service rates of all previously active sessions are decreased abruptly.



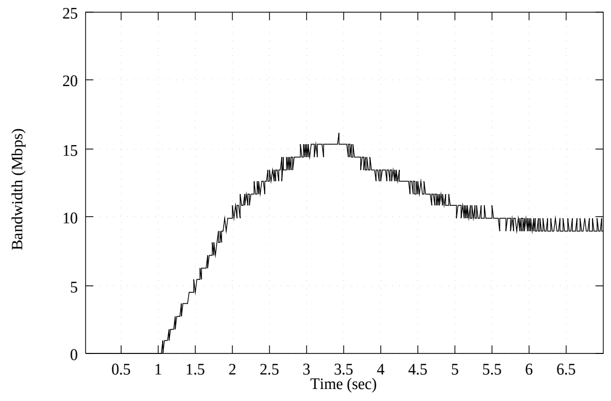
(a) PGPS (or S^{2}PGPS with $T = 0$ sec).



(b) S^{2}PGPS with $T = 0.8$ sec



(c) S^{2}PGPS with $T = 1.2$ sec



(d) S^{2}PGPS with $T = 2.2$ sec

Figure 7: Available bandwidth of Session 1.

Figure 7 depicts the available bandwidth for Session 1 for different values of the period T of the slow-start phase under S^{2}PGPS. From Figure 7 we can draw interesting conclusions about S^{2}PGPS. For values of T less than 1 sec only one session is in the slow-start phase in the interval [1 sec, 5 sec]. In these cases, the change of service rates is carried out in a linear fashion. For example, when $T = 0.8$ sec, the service rate of Session 1 is increased linearly in the interval [1 sec, 1.8 sec]. When Session 2 starts transmitting, the service rate of Session 1 is decreased linearly. This is also the case when Sessions 3 and 4 start transmitting. When $T > 1$s, then more than one session is in the slow-start phase at the same time. For example, when $T = 2.2$ sec, at time $t = 3.1$ sec three sessions (Sessions 1, 2, and 3) are in the slow-start phase. We can clearly see that the increase of service rates is not linear. Nevertheless, the smooth decrease of service rates is preserved. Moreover, at the end of the slow-start phase, every session receives from the scheduler its fair share of the bandwidth. Hence, in the presence of many sessions in the slow-start phase, S^{2}PGPS succeeds in providing smooth decreases of service rates.

7.3 Experiment 3

We assume that we have a video transmission system over PVC with a bandwidth of 1 Mbps. On this PVC, three MPEG movie transmissions are multiplexed. Sessions 1 and 2 transmit the same movie, "camera", and Session 3 transmits the movie "dino". The MPEG movie sequences are taken from a publicly available library of MPEG traces [18]. We refer to [18] for a detailed discussion and statistical analysis of these traces. Sessions 1 and 2 start transmitting at time $t = 0$ sec and they have a worst-case delay bound of 0.4 sec. Session 3 has a worst-case delay of 0.2 sec and starts transmitting at time $t = 3$ sec. These parameters are illustrated in the following table:

movie	frames per s	worst-case delay	ϕ	transmission start
camera	17	0.4 sec	1	$t = 0$ sec
camera	17	0.4 sec	1	$t = 0$ sec
dino	24	0.2 sec	2.4	$t = 3$ sec

Table 2: Parameters.

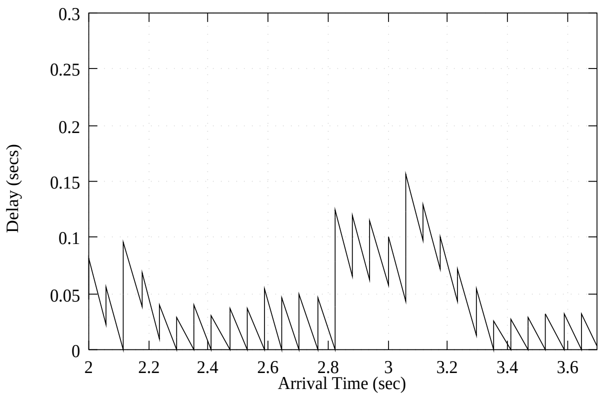
First, we use PGPS as the scheduling discipline for the multiplexing of the three sessions. Figure 8(a) depicts the delay of Sessions 1 and 2 if Session 3 did not transmit any packets at all and Figure 8(b) depicts the delay when Session 3 starts transmitting at time $t = 3$ sec. Due to the burstiness of MPEG traffic, abrupt increases of delays occur even when only Sessions 1 and 2 are active. The abrupt delay increase at time $t = 2.8$ sec in Figure 8(a) shows this phenomenon. However, the delay increase caused by the burstiness of the MPEG sources is smaller than the delay increase caused by the arrival of Session 3 at time $t = 3$ sec. From Figure 8(b) we observe that as soon as Session 3 starts transmitting at $t = 3s$, the delay of Sessions 0 and 1 is increased by almost 100 msec, corresponding to a 66% increase of delay. This example illustrates that even with very bursty types of traffic, such as MPEG, the abrupt delay increase when new sessions start transmitting can be dramatic.

Figure 9 depicts the delay of Sessions 1 and 2 when S^{2}PGPS is used. The figures show that S^{2}PGPS indeed mitigates the problem of sudden delay increase. In fact, Figure 9(a) shows that when the period T of the slow-start phase is $T = 0.4$ sec, S^{2}PGPS reduces the delay increase back by about 50 msec. From Figure 9(b) we can see that for a larger value of T ($T = 0.8$ sec), the delay of Sessions 1 and 2 assumes high values for a smaller time period. This implies that with S^{2}PGPS, the increase in delay is not only smaller, but it also lasts for a smaller period of time.

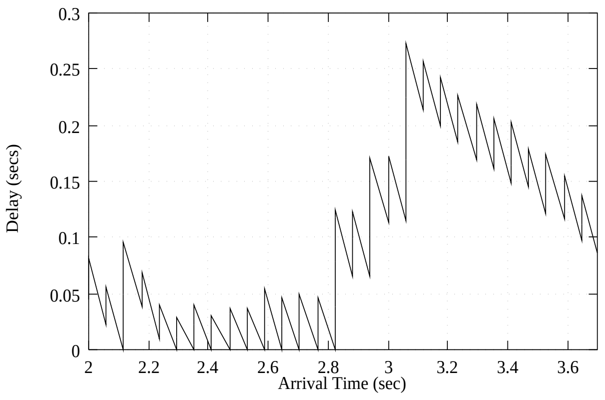
However, we expect that S^{2}PGPS will have a negative effect on the delay of Session 3. Figures 10 and 11 illustrate this inherent trade-off of S^{2}PGPS.

Figure 10 depicts the delay of Session 3 under PGPS. As Session 3 receives more than 50% of the available bandwidth, its delay is always smaller than 100 msec. However, when S^{2}PGPS is used, its delay is increased.

Figure 11 depicts the delay of Session 3 under S^{2}PGPS. We can see that the effect of S^{2}PGPS in an increase in delay of 220 msec (when $T = 0.4$ sec) or 330 msec (when $T = 0.8$ sec). It is clear, however, that this increase of delay is temporary; it lasts only while Session 3 in the slow-start phase.

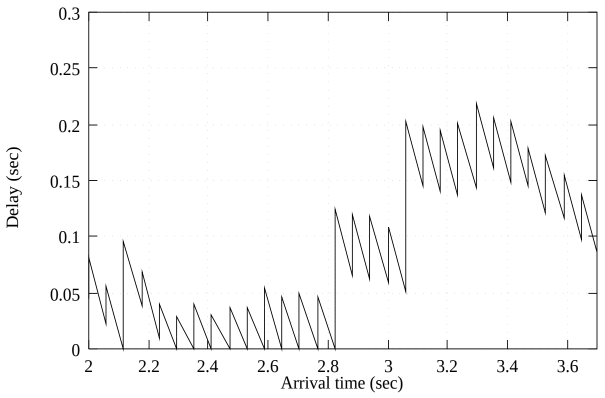


(a) Only Sessions 1 and 2 transmit

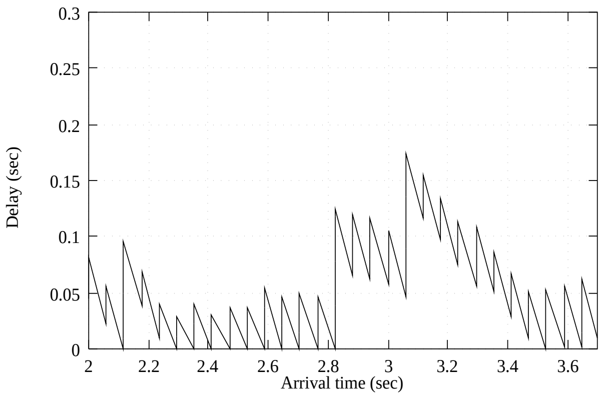


(b) All sessions transmit

Figure 8: Abrupt Increase of Delay under PGPS.



(a) S^2PGPS with $T = 0.4$ sec



(b) S^2PGPS with $T = 0.8$ sec

Figure 9: Delay for Sessions 1 and 2 under S^2PGPS.

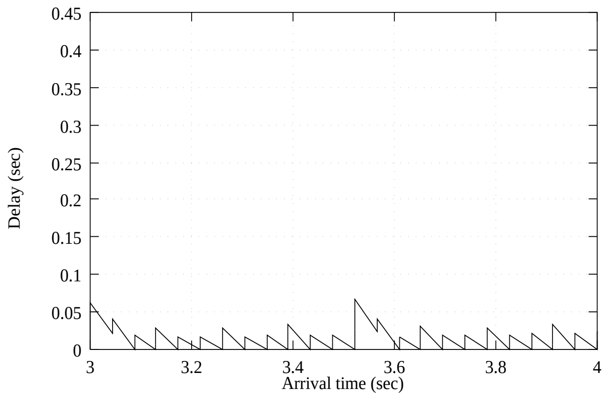
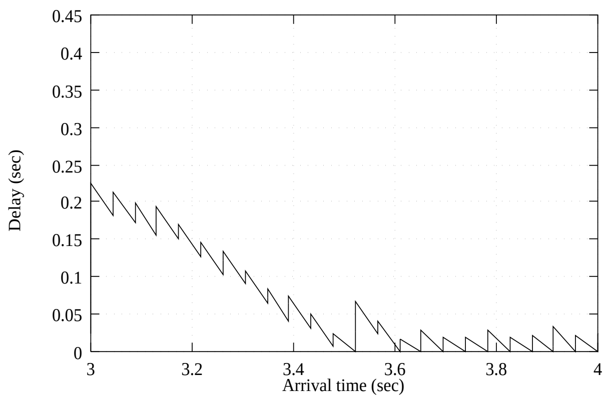
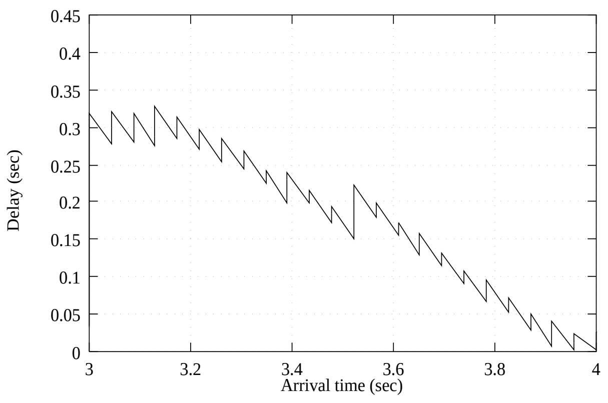


Figure 10: Delay of Session 3 under PGPS.



(a) S^2PGPS with $T = 0.4$ sec



(b) S^2PGPS with $T = 0.8$ sec

Figure 11: Delay for Session 3 under S^2PGPS.

8 Conclusion and Future Work

In this paper we have shown that the GPS scheduling discipline is unable to provide graceful degradation of service since the service rates of active sessions decrease abruptly when new sessions start transmitting. We have proposed and analyzed a modification to GPS, called Slow-Start GPS or S^2GPS that remedies this problem. We have presented concrete cases where this new scheduling discipline can be utilized. Adaptive applications and congestion control can benefit from S^2GPS and provide better QoS guarantees to the end user. We have shown how S^2GPS can be efficiently implemented in packet-switched networks; the packetized version of S^2GPS, S^2PGPS, can be implemented using the concept of virtual time. Finally, by simulations we have tested the effectiveness of S^2GPS.

We have shown that if only one session is in the slow-start phase, then S^2GPS provides linear increase of the service rate. When several sessions are in the slow-start phase, the increase of service rates is carried out in a smooth fashion, but not linearly. Additional work is needed for the design of a scheduler that guarantees linear increase of service rates, if multiple sessions are in the slow-start phase.

References

[1] ATM Forum, ATM Forum Traffic Management Specification Version 4.0, April 1996.

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

[3] J. C. R. Bennett and H. Zhang. WF2Q: Worst-case fair weighted fair queueing. In *Proc. IEEE Infocom '96*, March 1996.

[4] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. IETF RFC 1633, July 1994.

[5] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: architecture and mechanisms. In *Proc. ACM Sigcomm'92*, pages 14–26, August 1992.

[6] R. L. Cruz. A calculus for network delay, part I: network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

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

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proc. ACM Sigcomm'89*, pages 1–12, 1989.

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

[10] P. Goyal and H. M. Vin. Generalized guaranteed rate scheduling algorithms: a framework. Technical Report TR95-30, University of Texas at Austin.

[11] 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.

[12] V. Jacobson and M. J. Karels. Congestion avoidance and control. In *Proc. ACM Sigcomm'88*, 1988.

[13] J. Liebeherr, D. E. Wrege, and Domenico Ferrari. Exact Admission Control in Networks with Bounded Delay Services. *IEEE/ACM Transactions on Networking*, 4(6):885 – 901, December 1996.

[14] P. McKenney. Stochastic fair queueing. In *Proc. IEEE Infocom'90*, June 1990.

[15] O. Ndiaye. *An efficient implementation of a hierarchical weighted fair queue packet scheduler*. PhD thesis, Massachussets Institute of Technology, May 1994.

[16] 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.

[17] 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.

[18] O. Rose. Statistical properties of MPEG video traffic and their impact on trafic modeling in ATM systems. Technical Report Technical Report 101, University of Wurzburg, February 1995.

[19] D. Saha, S. Mukherjee, and S. H. Tripahi. Carry-over round robin: a Simple cell scheduling mechanism for ATM networks. In *Proc. IEEE Infocom'96*, pages 630–637, 1996.

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

[21] D. Stiliadis and A. Varma. Design and analysis of frame-based fair queueing: a new traffic scheduling algorithm for packet-switched networks. In *Proc. ACM Sigmetrics'96*, pages 104–115, May 1996.

[22] D. Stiliadis and A. Varma. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. In *Proc. ACM Sigcomm'96*, pages 1d.2.1–1d.2.9, 1996.

[23] O. Yaron and M. Sidi. Generalized processor sharing networks with exponentially bounded burstiness arrivals. *Journal of High Speed Networks*, 3:375–387, 1994.

[24] Z. L. Zhang, D. Towsley, and J. Kurose. Statistical analysis of generalized processor sharing scheduling discipline. In *Proc. ACM Sigcomm'94*, pages 68–77, August 1994.

A Appendix

Lemma 1 *Let Session k be (σ_k, ρ_k)-constrained. If $T > \frac{2\sigma_k g_k}{\rho_k^2}$, then the worst-case delay occurs for an arrival at time t^* that departures at a time earlier than T.*

Proof: We need to prove that

$$t^* + \delta_k(t^*) \leq T \tag{52}$$

From (21) and (24) we have that (52) is equivalent to:

$$\sqrt{\frac{2T(\sigma_k + \rho_k(\frac{\rho_k T}{2g_k} - \frac{\sigma_k}{\rho_k}))}{g_k}} \leq T$$

It can be easily found that this equivalent to $\rho_k \leq g_k$, which holds per assumption. Hence, (52) holds. $\square$

Lemma 2 *Any arrival of a session that occurs outside the slow-start phase of that session, has a delay less than the worst-case delay.*

Proof: As in the proof of Theorem 1, we assume that a session becomes active at time 0. Therefore, an arrival outside the slow-start phase has an arrival time $t > T$. We will prove that the maximum delay that corresponds to this case is less than δ_{max} (as given in (15)). Note that for ease of notation we again drop the index k.

As in Case 2 in the proof of Theorem 1, we find that the delay for an arrival at time $t > T$ is $\delta(t) = (\frac{T}{2} + \frac{\sigma}{g}) + \frac{\rho - g}{g} t$. As $\delta(t)$ is a monotonously decreasing function of time, the maximum delay will be at $t = T$, i.e.,

$$\delta(T) = \frac{\sigma + \rho T}{g} - \frac{T}{2} \tag{53}$$

We need to examine the following cases:

- If $T < \frac{2\sigma}{g}$, then from (15) we have that $\delta_{max} = \frac{T}{2} + \frac{\sigma}{g}$. It can be easily verified that $\delta(T) < \delta_{max}$.
- If $\frac{2\sigma}{g} \leq T < \frac{2\sigma g}{\rho^2}$, then from (15), $\delta_{max} = \sqrt{\frac{2T\sigma}{g}}$. Let us suppose that:

$$\delta(T) > \sqrt{\frac{2T\sigma}{g}} \tag{54}$$

As we have $T \geq \frac{2\sigma}{g}$, we obtain:

$$\frac{\sigma}{g} + (\frac{\rho}{g} - \frac{1}{2})T > \sqrt{\frac{2\sigma}{g}\frac{2\sigma}{g}} \tag{55}$$

which yields:

$$(\frac{\rho}{g} - \frac{1}{2})T > \frac{\sigma}{g} \tag{56}$$

This can hold only if $\rho > \frac{g}{2}$. As $T < \frac{2\sigma g}{\rho^2}$, we obtain:

$$(\frac{\rho}{g} - \frac{1}{2})\frac{2\sigma g}{\rho^2} > \frac{\sigma}{g} \tag{57}$$

This implies:

$$0 > \left((\frac{\rho}{g}) - 1 \right)^2 \tag{58}$$

which is a contradiction. Hence, $\delta(T) \leq \delta_{max}$.

- If $\frac{2\sigma g}{\rho^2} \leq T$, then from (15), $\delta_{max} = \frac{\sigma}{\rho} + \frac{\rho T}{2g}$. Let us suppose that:

$$\delta(T) > \frac{\sigma}{\rho} + \frac{\rho T}{2g}$$

Then, by using (53) we obtain:

$$\frac{\sigma}{g} + \frac{\rho - g}{2g} T > \frac{\sigma}{\rho}$$

which is a contradiction, because $\rho \leq g$.

Thus, $\delta(T) \leq \delta_{max}$. The proof is complete if we recall that for $t > T$, we have $\delta(t) < \delta(T)$. $\square$