On Adaptive Bandwidth Sharing with Rate Guarantees

N.G. Duffield† T. V. Lakshman* D. Stiliadis*

† AT&T Laboratories
Rm A175, 180 Park Avenue
Florham Park,
NJ 07932-0971 USA

* Bell Labs
Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733, USA

Abstract—**The objective of recent research in fair queueing schemes has been to efficiently emulate a fluid-flow generalized (weighted) processor sharing (GPS) system, as closely as possible. A primary motivation for the use of fair-queueing has been its use as a means of providing bandwidth guarantees and as a consequence end-to-end delay bounds for traffic with bounded burstiness. The rate guarantees translate to scheduling weights which are set when admission control is done. A consequence of fair queueing systems closely emulating GPS is that when one or more connections are not backlogged, any "excess" bandwidth is distributed to backlogged connections in proportion to their weights. However, weights are set based on the long-term requirements of traffic flows and not in any state-dependent manner that reflects instantaneous needs. In this paper, we question the notion that queueing systems should closely emulate a GPS system. Instead of emulating GPS, we propose three modified scheduling schemes which preserve the rate guarantees of fair queueing (and hence preserve deterministic delay bounds) but adaptively redistribute the excess bandwidth such that either losses are reduced or delays equalized. We compare the performance of the proposed schemes to that of fair queueing using different traffic sources such as voice and video, as well as sources which have aggregate long-range dependent behavior. We find that the proposed schemes, in comparison to packet GPS (PGPS), reduce packet losses and curtail the tails of delay distributions for real-time traffic and hence permit the use of significantly smaller playout buffers for the same network load.**

Keywords—**fair queueing, scheduling, jitter, delay bounds, real-time traffic, traffic management**

I. Introduction

Much research attention has recently been focused on the development of fair-queueing systems [3], [10], [19], [20], [21], [24] that, within the non-preemption constraints of a packet system, closely emulate fluid flow generalized processor sharing systems (GPS) and operate at high speeds. Demers, Keshav, and Shenker proposed a fair queueing scheme that emulated GPS by using a simulated fluid flow GPS system as a reference and basing packet scheduling decisions on the order of departures in the simulated GPS system. The computational burden of simulating GPS was reduced in the self-clocked fair queueing scheme proposed by Golestani [10] which showed how fair queueing could emulate GPS without simulating GPS for reference. Subsequent work has led to both improved emulation bounds, further reductions in computational complexity and efficient methods for implementation [11], [20], [21], [24]. However, there is no detailed examination of whether the exclusive emulation of GPS is always appropriate for all networking applications.

In this paper, we question the notion that queueing systems must emulate GPS closely. A primary motivation for the expected use of fair queueing in routers and switches is its ability to guarantee worst case end-to-end delay bounds for leaky bucket controlled sources [17]. However, to guarantee worst case delay bounds the scheduler merely needs to isolate flows so that each flow receives its guaranteed share of the link bandwidth [1], [19]. The guarantees do not depend on the property of GPS systems that the unused shares of non-backlogged flows be redistributed to backlogged flows in proportion to their guaranteed fractional share of link bandwidth (or equivalently scheduling weights). A fair queueing system need act as one only when all connections are backlogged. When excess bandwidth (i.e. bandwidth beyond what has already been guaranteed to backlogged connections) is available, the fair queueing system, instead of emulating GPS that distributes excess bandwidth according to long term needs, should redistribute this excess bandwidth in a manner which reflects the current or instantaneous needs of backlogged flows (i.e. a flow state dependent redistribution). The redistribution policy can be picked to optimize measures such as packet loss probabilities or to control the tails of delay distributions.

It has been shown in [1], [23] that even when a FIFO scheduler is used and all sources are leaky-bucket shaped, the mean and 99th percentile of delays are much lower than the worst-case bounds guaranteed by a GPS server. However, there are service disciplines that can distribute the bandwidth in a more elegant manner. Panwar et.al. [16] showed, that the shortest time to extinction policy is actually optimal for scheduling customers with deadlines. Furthermore, a policy that simply serves the longest queue has been shown to require less buffer space to prevent buffer losses than FIFO or other per-flow disciplines [6], [13]. Similarly, policies that serve the queue that is more likely to violate a bound are proved to provide optimal buffer utilization [9]. However, such disciplines by themselves cannot provide any bandwidth guarantees. Actually, low bandwidth constant bit rate connections are likely to starve for long intervals of time, until their queue becomes at least as large as the queues of heavily bursty, high bandwidth connections. Similarly, users that can accept a significant loss rate, using some form of Forward Error Correction, can increase their throughput by simply sending more packets. There is no method to control the bandwidth of individual users.

We propose three modified fair queueing schemes that adaptively redistribute excess bandwidth while, like fair queueing, guaranteeing each flow its specified share of the link bandwidth. This preserves fair queueing's ability to provide worst case end to end delay bounds and the schemes work like fair queueing when there is no excess bandwidth. In addition, the schemes provide worst-case fairness, bounding the time that a connection may not see any service. The excess bandwidth, however, is redistributed as follows:

1. Longest delay first (LDF) that serves the flow with current longest delay.
2. Least time to overflow (LTO) that serves the flow with minimum difference between maximum allowed delay and current delay.
3. Least time to overflow with leaky buckets (LTO_LB) that serves the flow which would cause buffer overflow first if worst case arrivals happen.

We compare the performance of these policies to PGPS using both trace driven simulations as well as simulations with traffic models for various types of sources.

The LDF policy uses excess bandwidth to reduce the variance of the delay distribution. This has the benefit of reducing the playout buffer for voice and video sources. Simulations with video traces and with voice traffic show that indeed this policy performs better than PGPS without any sacrifice of worst case guarantees. Since the deviation from the maximum allowed delay is not taken into account, flows with small delay bounds (like voice) get almost no excess bandwidth in the presence of flows with large delay bounds. Long term congestion or small errors in assigning weights can result in these flows experiencing losses much more than flows with large delay bounds. The LTO policy tries to minimize packet losses by assigning excess bandwidth under the assumption that the flow which is likely to overflow the quickest has the most instantaneous bandwidth need. In doing so, it takes into account the current deviation of each flow from its maximum allowed delay. Simulations with a mix of CBR, voice, and video sources with very different delay bounds show that this policy reduces losses for all classes as well as reduces the variance of delay for each class. The LTO_LB policy goes one step further by determining the connection that is most likely to overflow its buffer under the assumption that arrivals are leaky-bucket shaped.

A scheduling and buffer sharing scheme that tries to guarantee specific delays while ensuring loss-free behavior is presented in [7]. Our proposal is fundamentally different from the scheme presented in [7]. We assume that our system does not have enough buffers to guarantee zero losses. So our goal is not to define a system that will guarantee some specific delays and at the same time zero losses. Instead, we investigate the effect of state-dependent scheduling mechanisms in minimizing the tails of the delay distributions while providing isolation, and also investigate the effect of such mechanisms on the number of buffers that need to be reserved if limited losses are acceptable.

II. Architecture

Fair Queueing systems attempt to offer the same *normalized service* to any two connections that are continuously backlogged during an interval of time $(t_1, t_2]$, where by normalized service we mean the ratio of the offered service and the allocated rate. The basic feature of such a system is that it treats connections in the same way, irrespective of their burstiness characteristics. Our claim is that end-users are not necessarily interested in a fair distribution of free bandwidth. Instead, for many applications, end-users are merely interested in minimizing their packet losses or seeing low tails in their delay distributions. Of course, the queueing scheme must be able to provide some guarantees regarding worst-case service offered to a connection when the system is fully loaded. Isolation and protection requirements such as these were defined in [1] as necessary to protect the system from misbehaving users that will try to monopolize resources.

A. Preliminaries

The definition of our state-dependent queueing system uses the methodology presented in [20] for designing fair queueing systems. The method is based on the class of servers called Rate Proportional Servers (RPS). The key concept is a virtual time function $v_i(t)$ or *system potential* associated with each connection in the system that represents the total normalized service offered to a connection during its backlogged periods. The virtual time of a connection is defined as a non-decreasing function of time during a system-busy period. When connection i is backlogged, its virtual time increases exactly by the normalized service it receives. That is, for any interval of time $(\tau, t]$ that connection i is continuously backlogged,

$$v_i(t) - v_i(\tau) = \frac{W_i(\tau, t)}{g_i}.$$

where, $W_i(\tau, t)$ denotes the amount of service received by session i during the interval $(\tau, t]$ and g_i is the scheduling weight allocated to connection i. When the connection is idle, however, the virtual time of the connection is updated through the *system virtual time*. The system virtual time is a non-decreasing function of time that keeps track of the progress of the total work done by the scheduler. When an idle session i becomes backlogged at time t, its virtual time $v_i(t)$ is set to $v(t)$ to account for the service it missed. Schedulers use different functions to maintain the system virtual time, giving rise to widely different delay and fairness-behaviors. Exactly this flexibility of the rate-proportional servers allows a simple implementation of the system without the requirement of simulating in parallel a fluid system.

At any instant t, the scheduler services only the subset of connections with the minimum virtual time, and each connection in this subset receives service in proportion to its reserved rate g_i. Thus, the scheduler can be seen to increase the virtual times of the connections in this subset at the same rate. A packet server approximates this behavior by sorting the packets in the order of their finishing virtual times under the fluid server, and transmitting them in increasing order of the finishing virtual times. The finishing virtual time of the kth packet of a session i, arriving at the scheduler at time t, is computed as

$$F_i^k = \min(F_i^{k-1}, v(t)) + \frac{L_i^k}{g_i}, \tag{1}$$

where L_i^k is the length of the packet and g_i the allocated rate of session i. That is, the finishing virtual time is computed by adding the service time of the packet at the allocated rate to its starting virtual time. The latter, in turn, is the minimum of the finishing virtual time of the previous packet received from session i, and the current system virtual time.

At the time that a connection becomes backlogged, its virtual time is updated based on the system virtual time function that keeps track of the progress of the total work done by the scheduler. When an idle session i becomes backlogged at time t, its virtual time $v_i(t)$ is set as

$$v_i(t) = \max(v_i(t-), v(t)),$$

to account for the service it missed. Rate-Proportional Servers may use a wide range of functions to maintain the system virtual time, but the function chosen must satisfy two fundamental properties: First, during any interval $(t_1, t_2]$ within a system-busy period, the system virtual time function must be increased with a rate of at least one, that is,

$$v(t_2) - v(t_1) \geq t_2 - t_1. \tag{2}$$

Second, the system virtual time function must never exceed the virtual time of any backlogged connection. These two conditions are sufficient to achieve a delay bound equal to that of PGPS.

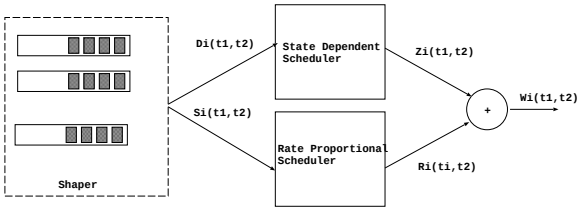

Fig. 1. Scheduling System

B. State Dependent Fair Queueing

An illustration of the logical structure of our system is presented in Figure 1. Initially, all arriving packets enter per-connection queues in a shaper device. A packet is considered as having arrived to the system only when the last bit of the packet arrives to the system. Similarly, packet service is considered to be completed only when the last bit of the packet has been served. The shaper releases packets to the scheduler with a rate exactly equal to the allocated. We denote by $S_i(\tau, t)$ the service offered by the shaper to connection i during an interval of time $(\tau, t]$. Packets are transferred from the shaper to the the scheduler with infinite capacity. Let $R_i(\tau, t)$ denote the service offered by the RPS scheduler.

Packets that have not become eligible for service remain in the corresponding connection queue in the shaper, while all the eligible packets wait for service in the RPS scheduler queue. Service is always provided from the RPS scheduler queues, as long as packets are available there. When all scheduler queues are empty, a *state dependent scheduler* (SDS) is invoked and selects a packet from the shaper queues for transmission. The service offered to a connection does not change the state of the shaper. Let us denote with $D_i(t_1, t_2)$ and $Z_i(t_1, t_2)$ the amount of traffic that is forwarded and serviced by the SDS scheduler respectively. If the state dependent scheduler is work-conserving, then the system is work conserving as well. Note also, that for every time t, $D_i(t_1, t_2) = Z_i(t_1, t_2)$. We will refer to these systems as simply FQ-SDS.

An argument that can be made against the above adaptive policies, is that a greedy user will always receive more bandwidth. In such a case, users may request less bandwidth and try to receive more by sending more bursty traffic. Limiting such behavior can be explicity done in many ways. The simplest is policing. In most networks that provide some type of guarantees, it is reasonable to assume that there is some kind of policing. So misbehaving users will be easily isolated. If policing is not a viable option, the adaptive policies can themselves be tailored to counter greedy behavior. Instead of picking the user with maximum queue length or the minimum time to overflow, we can, for example, pick the second such candidate. In this case, users that consistently overflow the buffers cannot depend on receiving any excess bandwidth; it will not be in their best interests to be excessively bursty.

C. Redistributing Excess Bandwidth

This section describes the schemes that we use for the state dependent component of our schedulers. Note that this component is used only to redistribute the excess bandwidth and does not affect the guaranteed rates. The basic idea is that the scheduler, in picking connections to serve, should use information about the state of connections so as to minimize packet losses, reduce average delays for some traffic classes, or shape delay distributions.

Simple Longest Queue First has been shown to minimize the overall packet losses in a system with finite buffers [16]. Here, the scheduler always picks a packet from the longest queue for transmission. However, if the arrival rates and guaranteed shares are such that the expected delays are are not the same for all connections then the longest queue first scheme will always favor high delay (and generally high bandwidth connections). Also, if buffer allocations are not the same, serving the longest queue first may lead to buffer overflow for connections with a low buffer or low relative bandwidth guarantee.

To account for these differences in buffer sizes and bandwidth requirements, we can simply weight the queue size by the allocated bandwidth to make the scheduler a Longest Delay First scheduler. The scheduler then uses the excess bandwidth to try and equalize the delays of all connections in the system and minimize the packet losses if buffer sizes have been allocated proportional to bandwidth requirements. We will refer to this algorithm as *FQ-LDF*.

Note that this still does not take into account the maximum allowed delay for each connection. An enhancement to FQ-LDF is to explicitly take into account the different delay requirements that applications may have. We would like to be able to allow low maximum delays to voice sources, somewhat higher maximum delays to video sources, and much higher maximum delays for data sources. Also, constant bit rate sources may require extremely low delays, even when their bandwidth allocation is very low. To account for these differences, a

maximum delay, that is usually a function of the buffer allocated to a connection, is associated with each connection. The scheduler serves connections that are most likely to exceed their allocated maximum delay. Let D_i denote the maximum delay allocated to connection i. Let us also denote by $d_i = Q_i(t)/g_i$ the delay that the last packet of the queue associated with that connection at time t may see. The connection that should be served is the one that is more likely to overflow and exceed its delay bound. Thus, connections should be served in increasing order of the $(D_i - d_i)$ values.

The above schemes have not taken into account other characteristics of the source that may be known during admission control. Characteristics could be the maximum burst that can be transmitted by a connection or its peak arrival rate. For example, if sources are leaky bucket shaped, as is the case for sources in the Controlled Load traffic class, we could use this information to predict future arrivals from sources and calculate more precisely the connections that are likely to overflow or see delays higher than the allocated. In systems, where a dual, or multiple leaky bucket is used to shape the input traffic, a similar approach can be used. In fact, we do not require that a connection is shaped by a leaky bucket. All we need is that there is a function specified that provides information as to the maximum burst that can arrive for a connection, as well as the worst-case time that this burst may arrive.

The main idea consists of tracking the state of the connection based on its packet arrivals. This tracking is similar to the policing function that is used to drop packets from connections when they exceed their traffic specification. Based on the state of the connection we can estimate the worst-case arrivals for a given connection. In order to minimize the effect of worst-case arrivals for connections, we can serve connections in a way that will minimize the probability that they will be penalized for such an arrival pattern. Note, that since the system is only distributing the instantaneous available bandwidth in a state dependent manner and since connections are unlikely to send their worst case traffic at the same time, such an approach may lead to significant improvements in maximum delays or packet losses.

Let $B_i(\tau, t)$ be the function that describes the maximum amount of traffic that can arrive for connection i during the interval $(\tau, t]$ and let $A_i(\tau, t)$ denote the actual arrivals from session i during the same period. That is, for every interval $(\tau, t]$:

$$A_i(\tau, t) \leq B_i(\tau, t) \tag{3}$$

Then at time t, a maximum burst equal to

$$\Delta B_i \leq B_i(\tau, t) - A_i(\tau, t) \tag{4}$$

may arrive with the peak rate. If the queue size of connection i is equal to $Q_i(t)$ and the maximum buffers allocated to that connection are β_i, then a connection may overflow if

$$\beta_i - Q_i(t) \geq \Delta B \tag{5}$$

Among the connections that may overflow, we will select to transmit a packet from the connection that may overflow sooner, if the maximum burst arrives with the peak rate allocated to that connection. Or,

$$j = min_{k:\beta_k - Q_k(t) \geq \Delta B_k}\left(\frac{\Delta B_k}{P_k}\right) \tag{6}$$

where P_i is the peak rate allocated to connection i.

III. Analysis

A. Preliminaries

Definition 1: A **system busy period** is a maximal interval of time during which the server is never idle.
During a system busy period the server is always transmitting packets.

Definition 2: A **backlogged period for session** i is any period of time during which packets belonging to that session are continuously queued in the system.
Let $Q_i(t)$ represent the amount of session i traffic queued in the server at time t, that is,

$$Q_i(t) = A_i(0, t) - W_i(0, t).$$

A connection is backlogged at time t if $Q_i(t) > 0$.

Definition 3: A **session** i **busy period** is a maximal interval of time $(\tau_1, \tau_2]$ such that for any time $t \in (\tau_1, \tau_2]$, packets of connection i arrive with rate greater than or equal to g_i, or,

$$A_i(\tau_1, t) \geq g_i(t - \tau_1).$$

A session busy period is the maximal interval of time during which if the session were serviced with exactly the guaranteed rate, it would remain continuously backlogged. It is important to realize the basic distinction between a session backlogged period and a session busy period. The latter is defined only in terms of the arrival function and the allocated rate. Thus, the busy period serves as an invariant for evaluating the worst-case behavior of different scheduling algorithms under the same arrival pattern.

B. Worst-Case Performance

In [19] it was shown that any RPS scheduler has the same worst-case performance as a Weighted Fair Queueing scheduler, and it can thus guarantee the same worst-case delay bounds. However, the addition of the shaping mechanism and the state dependent mechanism affects the arrival pattern at the scheduler. The intuition is that the SDS scheduler is only invoked when there is a free bandwidth available. Worst-case performance usually assumes a fully utilized system. We will now show that the worst-case service offered by the system as described above is not affected by the method by which free bandwidth is distributed.

Our main goal is to provide a lower bound for the service offered by the system to a session i during a session busy period. Note, that according to our definition a session busy period is defined as the maximal interval of time during which the traffic arriving for that connection is at least equal to the allocated bandwidth. Notice, however, that the traffic that arrives to the RPS scheduler can be lower than the traffic arriving to the system. The reason is, that the connection receives excess bandwidth from the SDS server. That means, that the busy periods that the RPS scheduler sees for a connection i, may be different than the connection busy periods as seen by the system. It is easy to verify, however, that a busy period in the RPS system can continue after the end of a busy period as seen by the whole system. We will first consider the fluid modeled RPS system. Let us denote with $W_i^F(\tau, t)$ the service offered by the fluid RPS server and with $W_i^P(\tau, t)$ the service offered by the corresponding packet-by-packet server.

Lemma 1: Let τ be the beginning of a busy period for connection i. The worst case offered service to connection i for any time t during the same busy period is bounded by

$$W_i^F(\tau, t) \geq g_i(t - \tau)$$

The proof is in the Appendix. The basic idea is that since the shaper is offering service at least equal to that reserved for connection i, the scheduler will always have enough packets to service connection i with a rate equal to the reserved.

Note, that we have made no assumption for the SDS server. We can thus assume that it is a packet system. In addition, the packet and fluid RPS systems see exactly the same arrivals, and they are both work conserving systems. Thus, in both systems, the SDS server is serving packets during the same intervals of time. Thus, if the packet server offers different service than the fluid server, this difference is due to the discrepancy between the packet-by-packet and fluid RPS server. But we know from [19] that this difference is bounded by $L_i/g_i + L_{max}/r$ where L_i is the maximum packet size for connection i, L_{max} is the maximum packet size of any connection in the server and r is the link capacity. We can thus write:

Corollary 1: The worst case offered service to the packets of the j-th busy period of connection i that started at time τ is

$$W_{i,j}^P(\tau, t) \geq g_i(t - \tau - \frac{L_i}{g_i} - \frac{L_{max}}{r}).$$

This Corollary is a sufficient condition to prove that the system offers the same worst-case end-to-end delays as PGPS under the rate-proportional assignments [19].

C. Worst-Case Fairness

Worst-case fairness was initially defined by Parekh in [17] and later expanded by Bennett and Zhang [24]. The measure is based on the worst-case delay for clearing the backlog of a session's queue. According to this, for every session i that is continuously backlogged during the interval $(t_1, t_2]$,

$$W_i(t_1, t_2) \geq (t_2 - t_1)g_i - C_i, \tag{7}$$

where $W_i(t_1, t_2)$ represents the service offered to session i during the interval $(t_1, t_2]$, and C_i is a constant. We will call the smallest value of C_i satisfying this inequality as the *worst-case fairness index* (WFI). Bennett and Zhang [24] defined a fairness parameter based on the smallest value of C_i satisfying the inequality, and normalizing it to the allocated rate of the session.

The worst-case fairness index measures the deviation between the service received by a session in the packet-level scheduler and the service received by the same session in a corresponding fluid server, where the session is serviced at a rate of g_i at each instant. Minimizing this parameter improves the traffic mix at the output of the scheduler [24]. However, it should be noted that this parameter does not specify how the free bandwidth left over by idle sessions is distributed across the active ones. Specifically, even for the algorithms that distribute the free bandwidth in a state dependent manner, we can show that they offer the same worst-case fairness as Worst-Case Weighted Fair Queueing [24]. The proof is in the Appendix.

IV. Performance Studies

We simulated the system shown in Figure 1 using a variety of sources to generate traffic and using the different schedulers (FQ-LDF, FQ-LTO, and PGPS) that were described in the previous section. In the simulated system, each source is connected to a leaky bucket shaper. The shaper is connected via an access line to the multiplexing system. For each connection, the scheduler in the multiplexer maintains a separate buffer whose drain time at the guaranteed rate is equal to the connection's maximum allowed delay D_i. The sources used are:

1. Two-state exponential On-Off sources: We used these to emulate compressed voice sources and so used small maximum allowed delays for these sources. The mean load generated per connection is 1/32 of the link capacity.
2. Video teleconference sources: We used the DAR(1) model which is a Markov chain determined by three parameters: the mean, variance, and one frame-lag correlation ρ. The transition matrix is computed as:

$$P = \rho I + (1 - \rho)Q \tag{8}$$

   where I is the identity matrix, and each row of Q consists of the negative binomial (or gamma) probabilities $(f_0, ..., f_K, F_K)$ where $F_K = \sum_{k>K} f_k$ and K is the peak rate. The DAR(1) model matches the autocorrelation of the data over approximately hundred frame lags and has been shown to be a model of video teleconferences accurate enough for traffic studies [5]. We used video sources with a mean rate of 1.5 Mbps, a peak-to-mean ratio of 5, and $\rho = .98$.
3. CBR sources.
4. Data sources. These were modeled as On-Off sources where the on-time is Pareto distributed, and the off-time is exponential. The density of the Pareto distribution is given by

$$f(x) = a^{k-1}(k-1)/x^k, x >= a, a = (k-2)/(k-1)\mu \tag{9}$$

   We set $k = 5/2$, resulting in infinite variance of the on-times and long-range dependence in the aggregate traffic [12], [22]. The mean on-time was set to 6 packet transmission times and the mean off-time to 200.

The leaky bucket shapers were turned off by setting their token rates equal to the access line rate.

We first compared the performance of FQ-LDF with PGPS (which is simulated exactly) using only one type of source. Table I shows packet loss rates for different small buffer sizes from a simulation with 32 voice sources. All connections were served with equal weight and the utilization is about .98. At this high load, PGPS typically has twice the losses of FQ-LDF and redistribution of excess bandwidth to the longest delay queue reduces the aggregate loss rate. We find similar results of lower loss rates for FQ-LDF with video sources as well. With Pareto sources, the loss rates are mostly similar for FQ-LDF and PGPS. This can be accounted for by the heavy-tailed distributions of the on periods. From this property it follows that the most likely manner in which a large burst in the aggregate over all such sources occurs, is that the burst of one source is far larger than the rest. (Such consequences of underlying infinite variance models have been termed the Joseph and Noah effects; see [14], [15]). Most of the other sources have non-backlogged queues and both schedulers give most of the available bandwidth to the one very long queue. Hence, PGPS and FQ-LDF tend to behave almost identically.

Apart from packet losses, we also compared the delay distribution resulting from FQ-LDF to that of PGPS. For the 32 voice source simulation with large enough buffers to avoid packet losses, Figure 2 shows the tail frequencies (the number of times a given delay is exceeded) on a log scale. The plots show that FQ-LDF needs significantly smaller playout buffers than PGPS. We see the same effect for video sources from the boxplots in Figure 3. The utilization is .88 and the buffers are large enough so that there are no losses. The distribution of mean delays experienced by different connections with FQ-LDF and PGPS is shown in the boxplots by showing the median (the line in the middle of the box) of the means, the quartiles (the upper and lower edges of the box enclosing the median), and by showing the range of values outside the quartiles by the lines extending out from the box. The curtailment of the tail of the delay distribution due to the use of FQ-LDF is evident. Thus, for similar loads FQ-LDF has a lower playout buffer requirement than PGPS.

Next, we compare the performance of FQ-LTO with PGPS. FQ-LDF uses the excess bandwidth to equalize delays and it works very well when all sources have the same allowed delays. When different sources have different maximum allowed delays, the FQ-LTO scheduler which takes delays into account is expected to perform better than FQ-LDF and PGPS. We use 32 sources with 8 sources from each of the types described before. CBR sources are guaranteed their mean rate. They generate at most 2 back-to-back packets and have a small buffer allocation of 5 packets. Voice sources are also guaranteed their mean rate and have a maximum buffer allocation of 1200 packets (about 10 ms delay at their guaranteed rates). Video sources are guaranteed two times their mean rates and have a buffer of 12000 packets. Data sources are given a large buffer of 40000 packets. Since they are not delay sensitive, they are guaranteed a rate below their mean rate. Data sources are expected to use some the excess bandwidth from the other classes to make up for their allocation below their means.

Figure 4 shows the box plots for mean delays and 99^{th} percentile delays for the voice (exponential on-off), video, CBR and data classes. The PGPS scheduler gives excess bandwidths to the data class in proportion to the weights. Since the data class has a very large delay allowance, the FQ-LTO scheduler gives priority to the low delay classes in the distribution of excess bandwidth. Consequently, the lower delay classes have very compact delay distributions whereas the very high delay data class (which also has underallocated guaranteed bandwidth) has better tail behavior with PGPS. The FQ-LTO scheduler achieves what one would want in this mixed scenario. Worst case delay bounds are always met even for data (if it is given appropriate guaranteed rates). The lower delay classes have very small tails and hence need drastically lower playout buffers than when using PGPS.

Finally, we study the situation where we have the same mix of sources as before but their guaranteed rates are different even within a source class. We deliberately set D_is to be below what they would be if allocation was strictly based on deterministic bounds, i.e, we assume that some statistical multiplexing will happen. The question then is how well LTO, which tries to prevent buffer overflows and minimize delays, compares with PGPS. The results are shown in Figure 5. We get very low average delays for real-time sources, in comparison to the delays experienced under PGPS, even when their bandwidth allocations are very low. This is because the D_is are set low for real-time sources, and some of the excess bandwidth which would have gone to data sources under the PGPS scheme is now used to further reduce the delays of real time sources. The differences in delays between LTO and PGPS can be very high as is seen by comparing delays for the voice class. Note that if data traffic losses must be prevented, this can be done by giving those connections an appropriate weight.

V. Discussion and Concluding Remarks

The main contribution of this paper is the modified fair queueing proposal that redistributes excess bandwidth in a state dependent manner. The drawback of GPS that all fair queueing systems inherit in their close emulation of GPS is that GPS severely restricts state-dependent bandwidth sharing. The only state-dependency in GPS is in the number of backlogged connections. There is no further latitude and sharing is determined by the guaranteed rates which are set based on long term needs of the connections. This restriction on bandwidth sharing is more stringent than that necessary to preserve a key property of fair queueing, the ability to guarantee delay bounds for leaky bucket controlled traffic sources. Consequently, for many applications, there is no need for fair queueing systems to emulate the potentially suboptimal excess bandwidth sharing of GPS.

With this in mind, we propose modified fair queueing schemes that emulate fair queueing only in that they provide rate guarantees and behave like fair queueing when all connections are backlogged. This is sufficient to guarantee worst case delay bounds using our scheme. Furthermore, the schemes we propose are worst case fair. Our schemes do not emulate GPS's method for excess bandwidth redistribution. Instead, we proposed three methods (the longest delay first, longest time to overflow, and longest time to overflow taking into account leaky bucket states) for redistributing the excess bandwidth. Simulations show that the FQ-LDF and FQ-LTO policies perform very well in comparison to PGPS. The adaptive bandwidth redistribution reduces packet losses and makes the delay distributions less skewed. The control of delays, in comparison to PGPS, is very significant when very delay sensitive traffic, such as CBR, is mixed with traffic with more laxity in delays. The low delay class has very short tails in comparison to PGPS and hence would need much lower playout buffers. Analytical explanation of our simulation results remains a challenging problem. Unless the fairness requirement of GPS is required for policy reasons, fair queueing systems can improve their performance by adaptive redistribution of excess bandwidth without losing their worst case fairness property or their ability to guarantee worst case delay bounds.

Similar extensions of our scheme can be easily applied to other types of schedulers based either on Earliest Deadline First [8], or on Service Curves [2]. Note, that both of these approaches define how packets must be served in the worst-case so as not to violate some specific delay bounds. However, there is no requirement as to how packets can be served if there is available free bandwidth in order to minimize the tails of the delay distributions. Actually, for both of the above approaches there is no harm if packets are transmitted earlier than expected. Our adaptive bandwidth redistribution techniques only define the method for selecting packets when excess bandwidth is available.

Buffer size in pkts/conn	Cell Loss Ratio:PGPS	Cell Loss Ratio:FQ-LDF
75	13.18e-3	8.3e-3
100	8.198e-3	4.938e-3
125	5.61e-3	3.61e-3
150	4.245e-3	2.849e-3
250	2.061e-3	1.025e-3
350	9.614e-4	1.054e-4
425	4.876e-4	$< 10^{-6}$
500	2.8478e-4	$< 10^{-6}$

TABLE I

CELL LOSS RATIOS FOR 32 EXPONENTIAL ON-OFF SOURCES WITH SHORT BUFFERS

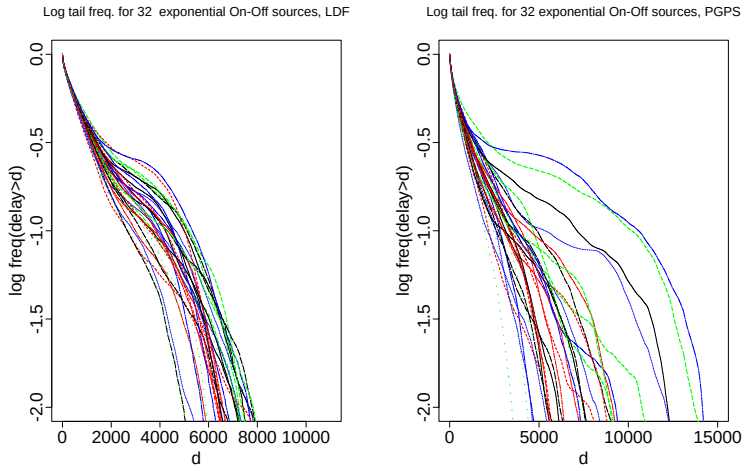

Fig. 2. Log tail frequencies for 32 exponential On-Off sources, according to service discipline (FQ-LDF or PGPS).

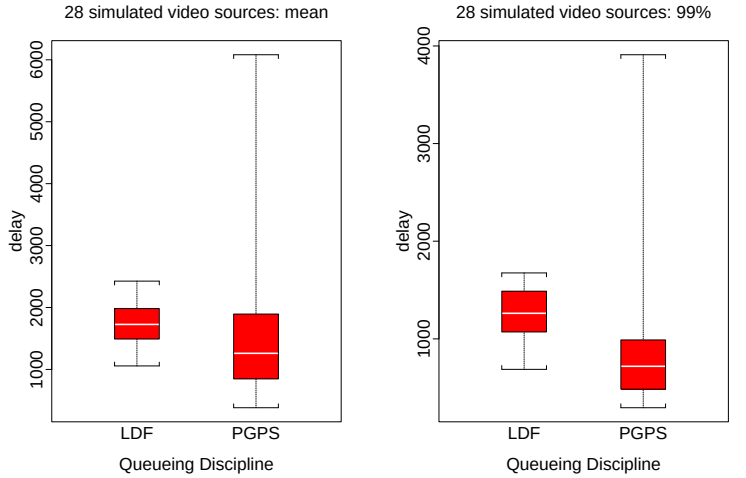

Fig. 3. Boxplots of mean delay (left plot) and 99^{th} percentile delay (right plot) according to excess service discipline (in each plot: FQ-LDF on left, PGPS on right) for 28 simulated video sources. Key: shaded between first and third quartiles; clear bar is median; boundaries at minimum and maximum.

REFERENCES

[1] D. Clark, S. Shenker, L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", *Proceedings of ACM SIGCOMM 1992*, pp. 14-23.

[2] R. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal on Selected Areas In Communications*, vol. 13, pp. 1048–1056, August 1995.

[3] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm" *Internetworking: Research and Experience*, pp, 3-26, vol. 1, 1990.

[4] N. Duffield, T.V. Lakshman, D. Stiliadis "On Adaptive Bandwidth Sharing with Rate Guarantees," Bell Laboratories Technical Memorandum 113470-971215-07TM.

[5] A. Elwalid, D. Heyman, T. V. Lakshman, D. Mitra, A. Weiss, "Fundamental Bounds and Approximations for ATM Multiplexers with Applications to Video Teleconferencing", *IEEE Journal on Selected Areas in Communications: Special Issue on Fundamental Advances in Networking*, pp. 1004-1016, August 1995.

[6] H. Gail, G. Grover, R. Guerin, S. Hantler, Z. Rosberg, M. Sidi, "Buffer

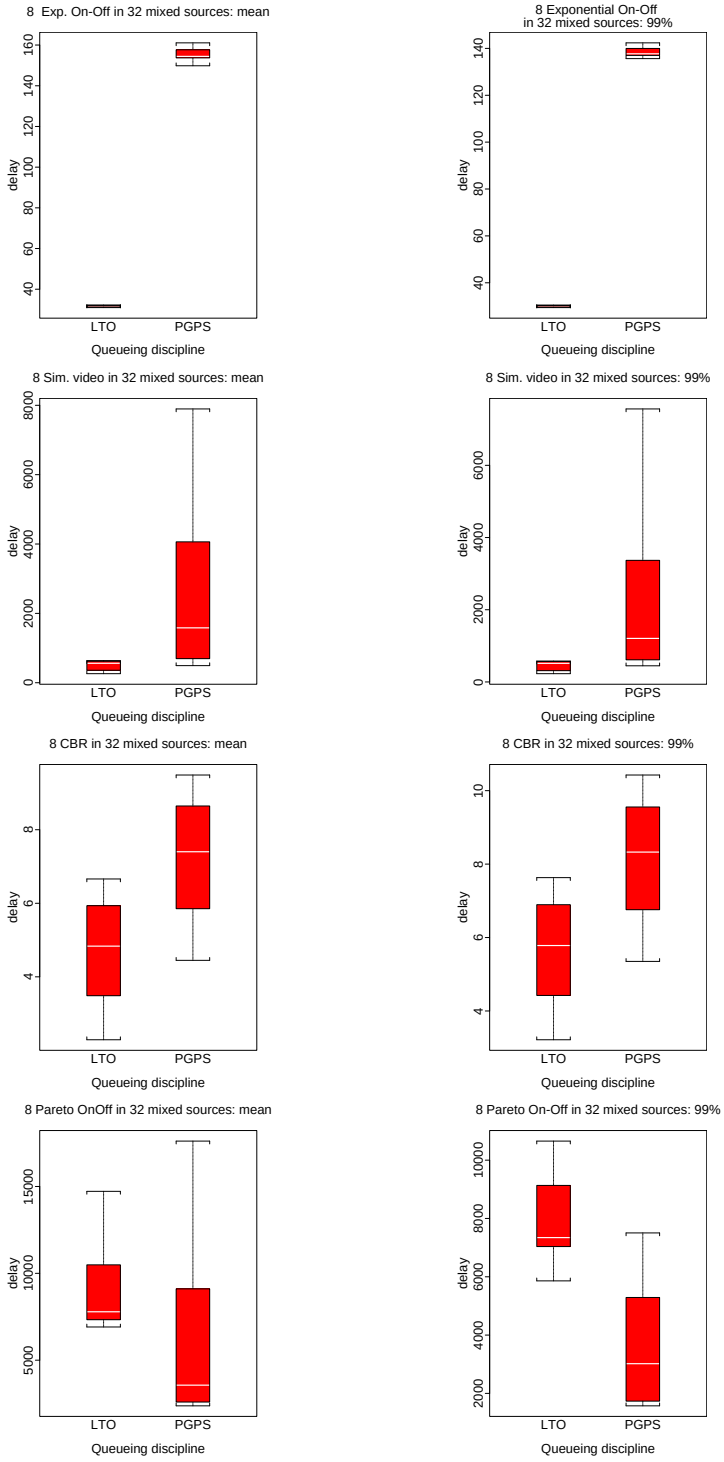

Fig. 4. Boxplots of mean delay (left column) and 99^{th} percentile delay (right column) according to excess service discipline (in each plot: FQ-LTO on left, PGPS on right), displayed by row for each of following 2 classes of sources out of 4 classes being multiplexed: (top to bottom) Exponential On-Off, Simulated Video, CBR, Pareto On-Off. 32 sources total. Key: shaded between first and third quartiles; clear bar is median; boundaries at minimum and maximum.

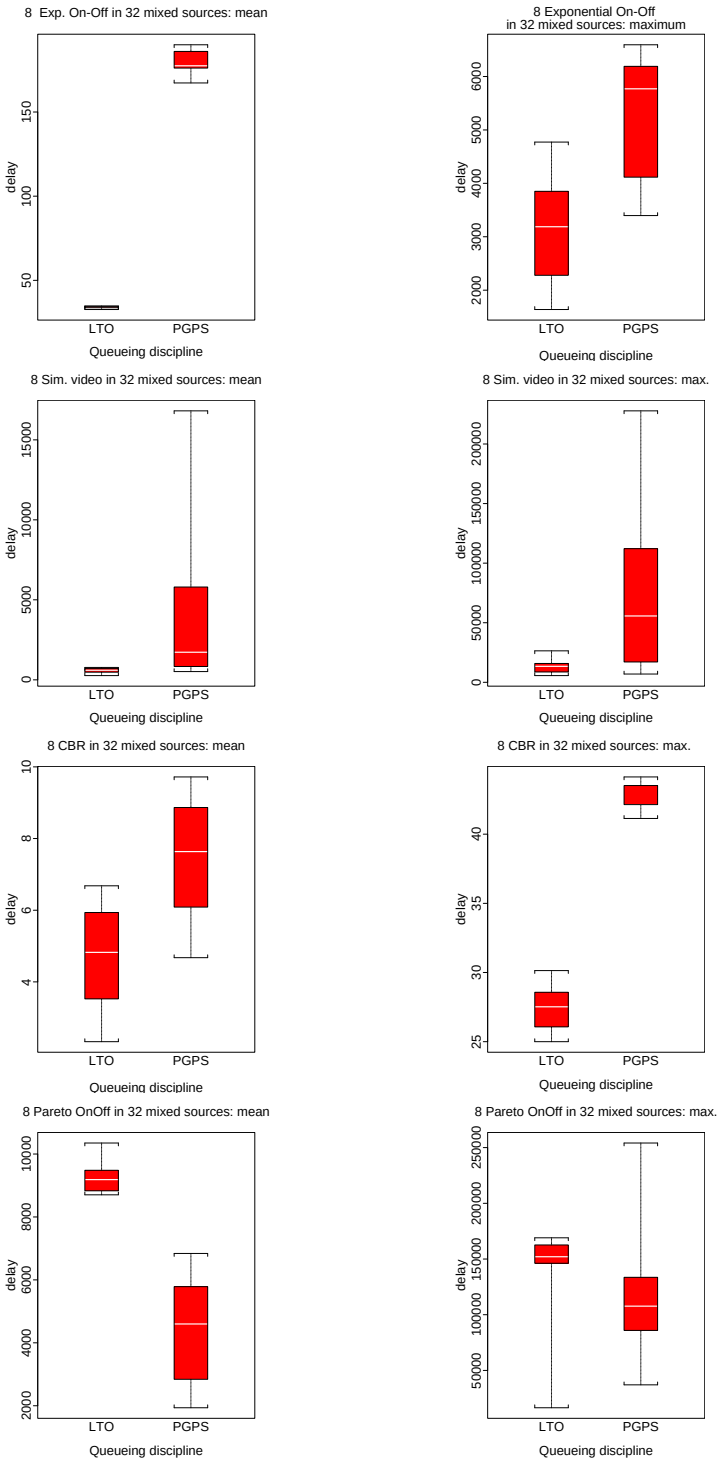

Fig. 5. Boxplots of mean delay (left column) and maximum delay (right column) according to excess service discipline (in each plot: LTO on left, PGPS on right), displayed by row for each of 4 classes of sources: (top to bottom) Exponential On-Off, Simulated Video, CBR, Pareto On-Off. 32 sources total. Key: shaded between first and third quartiles; clear bar is median; boundaries at minimum and maximum.

size requirements under longest queue first", *Proceedings IFIP'92*, 1992
[7] L. Georgiadis, R. Guerin and A. Parekh "Optimal multiplexing on a single link: Delay and buffer requirements", in *Processdings of IEEE INFOCOM'94*, pp.524-532.
[8] L. Georgiadis, R. Guerin, V. Peris and K.N Sivarajan "Efficient network QOS provisioning based on per-node traffic shaping," in *Proceedings of IEEE INFOCOM'96*, pp.102-110.
[9] A. Birman, H.R. Gail, S.L. Hantler and Z. Rosberg, "An optimal service policy for buffer systems," *Journal of the Association for Computing Machinery*, pp. 641-57. vol. 42, no. 3, May 1995.
[10] S. J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications", *Proceedings of INFOCOM'94*, pp. 636-646., 1994.
[11] P. Goyal, H.M. Vin and H. Chen, "Start-time Fair Queing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", *Proceedings ACM SIGCOMM'96*, pp. 157-169.
[12] K. R. Krishnan, "The Hurst Parameter of Non-Markovian On-Off Traffic Sources", Internal Bellcore Report, 1995.
[13] D. S. Lee, "Weighted Longest Queue First: An Adaptive Scheduling Discipline for ATM Networks", *Proceedings INFOCOM'97*, 1997.
[14] B.B. Mandelbrot and J.W. Van Ness (1968). "Fractional Brownian Motions, Fractional Noises and Applications". *SIAM Review*, **10** 422–437
[15] B.B. Mandelbrot and J.R. Wallis (1968). "Noah, Joseph, and Operational Hydrology". *Water Resour. Res.*, **4** 909–918
[16] S. Panwar, D. Towsley, J. Wolf, "Optimal scheduling policies for a class of queues with customer deadlines to the beginning of services", *Journal of ACM*, vol. 35, no. 4, pp. 832—844, 1988.
[17] A. K. Parekh R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case", *Proceedings of INFOCOM'93*, pp. 521-530, 1993.
[18] J.L. Rexford, A.G. Greenberg, F.G Bonomi, "Hardware Efficient Fair Queueing Architectures for High-Speed Networks", *Proceedings of INFOCOM'96*, pp. 120-128, 1996.
[19] D. Stiliadis, A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms", *Proceedings of INFOCOM'96*, pp. 111-119, 1996.
[20] D. Stiliadis and A. Varma, "Design and analysis of Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks", *Proceedings of ACM SIGMETRICS '96*, pp. 104-115, May 1996.
[21] S. Suri, G. Varghese and G. Chandramenon, "Leap Forward Virtual Clock: A New Fair Queuing Scheme with Guaranteed Delays and Throughput Fairness" *Proceedings INFOCOM'97*, 1997.
[22] W. Willinger, M.S. Taqqu, R. Sherman, D.V. Wilson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", *Proceedings of ACM SIGCOMM 1995*.
[23] D. Yates, J. Kurose, D. Towsley, M.G. Hluchyj, "On per-session end-to-end delay distributions and the call admission problem for real-time applications with QoS requirements", *Proceedings of ACM SIGCOMM 1993*, pp.2-12, September 1993.
[24] J.C.R. Bennett, H. Zhang, "WF^{2}Q: Worst-case Fair Weighted Fair Queueing", *Proceedings of INFOCOM'96*, pp. 120-128, 1996.

APPENDIX

WORST-CASE FAIRNESS OF FQ-SDS

Let us denote by t_k, the times that a packet is released by the shaper to the RPS scheduler. In order to be able to calculate an exact bound for the worst-case fairness we need to account for the fluid server that the RPS system is simulating. Let us assume that $W_i^F(\tau,t)$ is the service offered by this fluid to server to connection i during an interval $(\tau,t]$. We will first prove the following lemma:

Lemma 2: Let $W_i^F(\tau,t)$ be the service received by session i in the fluid RPS system during an interval $(\tau,t]$ in which it is continuously backlogged. If at time τ a packet was released from the shaper then,

$$W_i^F(\tau,t) \geq g_i(t-\tau).$$

Proof: We know that since the beginning of the system busy period the shaper releases packets with rate equal to g_i. We also know that the connection virtual time is increasing by the normalized service offered to the connection, when the connection is not idle. Otherwise, it increases based on the system virtual time. Let us denote with $t' < \tau$, the last time that connection virtual time was updated by the use of the system virtual time. Thus, for time t',

$$v(t') = v_i(t') \tag{10}$$

Obviously, this happened after an idle period of connection, where the connection virtual time has remained behind the system virtual time. This update was initiated by a packet arrival. That means that at time t' a packet was released from the shaper. We know that the total arrivals to the RPS scheduler since time t' are bounded by

$$S_i(t',\tau) \leq g_i(\tau - t') \tag{11}$$

We also know that since time t' the connection virtual time has only increased by the normalized service offered to it. Thus,

$$\begin{aligned} v_i(\tau) - v_i(t') &= \frac{W_i^F(t',\tau)}{g_i} && (12)\\ &\leq \frac{S_i(t',\tau)}{g_i} \quad \text{From Eq.11} && (13)\\ &\leq (\tau - t') && (14)\end{aligned}$$

We also know that the system virtual time is increasing with a rate at least linear to the real time, or

$$v(\tau) \geq v(t') + (\tau - t'). \tag{15}$$

From Eq. (10),(14) and (15) we can conclude that

$$v_i(\tau) \leq v(\tau) \tag{16}$$

Thus when a packet is released from the shaper at time τ, the connection virtual time $v_i(\tau)$ will be updated to a value no larger to that of the system virtual time. During the interval $(\tau,t]$, the connection virtual time is increasing by the normalized service offered to that connection. Thus,

$$v_i(t) - v_i(\tau) = \frac{W_i^F(\tau,t)}{g_i}. \tag{17}$$

Therefore,

$$W_i^F(\tau,t) = g_i(v_i(t) - v_i(\tau)). \tag{18}$$

Since $v_i(t) \geq v(t)$ and $v_i(\tau) \leq v(\tau)$, we can re-write this as

$$W_i^F(\tau,t) \geq g_i(v(t) - v(\tau)). \tag{19}$$

During the interval $(\tau,t]$, the system potential is increasing with a rate at least equal to that of real time. That is,

$$v(t) - v(\tau) \geq t - \tau. \tag{20}$$

From equations (19) and (20), we have $W_i^F(\tau,t) \geq g_i(t-\tau)$. □ We can now evaluate the worst-case fairness index of the packet-based system.

Theorem 1: The queueing delay of a session-i packet arriving at time τ in a packet FQ-SDS system is upper-bounded by

$$d_i^P(\tau) \leq \frac{Q_i^P(\tau)}{g_i} + \frac{L_{max}}{r} + L_i\left(\frac{1}{g_i} - \frac{1}{r}\right), \tag{21}$$

where $Q_i^P(\tau)$ is the backlog of session i at time τ, L_i the maximum packet size of session i, and L_{max} the maximum packet size across all sessions.

Proof: Let us denote with t_k the time at which the k-th packet of session i is released from the shaper to the scheduler. Note that this time is identical in both the fluid and packet versions considered. We will prove the theorem for each instant t in the interval $(t_k, t_{k+1}]$. Let, $Q_i^F(t)$ be the backlog in the fluid server at time t and $Q_i^P(t)$ that in the packet-by-packet server. By Lemma 2, the service offered to session i after time t_k is with a rate at least equal to g_i, and without any latency.

Let us assume that the backlog of connection i is cleared at some time t^* in the fluid server. Then we can write

$$\begin{aligned} Q_i^F(t) &= W_i^F(t,t^*) \\ &= W_i^F(t,t_{k+1}) + W_i^F(t_{k+1},t^*) \\ &\geq W_i^F(t,t_{k+1}) + \max(g_i(t^* - t_{k+1}),0) && (22)\end{aligned}$$

We assume that if $t^* < t_{k+1}$, then $W_i^F(t_{k+1},t^*) = 0$. We can re-write Eq. (22) as

$$\begin{aligned} t^* &\leq t_{k+1} + \frac{Q_i^F(t) - W_i^F(t,t_{k+1})}{g_i} \\ &\leq (t_{k+1} - t) + t + \frac{Q_i^F(t) - W_i^F(t,t_{k+1})}{g_i}. && (23)\end{aligned}$$

The above equation holds for the fluid sever. The time to clear the backlog at time t in the fluid server is thus given by $d_i^F(t) = t^* - t$. The packet sever may lag the fluid server at most by the transmission time of one packet [19]. Therefore, the time $d_i^v(t)$ needed to clear the backlog of connection i in the packet-by-packet server after time t is bounded by

$$\begin{aligned} d_i^v(t) &\leq d_i^F(t) + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^F(t) - W_i^F(t, t_{k+1})}{g_i} + (t_{k+1} - t) + \frac{L_{max}}{r} \end{aligned} \tag{24}$$

To evaluate the above expression, we will consider two separate cases.
Case 1: $Q_i^v(t) \geq Q_i^F(t)$. Let L_i^k denote the size of the kth packet of session i. Then,

$$\begin{aligned} W_i^F(t, t_{k+1}) &= \frac{L_i^k}{g_i} - W_i^F(t_k, t) \\ &\geq \frac{L_i^k}{g_i} - \min(L_i^k, r(t - t_k)). \end{aligned} \tag{25}$$

From Eq. (24) and (25) we can write

$$d_i^v(t) \leq (t_{k+1} - t) + \frac{Q_i^F(t)}{g_i} - \frac{L_i^k}{g_i} + \min(L_i^k, r(t - t_k)) + \frac{L_{max}}{r}. \tag{26}$$

By using the hypothesis that $Q_i^P(t) \geq Q_i^F(t)$, this becomes

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t)}{g_i} - \frac{L_i^k}{g_i} + \frac{\min(L_i^k, r(t - t_k))}{g_i} + \frac{L_{max}}{r}. \tag{27}$$

The right-hand side of the above equation is maximized when $L_i^k = r(t - t_k)$. Thus,

$$\begin{aligned} d_i^P(t) &\leq (t_{k+1} - t_k - \frac{L_i^k}{r}) + \frac{Q_i^P(t)}{g_i} - \frac{L_i^k}{g_i} + \frac{L_i^k}{g_i} + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^P(t)}{g_i} + L_i^k(\frac{1}{g_i} - \frac{1}{r}) + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^P(t)}{g_i} + L_i(\frac{1}{g_i} - \frac{1}{r}) + \frac{L_{max}}{r}. \end{aligned} \tag{28}$$

Case 2: $Q_i^F(t) > Q_i^P(t)$. In this case, let $\Delta Q = Q_i^F(t) - Q_i^P(t)$. We can re-write Eq.(24) as

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t) + \Delta Q}{g_i} - \frac{W_i^F(t, t_{k+1})}{g_i} + \frac{L_{max}}{r}. \tag{29}$$

At time t, the packet-by-packet server has offered more service to connection i than the fluid server. However, packet $k+1$ has not yet been released by the shaper. Thus, the additional service that fluid server has to offer until time t_{k+1} is equal to the additional service that the packet-by-packet server has offered until time t, plus the service it will offer until time t_{k+1}. That is,

$$W_i^F(t, t_{k+1}) = \Delta Q + W_i^P(t, t_{k+1}). \tag{30}$$

From Eq. (29) and (30),

$$d_i^P(t) \leq (t_{k+1} - t) + \frac{Q_i^P(t)}{g_i} - \frac{W_i^P(t, t_{k+1})}{g_i} + \frac{L_{max}}{r}. \tag{31}$$

Notice that at time t_k, the packet-by-packet server can only be behind the fluid server. Let us assume without loss of generality, that

$$W_i^F(0, t_k) = W_i^P(0, t_k) + \Delta W. \tag{32}$$

For the service offered to connection i by the packet-by-packet server after time t, we can write

$$\begin{aligned} W_i^P(t, t_{k+1}) &= W_i^P(t_k, t_{k+1}) - W_i^P(t_k, t) \\ &\geq L_i^k + \Delta W - W_i^P(t_k, t) \\ &\geq L_i^k + \Delta W - \min(L_i + \Delta W, r(t - t_k)). \end{aligned} \tag{33}$$

Thus, Eq. (31) becomes

$$\begin{aligned} d_i^P(t) &\leq (t_{k+1} - t) + \frac{Q_i^P(t)}{g_i} \\ &\quad -(L_i + \Delta W) + \min(L_i^k + \Delta W, r(t - t_k)) + \frac{L_{max}}{r}. \end{aligned} \tag{34}$$

The right-hand side of the above equation is maximized when $L_i + \Delta W = r(t - t_k)$. Therefore,

$$\begin{aligned} d_i^P(t) &\leq t_{k+1} - t_k - \frac{L_i^k + \Delta W}{r} + \frac{Q_i^P(t)}{g_i} + \frac{L_{max}}{r} \\ &\leq \frac{L_i^k}{g_i} - \frac{L_i^k + \Delta W}{r} + \frac{Q_i^P(t)}{g_i} + \frac{L_{max}}{r} \\ &\leq \frac{Q_i^P(t)}{g_i} + \frac{L_{max}}{r} + L_i^k(\frac{1}{g_i} - \frac{1}{r}) \\ &\leq \frac{Q_i^P(t)}{g_i} + \frac{L_{max}}{r} + L_i(\frac{1}{g_i} - \frac{1}{r}). \end{aligned} \tag{35}$$

This concludes the proof of Theorem 1. Note that the worst-case fairness index of $(L_{max}/r) + L_i/g_i - L_i/r$ given by the theorem is identical to that of WF2Q [24].