Fair Queuing for Aggregated Multiple Links

Josep M. Blanquer*
Department of Computer Science
University of California Santa Barbara
blanquer@cs.ucsb.edu

Banu Özden
Bell Laboratories
Lucent Technologies
ozden@research.bell-labs.com

ABSTRACT

Provisioning of a shared server with guarantees is an important scheduling task that has led to significant work in a number of areas including link scheduling. Fair Queuing algorithms provide a method for proportionally sharing a single server among competing flows, however, they do not address the problem of sharing multiple servers. Multi-server systems arise in a number of applications including link aggregation, multiprocessors and multi-path storage I/O. In this paper we introduce a new service discipline for multi-server systems that provides guarantees for competing flows. We prove that this new service discipline is a close approximation of the idealized Generalized Processor Sharing (GPS) discipline. We calculate its maximum packet delay and service discrepancy with respect to GPS. We also discuss its relevance to several applications, in particular, Ethernet link aggregation.

1. INTRODUCTION

A large increase in networked services has been gradually driving packet-switched networks to carry a much larger variety of traffic. These classes of traffic range from simple downloads of static web pages or file transfers to multimedia streams and real-time trading. This increased variety is challenging the premises of the Internet's best-effort traffic, and demands different network requirements to be met simultaneously over the same links. An example of this heterogeneity would be a network that must simultaneously provide high bandwidth, low jitter and packet delay guarantees to ensure the correct performance of continuous backups, video streaming and network data acquisition applications, respectively. In order to meet these diverse requirements, network resources must be appropriately scheduled.

Fair Queuing service disciplines address this scheduling problem by allocating bandwidth fairly among competing traffic, regardless of their prior usage or congestion. In particular, these disciplines do not penalize traffic for the use of idle bandwidth. Fair queuing algorithms are based on the Generalized Processor Sharing (GPS), an idealized system that serves as a reference model for the fair queuing disciplines. GPS-based service disciplines are studied in the context of providing fairness as well as more strict Quality of Service (QoS) guarantees. Fairness offers protection from "misbehaving" traffic and leads to effective congestion control and better services for rate-adaptive applications. Strict QoS guarantees such as throughput or delays can also be provided by restricting the admission of traffic. In [15], it is demonstrated that GPS guarantees end-to-end delay for leaky-bucket constrained traffic. A number of approximations and heuristics of GPS were devised over the years [8, 17, 11, 15, 9, 5, 10]. Implementations of this work, known as Weighted Fair Queuing, can be found in current commercial routers or switches as well as in some servers [6] which provide differentiated qualities of service to distinct classes of clients.

An increased dependence on network services and the growing demand for bandwidth have generated the need for incremental scaling techniques. Grouping multiple links into a single logical interface has emerged as a popular bandwidth scaling method for high throughput switches and servers [3]. Numerous implementations of aggregation techniques between servers, routers and switches are currently deployed in industry [2, 16, 1, 12]. These existing implementations already provide different techniques for load balancing the traffic among the interfaces but none of them address the provision of QoS over these aggregated links.

Although GPS based service disciplines are extensively studied for scheduling a single link, they have not been applied to aggregated links. The provisioning of such systems is naturally described as a function of the total link capacity rather than for each of the links. This calls for a reference system that consists of a single GPS server operating at a rate equal to the sum of the the underlying servers' rates. In this paper, we study how packetized service disciplines with multiple servers can closely approximate such a GPS reference system. Many of the fair queuing results that were previously obtained for single server systems do not directly apply to multi-server systems. This is because the rate at which the packetized multi-server system operates may vary over time and thus differ from the rate of the reference system. Furthermore, the packetized multi-server system may reorder the packets to remain work-conserving.

The remainder of the paper is organized as follows. Section 2 will give some background on the Generalized Processor Sharing discipline. Section 3 will describe the sin-

*Work done as a summer intern at Bell Labs.

SIGCOMM'01, August 27-31, 2001, San Diego, California, USA..

gular properties of the multi-server disciplines. Section 4 and 5 will formally prove the maximum differences in packet departure and per-flow service discrepancy with respect to GPS. Section 6 will evaluate the fairness of the discipline and propose MSF^{2}Q , a fair queuing algorithm for multiple servers. In section 7 we will introduce some applications that closely follow the presented sharing model and describe how they can benefit from the implementation of the new disciplines. Section 8 will outline some of the related work that has been done in the area and in Section 9 we summarize our conclusions and present future work.

2. BACKGROUND

Generalized Processor Sharing (GPS) is a service discipline defined for sharing a server proportionally among a set of flows [15]. A GPS server operates at a fixed rate r and is work-conserving. A positive real number ϕ_i is assigned for each flow i. Let $\mathcal{F}$ denote the set of flow indices. At any given time, a flow is either backlogged or idle. A flow is backlogged at time t if some of the flow's traffic is queued at time t. Otherwise, the flow is idle. Let $W_i(\tau, t)$ be the amount of traffic for flow i served in the interval $[\tau, t]$. Then, a GPS server is defined as one for which

$$\frac{W_i(\tau,t)}{W_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}, j \in \mathcal{F} \quad (1)$$

holds for any flow i that is continuously backlogged during the interval $[\tau, t]$.

The weight of a flow determines the proportion of the server bandwidth that a flow receives when it is backlogged. During any time interval $[\tau, t]$ when the set of backlogged flows, denoted by $\mathcal{F}(\tau, t)$, is unchanged, a GPS server guarantees to a flow i, $i \in \mathcal{F}(\tau, t)$, a rate of $\frac{\phi_i}{\sum_{j\in\mathcal{F}(\tau,t)}\phi_j} r$. We denote the instantaneous rate of a flow i by $r_i(t)$.

If we are to provide strict QoS guarantees, then an admission mechanism is required so as to limit access and bandwidth shares. For example, by fixing the set of flows, a GPS server can guarantee to each flow i a minimum service rate of r_i:

$$r_i = \frac{\phi_i}{\sum_{j\in\mathcal{F}}\phi_j} r.$$

GPS is an idealized discipline that cannot be implemented since it assumes that the server transmits more than one flow simultaneously and that the traffic is infinitely divisible. GPS serves as a model for sharing a server among flows with respect to their weights. A number of packetized approximations to GPS have been devised [8, 17, 15, 9, 5, 10].

3. PROPORTIONAL SHARING OF MULTI-SERVER SYSTEMS

In this paper, we study proportional sharing of systems with multiple servers. There are numerous applications utilizing multi-server systems that can benefit from service guarantees. For example, the use of multiple network adapters for connecting a web or file server to a switch is becoming increasingly popular. Similarly, attaching a host to a RAID server via multiple I/O channels is emerging as the prevalent approach to increase I/O bandwidth between hosts and

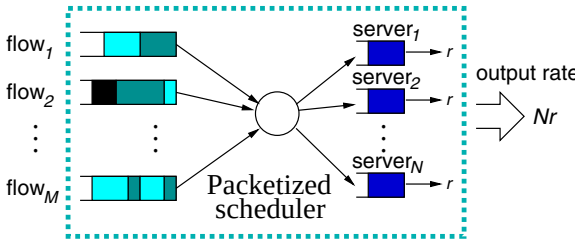

Figure 1: Packetized model for multiple servers.

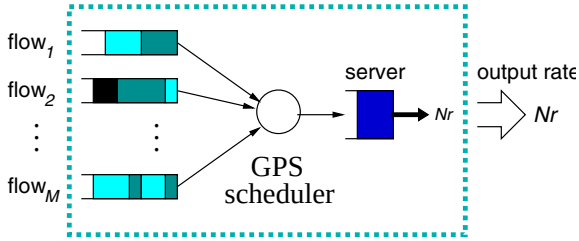

Figure 2: GPS model for multiple servers.

RAID servers. Such network and storage connections can be modeled as a packet system with multiple servers.

The problem of sharing multiple servers can be approached by partitioning the flows among the servers and scheduling them separately within each partition. One of the disadvantages of this technique is that bandwidth fragmentation can easily occur when the sum of the flow weights is not balanced across all partitions. Moreover, aside from the fragmentation problem, this technique also has drawbacks in handling sporadic flows. For example, it is quite common for a large number of applications to frequently switch flows between backlogged and idle states or to make extensive use of relatively short-lived connections. This partitioning approach is also cumbersome to deal with in the case where weight assignments result in bandwidth shares for a flow that exceed the rate of a single server. In this paper, we concentrate on an alternative approach to sharing multi-servers where a packet of any flow can be serviced at any of the servers.

Our system model consists of N servers each of which operates at a fixed rate of r. Our goal is to study packetized scheduling disciplines for multi-server systems that closely approximate the ideal case of a single GPS server with a rate of Nr (see Figures 1 and 2). We will refer to this GPS server as a $(GPS, 1, Nr)$ system denoting 1 server with an output rate of Nr being scheduled with the GPS discipline. Comparing the packetized disciplines against such a system allows the flows to be guaranteed a proportion of the total server capacity regardless of the value of N. This allows the proportions to remain valid without intervention when increasing the number of servers in the packetized system. For example, adding new interfaces to the link aggregation group of a high throughput web server will not change the proportions in which the different classes of services are served and will allow for the expansion of their minimum guaranteed rates.

As is customary in related literature, we assume that the arrival process to the packetized scheduling discipline is identical to that of the GPS discipline. We denote the arrival time of a packet p by a_p.

3.1 A Packetized Fair Queuing Discipline for Multi-Servers

In this section, we investigate the use of the WFQ packetized fair queuing service discipline, which is defined for a single server in [8, 15], in a multi-server system consisting of N servers each with a rate of r. We refer to such a system as a $(MSFQ\ , N, r)$ system. In the remainder of the paper, we will use GPS and MSFQ systems/servers to denote the $(GPS, 1, Nr)$ and $(MSFQ\ , N, r)$ systems respectively, without explicitly stating their number of servers and their rate.

When a server is idle and there is a packet waiting for service, MSFQ schedules the "next" packet. The "next" packet is defined as the first packet that would complete service in the the $(GPS, 1, Nr)$ system if no more packets were to arrive.

We compare how well a $(MSFQ\ , N, r)$ system approximates a $(GPS, 1, Nr)$ system. To do so, we calculate: i) the worst case delay that a packet experiences under MSFQ relative to GPS and ii) the discrepancy between the amount of traffic served for a flow under MSFQ and the amount under GPS. In order to prove these two quantities, we first need to establish a number of properties of MSFQ .

3.2 Preliminary Properties

Although MSFQ and its single-server counterpart WFQ are both based on the same policy for selecting the next packet to be serviced, MSFQ does not share some of the useful properties of WFQ. As a result, delay and service properties of MSFQ do not trivially follow from the single server case.

The first obstacle pertains to the busy periods of MSFQ with respect to GPS. While WFQ busy periods coincide with those of GPS, this property does not hold for MSFQ . To illustrate this, take the case of a busy period consisting of the transmission of a single packet. While GPS will be able to transmit the packet at full rate, Nr, the MSFQ server will only be able to use one of its N servers so the packet would be transmitted at a rate of r. In this case, by the time GPS has finished the job (end of GPS busy period), the MSFQ server still has the last $\frac{(N-1)L}{N}$ last bits of the packet left to transmit.

When GPS is busy, MSFQ is busy. However, the converse is not true. Thus for any τ,

$$W(0,\tau) \geq \bar{W}(0,\tau), \tag{2}$$

where $W(0,\tau)$ and $\bar{W}(0,\tau)$ denote the total number of bits serviced by GPS and MSFQ , respectively, by time τ. Since GPS and MSFQ busy periods do not coincide, in order to simplify the presentation, we will use the term *busy period* to refer to a busy period in the reference $(GPS, 1, Nr)$ system.

Furthermore, because they do not coincide, work from previous busy periods can accumulate under MSFQ . This may happen either at the beginning or in the middle of a busy period. Figure 3 depicts a case in which the backlog is being accumulated in both cases. In this example the packets arrive sequentially to the system such that there is always one packet at the GPS server being transmitted at full rate. This example raises the need to investigate whether the amount of work accumulating at MSFQ is bounded. The following theorem shows that such a backlog is indeed bounded. Let L_{max} denote the maximum packet length.

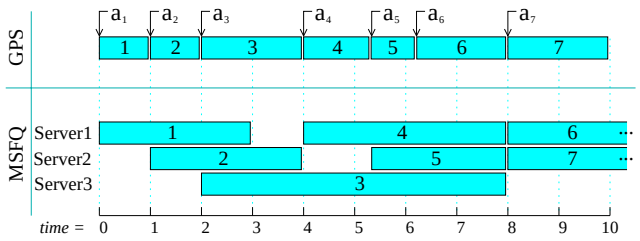

Figure 3: Example of backlog accumulation in multi-server packetized disciplines.

Theorem 1: For any τ,

$$W(0,\tau) - \bar{W}(0,\tau) \leq (N-1)L_{max}.$$

Proof: The slope of W alternates between Nr (when a busy period resumes) and 0 (between two consecutive busy periods). Since the slope of $\bar{W}$ is at most Nr at any given time, the difference $W(0,\tau) - \bar{W}(0,\tau)$ reaches its maximal value when a busy period ends under GPS. Let t be such time.
Case 1: At most $N-1$ MSFQ servers are busy at t: Since MSFQ is work-conserving, if a server is idle, we know that there is no packet waiting for transmission. Let k be the number of servers that are busy at time τ, $0 \leq k \leq N-1$. In the worst case, all the k servers have just started transmitting a packet of maximum length. Thus,

$$W(0,t) - \bar{W}(0,t) \leq kL_{max}.$$

Case 2: All MSFQ servers are busy at t: Let $[t_o, t]$ be the largest interval in which all MSFQ servers are busy. Since in the interval $[t_o, t]$ the slope of $\bar{W}$ is Nr,

$$W(0,t) - \bar{W}(0,t) \leq W(0,t_o) - \bar{W}(0,t_o).$$

If $t_o = 0$, then

$$W(0,t) = \bar{W}(0,t).$$

Otherwise, if $t_o > 0$, we know from Case 1,

$$W(0,t_o) - \bar{W}(0,t_o) \leq (N-1)L_{max}.$$

Thus, the theorem follows. □

This theorem is also important in order to calculate buffer requirements for multi-server systems. For example, buffer requirements of a GPS system servicing leaky-bucket shaped flows are studied in [15]. To provide similar guarantees to such flows under a multi-server packet system, this theorem implies the need for a buffer space of $(N-1)L_{max}$.

Another difference between multi-server and single-server schedulers is the discrepancy of packet departure times with respect to GPS. Let d_p be the time at which packet p departs from $(GPS, 1, Nr)$ system. MSFQ packets may not depart in increasing order of d_p. The order in which packets depart under MSFQ may be different than the order in which MSFQ schedules (i.e., begins transmitting/servicing) packets. This is because packets of a flow may be concurrently in service at different servers of MSFQ . This type of reordering does not occur in the single-server case. A second reason for reordering is due to "late" arrival of packets. Suppose that a server becomes idle at time τ. The next packet to depart under GPS may not have arrived at time τ. Since the server has no knowledge of when this packet will arrive, MSFQ cannot be both work conserving and also schedule

packets always in increasing order of d_p. This type of reordering also exists in the single-server packetized systems but the problem is intensified in the multi-server case.

We will use the following lemma in a number of theorems.

Lemma 1: Let a_k and b_k, be respectively the arrival time and scheduling time of packet k under any work conserving service discipline over N servers each with a rate of r. Let P be the set of packets scheduled before packet k since time a_k, including the packets in service at a_k. Packet k will be scheduled no later than:

$$b_k \leq a_k + \frac{\sum_{i \in P} L_i}{Nr}.$$

Proof: Given a load that must be scheduled before packet k, a work conserving service discipline schedules packet k latest, if the load is equally divided among the N servers such that all of them finish the work at the same time. □

Having established these new multi-server features, we are now ready to prove the two main properties of the new algorithm: maximum packet delay and per flow service discrepancy.

4. PACKET DELAY

Let $\bar{d}_p$ be the time at which packet p departs from the the $(MSFQ\ , N, r)$ system. Let L_{max} denote the maximum packet length. The following scenario is possible.

Example 1: All the N servers are idle before time t. N packets of flow 1, each with a length L_{max}, arrive at time t. Packet p of flow 2 arrives immediately after t. Let $\phi_2 >> \phi_1$. Thus, d_p is slightly after $a_p + \frac{L_p}{Nr}$, where L_p is the length of packet p. However, $\bar{d}_p$ is slightly before $a_p + \frac{L_{max}}{r} + \frac{L_p}{r}$. This is because when packet p arrives, each server under MSFQ is transmitting a packet, which arrived before packet p whose GPS finishing time is after d_p. Thus, $\bar{d}_p - d_p$ is close to $\frac{(N-1)L_p}{Nr} + \frac{L_{max}}{r}$. □

The next theorem shows that the case in the above example is indeed the worst case delay a packet experiences under MSFQ compared to GPS.

Theorem 2: For all packets p,

$$\bar{d}_p - d_p \leq \frac{(N-1)L_p}{Nr} + \frac{L_{max}}{r}.$$

Proof: Let p_k be the k^{th} packet that is scheduled under MSFQ , and a_k, $\bar{b}_k$ and $\bar{d}_k$ respectively be the arrival time, scheduling time and the departure time of p_k under MSFQ . We now show that

$$\bar{d}_k \leq d_k + \frac{L_{max}}{r} + \frac{(N-1)L_k}{Nr}$$

for $k = 1, 2, \cdots$.

Case 1: There is an interval $[t, \bar{b}_k]$, $t < \bar{b}_k$, in which at least one MSFQ server is continuously idle: If one of the MSFQ servers is idle before and until MSFQ schedules p_k, then p_k must have arrived at that moment. That is, $\bar{b}_k = a_k$. This is because MSFQ is work conserving. In this case, $d_k \geq a_k + \frac{L_k}{Nr}$ and $\bar{d}_k = a_k + \frac{L_k}{r}$. Thus,

$$\bar{d}_k \leq d_k + \frac{(N-1)L_k}{Nr}.$$

Case 2: There is an interval $[t, \bar{b}_k]$, $t < \bar{b}_k$, in which all MSFQ servers are continuously busy: Let g be the smallest integer, such that all MSFQ servers are busy between $\bar{b}_g$ and $\bar{b}_k$. Let m be the largest integer greater than or equal to g that satisfies both $g \leq m \leq k-1$ and $d_m > d_k$. Thus, for $m < i < k$,

$$d_m > d_k \geq d_i$$

Then, packet p_m is scheduled before packets $p_{m+1}, \cdots, p_k$ under MSFQ but p_m departs after all these packets under GPS.

Case 2.1: No such integer m exists: $p_g, p_{g+1}, \cdots, p_{k-1}$ are all scheduled before p_k under MSFQ and all leave the GPS server before p_k. Since MSFQ is work conserving, p_g must have arrived at $\bar{b}_g$. Since MSFQ schedules packets p_{g+1}, p_{g+2}, $\cdots$, p_k after p_g and is work conserving, $p_{g+1}, p_{g+2}, \cdots, p_k$ arrive at or after $\bar{b}_g$. Thus, $a_g \leq a_i$ for $g < i \leq k$. This implies

$$d_k \geq a_g + \sum_{i=g}^{k} \frac{L_i}{Nr}. \tag{3}$$

Under MSFQ , at time a_g at most $N-1$ other packets can be in service. Thus, before scheduling p_k, MSFQ must service at most $N-1$ packets of length L_{max} and $p_g, p_{g+1}, \cdots, p_{k-1}$. From Lemma 1, the earliest time MSFQ will schedule p_k is $\bar{b}_k \leq a_g + \frac{(N-1)L_{max} + \sum_{i=g}^{k-1} L_i}{Nr}$. Thus,

$$\bar{d}_k \leq a_g + \frac{(N-1)L_{max}}{Nr} + \sum_{i=g}^{k-1} \frac{L_i}{Nr} + \frac{L_k}{r}. \tag{4}$$

Equations 3 and 4 imply

$$\bar{d}_k \leq d_k + \frac{(N-1)L_{max}}{Nr} + \frac{(N-1)L_k}{Nr}.$$

Case 2.2: Such an integer m exists: Packet p_m begins transmission under MSFQ at $\bar{b}_m = \bar{d}_m - \frac{L_m}{r}$. Under GPS, the relative order, in which two packets p and p' depart, is independent of the other packets arriving after $\max\{a_p, a_{p'}\}$. Since MSFQ schedules packets in the increasing order of GPS packet departure times and since the departure time of p_m under GPS is greater than the departure times of $p_{m+1}, \cdots, p_k$,

$$\min\{a_{m+1}, ..., a_k\} > \bar{d}_m - \frac{L_m}{r}.$$

Since $p_{m+1}, \cdots, p_{k-1}$ arrive after $\bar{d}_m - \frac{L_m}{r}$ and depart before p_k does under GPS,

$$d_k \geq \frac{\sum_{i=m+1}^{k} L_i}{Nr} + \bar{d}_m - \frac{L_m}{r} \tag{5}$$

Since MSFQ decides to service p_m at $\bar{b}_m$, it must have served each packet that arrives before $\bar{b}_m$ and departed under GPS before d_m. Out of these packets, at most $N-1$ of them with a length L_{max} may be still in service at time $\bar{b}_m$. Thus, after $\bar{b}_m$ and before scheduling p_k, MSFQ must service at most $N-1$ packets of length L_{max} and $p_m, p_{m+1}, \cdots, p_{k-1}$. From Lemma 1, the earliest time MSFQ will schedule p_k is $\bar{b}_k \leq \bar{b}_m + \frac{(N-1)L_{max} + L_m + \sum_{i=m+1}^{k-1} L_i}{Nr}$. Thus,

$$\bar{d}_k \leq \frac{(N-1)L_{max} + L_m}{Nr} + \frac{\sum_{i=m+1}^{k-1} L_i}{Nr} + \frac{L_k}{r} + \bar{d}_m - \frac{L_m}{r} \tag{6}$$

Equations 5 and 6 imply

$$\bar{d}_k \leq d_k + \frac{L_{max}}{r} + \frac{(N-1)L_k}{Nr}$$

□

5. SERVICE PER-FLOW

Let $W_i(t,\tau)$ and $\bar{W}_i(t,\tau)$ be the amount of service (in bits) that flow i received in the interval $[t,\tau]$ under GPS and MSFQ respectively.

Example 2: Consider the scenario depicted in Example 2 with the following arrival pattern for flow 2. N packets of flow 2 each with length L_{max} arrives slightly after t. Since N servers of MSFQ are idle at t, we know that $W_i(0,t) = \bar{W}_i(0,t)$. Under GPS at time $t+\frac{L_{max}}{r}$, flow 2 receives almost another NL_{max} bits of service, whereas under MSFQ flow 2 does not get any service in $[t, t+\frac{L_{max}}{r}]$. Thus, $W_i(0, t+\frac{L_{max}}{r}) \approx \bar{W}_i(0,\frac{L_{max}}{r}) + NL_{max}$. □

We now prove that this is indeed the maximum amount at which the service a flow receives under GPS exceeds the one under MSFQ .

Theorem 3: For any τ,

$$W_i(0,\tau) - \bar{W}_i(0,\tau) \leq NL_{max}.$$

Proof: The difference $W_i(0,\tau) - \bar{W}_i(0,\tau)$ reaches its maximal value at a point when the slope of W_i decreases or the slope of $\bar{W}_i$ increases. Thus, for a flow i, the maximum difference can occur at one of the following events: When flow i becomes idle in the GPS system, when a packet of flow i begins transmission in the MSFQ system, or when an idle flow j, $j \neq i$ becomes backlogged in the GPS system while flow i is backlogged in GPS. The first case occurs when the slope of W_i becomes zero after GPS completes the transmission of the last packet of a flow i's backlog. Since this case occurs at one of the points when a packet of flow i departs the GPS system, we consider this more general scenario for the first case. Let b_k and $\bar{b}_k$ denote the time when packet k is scheduled under GPS and MSFQ respectively.

if : $\bar{b}_k \leq d_k$

$$W_i(0,d_k) - \bar{W}_i(0,\bar{b}_k) \leq (N-1)L_{max} + L_k. \qquad (7)$$

Case 1: Packet p_k of flow i departs GPS: $\bar{W}_i(0,\bar{b}_k) \leq \bar{W}_i(0,d_k)$ holds since $\bar{b}_k \leq d_k$. From Equation 7, it follows that

$$W_i(0,d_k) - \bar{W}_i(0,d_k) \leq NL_{max}.$$

Case 2: packet p_k of flow i is scheduled in MSFQ : Since $\bar{b}_k \leq d_k$ holds, $W_i(0,\bar{b}_k) \leq W_i(0,d_k)$. From Equation 7, it follows that

$$W_i(0,\bar{b}_k) - \bar{W}_i(0,\bar{b}_k) \leq NL_{max}.$$

otherwise: $\bar{b}_k > d_k$

We will prove both Case 1 and 2 in showing that for any τ in $[d_k,\bar{b}_k]$,

$$W_i(0,\tau) - \bar{W}_i(0,\tau) \leq NL_{max}.$$

Let p_k be the k^{th} packet that is scheduled under MSFQ There is an interval $[t,\bar{b}_k]$, $t < d_k$, in which all MSFQ servers are continuously busy. This is because if a server were idle in $[d_k,\bar{b}_k]$, MSFQ would have scheduled p_k. Let g be the smallest integer, such that all MSFQ servers are busy between $\bar{b}_g$ and $\bar{b}_k$. Let m be the largest integer greater than or equal to g that satisfies both $g \leq m \leq k-1$ and $d_m > d_k$. Thus, for $m < i < k$,

$$d_m > d_k \geq d_i$$

Then, packet p_m is scheduled before packets $p_{m+1}, \cdots, p_k$ under MSFQ but p_m departs after all these packets under GPS.

If no such integer m exists, this is equivalent to the Case 2.1 in the proof of Theorem 4. Since $\bar{b}_k > d_k$, $a_g < d_k$ must hold. In $[a_g,\tau]$, MSFQ might be servicing at most $N-1$ maximum length packets besides packets $p_g, p_{g+1}, \cdots, p_k$. In the worst case, $W_i(a_g,\tau) = (\tau - a_g)Nr$. If $\tau - a_g \geq \frac{L_{max}}{r}$, then

$$W_i(0,\tau) - \bar{W}_i(0,\tau) \leq (N-1)L_{max}.$$

If $\tau - a_g < \frac{L_{max}}{r}$, in the worst case, at time τ, $\bar{W}_i(0,\tau)$ is missing $(N-1)(L_{max} - (\tau - a_g)r)$ service from the packets before packet p_g and $(\tau - a_g)(N-1)r$ service from the packet packets $p_g, p_{g+1}, \cdots, p_k$. Thus,

$$W_i(0,\tau) - \bar{W}_i(0,\tau) \leq (N-1)L_{max}.$$

If such an integer m exists, this is equivalent to the Case 2.2 in the proof of Theorem 4. $a_m < d_k$ holds since $\bar{b}_k > d_k$ and as a result any packet MSFQ schedules in $[d_k,\bar{b}_k]$ before scheduling p_k must have been completed before p_k under GPS. In $[a_g,\tau]$, MSFQ might be servicing at most $N-1$ maximum length packets and p_m besides packets $p_{m+1}, \cdots, p_{k-1}$ (note that p_m is scheduled after p_k under GPS). Thus, same structure from the previous paragraph can be used to show

$$W_i(0,\tau) - \bar{W}_i(0,\tau) \leq NL_{max}.$$

Case 3: an idle flow j, $j \neq i$, becomes backlogged at t while flow i is backlogged in GPS: Let p_k be the k^{th} packet of flow i that completes under GPS. Since the potential slope decreases of W_i occurs between two consecutive flow i packet departures from GPS, t is in $[d_k, d_{k+1}]$ for some k. The proof follows since the interval $[d_k, d_{k+1}]$ is already covered by the cases proven above. □

6. FAIRNESS

So far, we've shown that a $(MSFQ\ , N, r)$ system closely approximates a $(GPS, 1, Nr)$ system in terms of the delay a packet can experience (Section 4) and the cumulative service a flow receives (Section 5). Another desirable property is to ensure that the amount of service a flow receives in the packetized system does not exceed arbitrarily the amount it would have received under GPS. This property leads to smoother output and "better" fairness [5]. Few metrics that quantify fairness are used in the GPS literature [9, 5, 10]. In this paper, we measure the fairness of a packetized discipline by the maximum difference of the amount of service any flow receives within any interval to the one the flow would have received under GPS. If the maximum difference is independent of the set of flows, we say that the packetized discipline provides bounded fairness. MSFQ does not enjoy this property since there is no constant c for which $\bar{W}_i(t,\tau) \geq W_i(t,\tau) - c$ holds for every interval $[t,\tau]$ [15]. Thus MSFQ can largely diverse from the ideal discipline by being far ahead in the completed work for a flow.

Service disciplines with bounded fairness are especially desirable for rate adaptive applications and for congestion control algorithms. Being able to schedule packets much

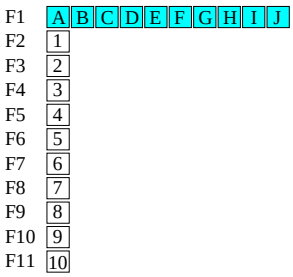

Figure 4: Queued packets at time t=0.

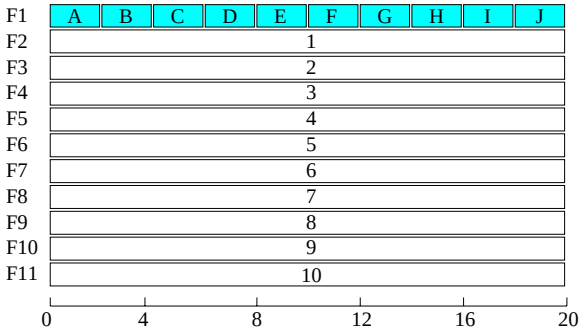

Figure 5: GPS packet scheduling.

earlier than the reference system, can cause the discipline to favor some flows and behave in a bursty way over given periods of time. This problem is addressed by Bennet et al. in [5] for the single server packetized system $(MSFQ\ , 1, r)$. Unfortunately, the clever solution presented by Bennet et al. does not apply directly to the multi-server case.

Example 3: Consider the case of 11 flows sharing 4 output servers. The first of the flows (F1) has a weight of 0.5 while each other flow has a weight of 0.05. At time 0, all packets have already arrived at the system, flow 1 has 10 packets while the other flows have only one each (Figure 4). For simplicity all packets have the same length of L. Figure 5 depicts the scheduling in the ideal system. Since MSFQ schedules packets in increasing order of GPS departure times, all of flow 1 packets will be scheduled before any other flow's. Figure 6 depicts this scenario where it can be seen that some of flow 1 packets are scheduled much earlier than the corresponding GPS discipline. For example, packet J is completed at time 12, that is 8 units earlier than in the ideal system. It can be shown that this "earliness" can be arbitrarily large and depends on the number of existing flows in the system. □

A solution to this problem for single WFQ server was proposed by Bennet et al. in [5]. Their method, called WF^{2}Q, consisted in restricting the packets eligible for scheduling to only the ones that have already started service in the GPS system. The scheduling of these packets was still done according to the WFQ discipline, that is in non-decreasing order of GPS finishing times. Conceptually, their method inserted a packet regulator at the exit of the flow queues which delayed the eligibility of the packets to the WFQ scheduler. Unfortunately, the direct application of this technique to multi-server systems does not fix the undesired burstiness problem and moreover, it makes the discipline non-work-conserving.

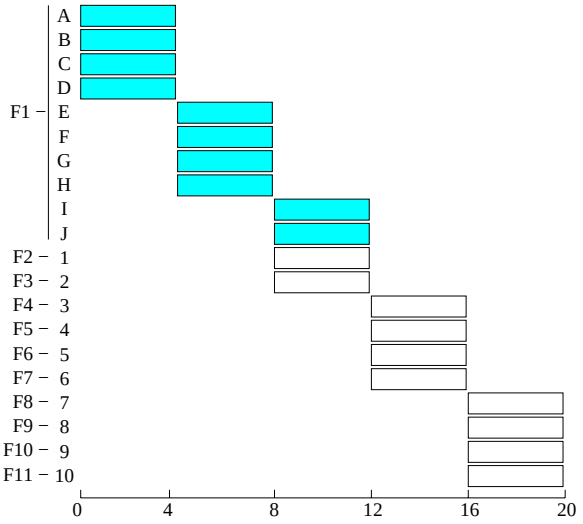

Figure 6: MSFQ scheduling without eligibility times.

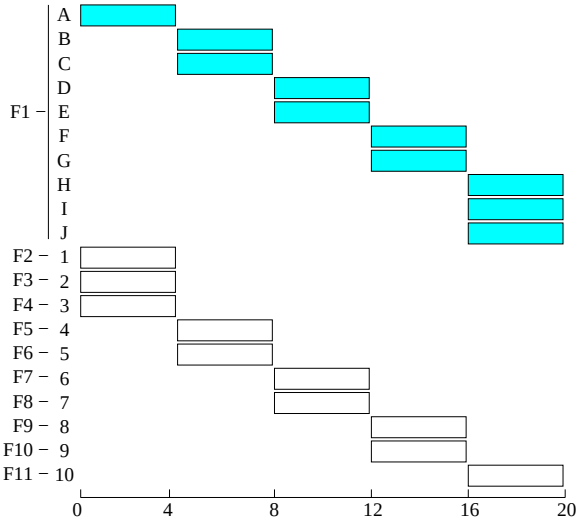

Figure 7: MSFQ scheduling using eligibility times.

An instance of the first problem is illustrated by Figure 7, which shows the scheduling output of Example 3 using a multi-server system with the WF^{2}Q discipline. It can be seen that packets from the first flow can still experience transmission periods that are as bursty as the previous case of Figure 6. Thus, the application of WF^{2}Q to multi-server case still does not lead to smooth schedules.

To illustrate that this regulator technique results into a non-work-conserving scheduling discipline, take the case where a large number of maximum length packets from a single flow are queued in the system at time t. In the GPS case, they will be scheduled sequentially at full rate of the server (Nr), irrespective of the weights of the flows. In this scenario, as Figure 8 shows, the second packet will not be eligible in the packetized system until the same packet gets scheduled in GPS, that is at $t + \frac{L_{max}}{Nr}$. Therefore, no matter how many servers there are available until that moment, they will remain idle even there's work to be done in the system. This

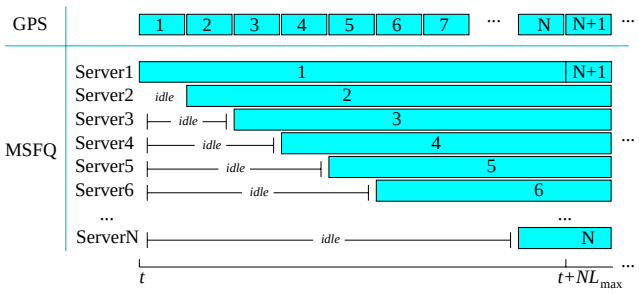

Figure 8: Non-conserving MSFQ scheduling example.

situation will continue to repeat until most of the first packet has been transmitted ($t + \frac{(N-1)L_{max}}{N}$) on one of the servers.

The WF^{2}Q regulator technique can be modified to become work-conserving. A simple extension would be if non-eligible packets were allowed to be scheduled to an idle server in cases where no other eligible packets were queued in the system. However, this modified version of WF^{2}Q does not enjoy the simple extension of the bound on $\bar{W}_i(0,\tau) - W_i(0,\tau)$ from $L_{i,max}$ in single server case (as proven in [5]) to $NL_{i,max}$ in multi-server case.

Consider an example with 2 flows sharing 10 output servers. The first flow has a weight 0.9 while the second one has a weight 0.1. $L_{2,max}$ is 1. All the packets of flow 2 arrive at time 0 and each has a length of $L_{2,max}$. The first packet of flow 1 arrives at time 0 and has a length 100. Flow 1 arrival rate is $0.9Nr$. Thus, the second packet of flow 2 arrives at time 100/0.9. At time 0, the first packets of flow 1 and 2 are eligible and they are scheduled. Since there are 8 idle servers and no eligible packets, to keep the system work-conserving, the non-eligible packets in the system are scheduled in the order of their GPS finishing times. Until the second packet of flow 1 arrives, 99 packets of flow 2 are scheduled. At this time, $\bar{W}_2(0, 100/0.9) - W_2(0, 100/0.9)$ is approximately 88.8, not $NL_{2,max} = 10$.

6.1 MSF^{2}Q

Our goal is to devise a packetized service discipline for multi-server systems that provides bounded fairness and generates "smooth" schedules. To this end, we introduce a new discipline, which we refer to as a (MSF^2Q , N, r) system or simply MSF^{2}Q .

We say that a packet is outstanding, if it is being transmitted or picked for transmission by the packetized system. Let $\hat{o}_i(t)$ denote the number of outstanding flow i packets at the MSF^{2}Q system at time t. We denote the work completed for flow i under MSF^{2}Q over the interval $[\tau, t]$ by $\hat{W}_i(\tau, t)$. At time t, when a server is idle and there is a packet waiting for service, MSF^{2}Q schedules among the flows that satisfy

$$\hat{W}_i(0,t) < W_i(0,t) \text{ or } (\hat{W}_i(0,t) = W_i(0,t) \text{ and } \hat{o}_i < \lceil \frac{r_i(t)}{r} \rceil)$$

the packet that would complete service in the GPS system earliest.

MSF^{2}Q reduces to WF^{2}Q if the number of servers is one. Figure 9 depicts the output of MSF^{2}Q in the previous scenario of Example 3. It can be seen that the resulting service is the closest achievable to the ideal discipline.

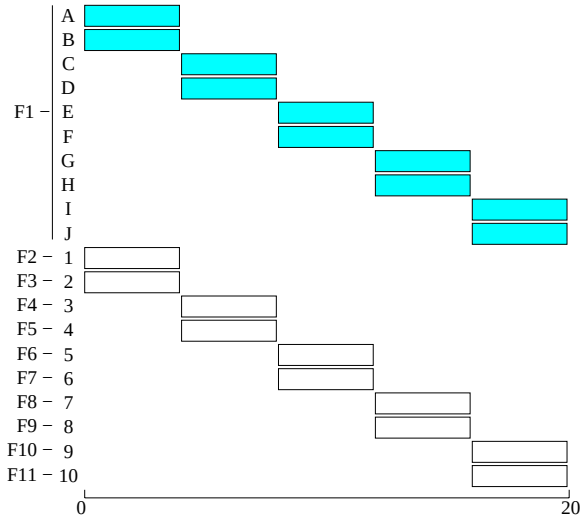

Figure 9: MSF^{2}Q scheduling.

6.2 Properties of MSF^{2}Q

The following theorem proves the bound for the extra amount of service a flow can receive at any time τ under MSF^{2}Q compared to GPS.

Theorem 4: Let $L_{i,max}$ denote the maximum packet length of flow i. For any time τ and flow i, the following property holds:

$$\hat{W}_i(0,\tau) - W_i(0,\tau) \leq NL_{i,max} \tag{8}$$

Proof: When there are no packets of flow i in transmission under MSF^{2}Q , the difference $\hat{W}_i(0,\tau) - W_i(0,\tau)$ is non-increasing. Thus, it is sufficient to show that for any i, during the transmission (including completion) of a flow i packet under MSF^{2}Q , $\hat{W}_i(0,\tau) - W_i(0,\tau) \leq NL_{i,max}$. At any time t when MSF^{2}Q picks the next flow i packet p_n for transmission the condition $((\hat{W}_i(0,t) < W_i(0,t))$ or $((\hat{W}_i(0,t) = W_i(0,t))$ and $\hat{o}_i < \lceil \frac{r_i(t)}{r} \rceil)$ holds for flow i. Since this condition holds for flow i and at most $N-1$ other flow i packets may be in transmission at t, ($\hat{W}_i$ may exceed W_i at most $NL_{i,max}$. □

The bound on $W_i(t,\tau) - \hat{W}_i(t,\tau)$ follows from combining the above theorem with the bound on cumulative per-flow service. We do not present in this paper the bounds on the packet delay and cumulative per-flow service for MSF^{2}Q , which can be calculated similar to the ones of MSFQ . It is worthwhile to mention that MSF^{2}Q is not work-conserving either. It is part of our future work to investigate the implications of work-conserving schedulers for multi-server systems on fairness bounds.

7. APPLICATIONS

There are numerous existing system architectures that follow very closely the multi-server model described in this paper. These systems can benefit from multi-server fair queuing disciplines to provide QoS guarantees on the access of their resources. We briefly discuss some of these application areas and describe how the multi-server model can be applied.

Link Aggregation is probably the most apparent example in the networking area. Ethernet link aggregation is a technique that allows the logical grouping of several network interfaces to allow for better scalability and fault-tolerance. The use of such technique is becoming increasingly popular since it provides a cost-effective and fault tolerant solution for incrementally scaling the network I/O capacity of the current high-end switches and servers. Many IEEE 802.3ad [13] standard and vendor-specific implementations are currently available [16, 14, 1, 2, 12]. The number of aggregated links on the existing systems varies largely among vendors and currently ranges from two to eight Fast/Gigabit Ethernet ports in either servers or switching elements. Although all of the available implementations utilize load balancing techniques such as round robin or static parameter hashing, none of these systems provide QoS guarantees over aggregated links.

Algorithms such as MSF^{2}Q can also be implemented to provide QoS guarantees in the access of storage I/O. For midrange and high-end storage systems, it is common to connect the RAID system to a host (e.g., Web server) with multiple SCSI or FC channels to improve the I/O performance. A number of storage vendors (e.g., EMC) are offering multi-path I/O software for load balancing and failover among the channels. Furthermore, the need for fairness and service guarantees for storage I/O is growing with the consolidation of clients' data and applications in the service providers' data centers. Since storage I/O traffic can be modeled as variable size packets, MSF^{2}Q type algorithms can be used to provide fair sharing of multiple I/O channels.

When distributing traffic across multiple links, as in the previous examples, the order in which the packets are received at the destination may be different from the order in which they were originally sent. Potential out-of-order delivery does not affect all applications. However, it may lower the expected end-to-end performance, for example, of TCP connections, since out-of-order reception of TCP packets may cause unnecessary retransmissions. Since current systems contain only few links but handle large number of flows, out-of-order-delivery due to multiple paths is not expected to be common. It is also important to note, that rather than being an artifact of our Fair Queuing algorithm, this misordering is an inherent problem of balancing load among multiple outgoing links and its impact should be studied.

8. RELATED WORK

Our work builds on a number of previous studies of GPS-based scheduling disciplines for single server systems. The concepts of Proportional Sharing and Generalized Processor Sharing are presented in [8], [11] and in [15]. Demers et al. introduced a packetized service discipline—WFQ in [8]. Later on, Parekh and Gallager proved in [15] that WFQ (a.k.a., PGPS) closely approximates the ideal GPS system in terms of packet delay and the cumulative per-flow service. Their work also showed end-to-end packet delay properties of a GPS system when flows are leaky bucket constrained. In [5], Bennet and Zhang observed that the service provided to a flow under WFQ may unboundedly exceed the amount of service received under GPS and indicated that this could lead to unsatisfactory fairness and have adverse effects on the behavior of adaptive flows. The same authors introduced the WF^{2}Q service discipline which incorporated the concept of eligibility times to WFQ and also showed that WF^{2}Q bounds the worst-case fairness. We addressed the same problem with a new algorithm called MSF^{2}Q designed for multi-server systems, since using WF^{2}Q for multi-server systems does not alleviate the burstiness problem. MSF^{2}Q is designed such that it reduces to WF^{2}Q when the number of servers is exactly one.

For single server systems, there have been several proposals [17, 9, 10] to approximate GPS with a lower computational complexity than WFQ. SFQ[10] showed that using start times was possible to get fairness guarantees. A later work from Bennet and Zhang [4] introduced the concept of hierarchical GPS and they presented a packetized algorithm for this model.

Weighted fair queuing for multi-server systems has not been studied in the packet scheduling literature (to best of our knowledge). Although there have been extensive work in the context of multiprocessor scheduling, only [7] considers a sharing model similar in spirit to GPS. The authors presented a scheduling algorithm called SFS for "fairly" sharing multiprocessors among threads. They introduced a model called GMS different than GPS as the reference system in order to capture the constraint that a thread can execute at one processor at a time. They experimentally showed that under certain workloads, their multiprocessor scheduling algorithms displays similar properties to their new reference system. This algorithm does not provide guarantees and therefore we did not consider it for multi-link packet scheduling.

Balancing packets across multiple interfaces is another topic that has been the focus of many studies. In [3], Adiseshu et al. designed a round-based iterative algorithm to distribute packets across multiple outgoing links. Although they used Fair Queuing concepts in their design, these concepts were not applied towards the provisioning of QoS guarantees. Their aim was to evenly balance the load of a single flow across N outgoing links rather than provide sharing guarantees of link utilization for several input streams. Moreover, we cannot compare the algorithm to MSFQ with even a single input flow since our approach does not require the input to be backlogged and its bounds are independent from the workload.

A number of implementations of link aggregation are currently available. They range from proprietary ones such as Adaptec's Duralink[2], SUN's SunTrunking[16], 3Com's Dynamic Access[1] or later ones following the new standard such as HP's Auto Port Aggregation[12]. However, while all these technologies are able to share multiple links from a stream of packets, none of them provide proportional sharing among multiple streams.

9. CONTRIBUTIONS AND FUTURE WORK

Link aggregation, or the aggregation of multiple interfaces into a single logical link, is becoming the predominant approach for bandwidth scaling. The existing link aggregation schemes aim for even distribution of traffic among the aggregated interfaces. However, fairness and service guarantees for flows sharing these aggregated links have not been addressed.

Although GPS-based fair queuing service disciplines are extensively studied for proportionally sharing a single link, the resulting concepts have not been applied in the cases

where several links are aggregated. In this paper, we studied packetized GPS-based service disciplines for these multi-server situations. Ideally, the provision of such systems is described as a function of the total link capacity. Therefore, we proposed a reference system consisting of a single GPS server operating at a rate that equals to the sum of the rates of the underlying servers.

Numerous fair queuing results previously obtained for single server systems do not directly apply to multi-server systems. This is because the rate at which the packetized multi-server system operates may vary over time and differ from the rate of the reference system. Furthermore, the packetized multi-server system may reorder the packets to remain work-conserving. We first analyzed the cumulative service, packet delay and per-flow cumulative service bounds for weighted fair queuing (WFQ) applied to a multi-server system. We then presented a new fair queuing algorithm—MSF^{2}Q that also bounds the additional amount of service a flow may receive under the packetized discipline compared to GPS, a property that is not shared by WFQ. As a result, MSF^{2}Q leads to smooth and fair schedules in finer time scales.

Our future plans include investigation of implementation issues, quantitative comparison of the approach presented in this paper to the alternative approach of partitioning flows among servers, and enhancing the algorithms for multiprocessors and cluster of servers. Other interesting extensions involve hierarchal GPS and servers with different rates.

10. REFERENCES

[1] 3Com's Dynamic Access. http://www.3com.com .

[2] Adaptec Duralink Software Suite. http://www.adaptec.com .

[3] H. Adiseshu, G. M. Parulkar, and G. Varghese. A Reliable and Scalable Striping Protocol. In *Proceedings of the ACM SIGCOMM*, August 1996.

[4] J. C. R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. In *Proceedings of the ACM SIGCOMM*, August 1996.

[5] J. C. R. Bennett and H. Zhang. WF^{2}Q: Worst-case Fair Weighted Fair Queueing. In *Proceedings of the IEEE INFOCOM*, San Francisco, March 1996.

[6] J. Blanquer, J. Bruno, E. Gabber, M. Mcshea, B. Özden, and A. Silberschatz. Resource Management for QoS in Eclipse/BSD. In *Proceedings of the First FreeBSD Conference*, Berkeley, California, Oct. 1999.

[7] A. Chandra, M. Adler, P. Goyal, and P. Shenoy. Surplus Fair Scheduling: A Proportional-Share CPU Scheduling Algorithm for Symmetric Multiprocessors. In *Proceedings of the USENIX 4th Symposium on Operating System Design and Implementation*, San Diego, California, Oct. 2000.

[8] A. Demers, S. Keshav, and S. Shenker. Design and Analysis of a Fair Queuing Algorithm. In *Proceedings of the ACM SIGCOMM*, Austin, Texas, September 1989.

[9] J. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *Proceedings of the IEEE INFOCOM*, Toronto, June 1994.

[10] P. Goyal, H. Vin, and H. Chen. Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. In *Proceedings of the ACM SIGCOMM*, August 1996.

[11] A. Greenberg and N. Madras. How Fair is Fair Queuing. *Journal of the ACM*, July 1992.

[12] Hewlett Packard's Auto-Port Aggregation. http://www.unix.hp.com .

[13] Amendment to Carrier Sense Multiple Access With Collision detection (CSMA/CD) Access Method and Physical Layer Specifications - Aggregation of Multiple Link Segments. IEEE 802.3ad Standard, 2000.

[14] Intel's Adaptive Load Balancing. http://www.intel.com.

[15] 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*, pages 344–357, June 1993.

[16] SUN Trunking Software. http://www.sun.com .

[17] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. In *Proceedings of the ACM SIGCOMM*, Philadelphia, 1990.