Hierarchical Fair Queueing: A Credit-based Approach for Hierarchical Link Sharing

Andrew Do-Sung Jun, Jinwoo Choe, and Alberto Leon-Garcia

Abstract: **In this paper, we propose a hierarchical packet scheduling technique to closely approximate a hierarchical extension of the generalized processor sharing model, *Hierarchical Generalized Processor Sharing* (H-GPS). Our approach is to undertake the tasks of service guarantee and hierarchical link sharing in an independent manner so that each task best serves its own objective. The H-GPS model is decomposed into two separate service components: the *guaranteed service* component to consistently provide performance guarantees over the entire system, and the *excess service* component to fairly distribute spare bandwidth according to the hierarchical scheduling rule. For tight and harmonized integration of the two service components into a single packet scheduling algorithm, we introduce two novel concepts of *distributed virtual time* and *service credit*, and develop a packet version of H-GPS called *Hierarchical Fair Queueing* (HFQ). We demonstrate the layer-independent performance of the HFQ algorithm through simulation results.**

Index Terms: **Hierarchical packet scheduling, fair queueing, GPS, H-GPS, H-PFQ, HFQ**

I. INTRODUCTION

In support of real-time applications such as packet video, link capacities in a data communications network need to be shared according to the *Quality of Service* (QoS) requirements of communication sessions. Depending upon service models supported, such as integrated services [1] or differentiated services [2], link sharing ought to be performed on the per-session or per-group basis. Should the network be shared by multiple administrative domains, requiring QoS guarantees for themselves as well as for the groups and/or sessions within the domains, then it is inevitable that link sharing structure of the network should be configured into multiple layers [3], [4]. Fig. 1 depicts an example of a hierarchical scheduling structure in a packet switch or a router, where a leaf node represents a communication session while a non-leaf node represents a group of such sessions. Node 11, for example, consists of nodes 9 and 10, and node 9, in turn, represents the group of nodes 1, 2, and 3.

In this paper, we develop the *Hierarchical Fair Queueing* (HFQ) algorithm for fair hierarchical bandwidth distribution with tight performance guarantees. HFQ is an approximation

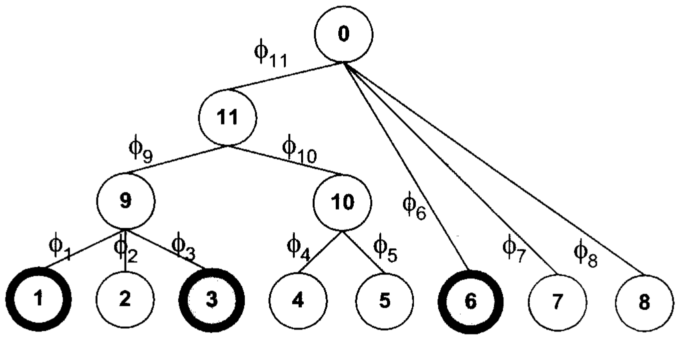


Fig. 1. A scheduling hierarchy example.

of the *Hierarchical Generalized Processor Sharing* (H-GPS) model defined in [5]. In H-GPS, service fairness is pursued in accordance with the *hierarchical link sharing rule*; that is, whenever bandwidth is spared by an inactive session, more "closely-related" sessions in the scheduling hierarchy will be given higher priority in claiming the spare bandwidth. Within the same degree of "kinship," the spare bandwidth is fairly shared in proportion to their service weights. If closest sessions cannot consume the spare bandwidth due to insufficient in-flows, then chances will be given to the next "closest" sessions. This process continues until either the spare bandwidth is fully consumed, or no session is backlogged in the entire system.

For an example of such H-GPS operation, consider the hierarchical scheduling structure in Fig. 1. If session 4 in Fig. 1 is not backlogged, the spare bandwidth has to be distributed only to session 5 as long as session 5 stays backlogged. In case that session 5 cannot provide enough traffic to consume all the spare bandwidth, session 5 (and, as a result, node 10) will become empty at some point of time, and then the combined spare bandwidth of sessions 4 and 5 will become available to sessions 1, 2, and 3 via nodes 11 and 9. Henceforth, we assume that packets arrive instantaneously to the system in batches to avoid the situation where the spare bandwidth is consumed only partially. Under this assumption, a session or node may require either as much bandwidth as it can claim (i.e., backlogged) or no bandwidth at all (i.e., empty).

Being aware of the performance discrepancy between the ideal GPS model and a GPS-based packet scheduling algorithm (such as WFQ [6], [7]), we expect that simple stacking of such single-level systems may result in error accumulation along the scheduling layers. Indeed, as will be illustrated through an example in the following section, "layer-dependent" performance characteristics are revealed in *Hierarchical Packet Fair Queueing* (H-PFQ) [5]. This error accumulation problem in H-PFQ stems from the fact that the "original" QoS requirement information of a session gets diluted with those of other sessions as

Manuscript received March 28, 2002; approved for publication by Bo Li, Division III Editor, August 28, 2002.

A. D.-S. Jun and A. Leon-Garcia are with the Edward S. Rogers, Sr. Department of Electrical and Computer Engineering at University of Toronto, Toronto, CANADA, e-mail: {ajun,alg}@comm.utoronto.ca.

J. Choe is with the Department of Electronic Engineering at Sogang University, Seoul, KOREA, e-mail: xinu@sogang.ac.kr.

This work was partially supported by Sogang University Research Grant #20011511.

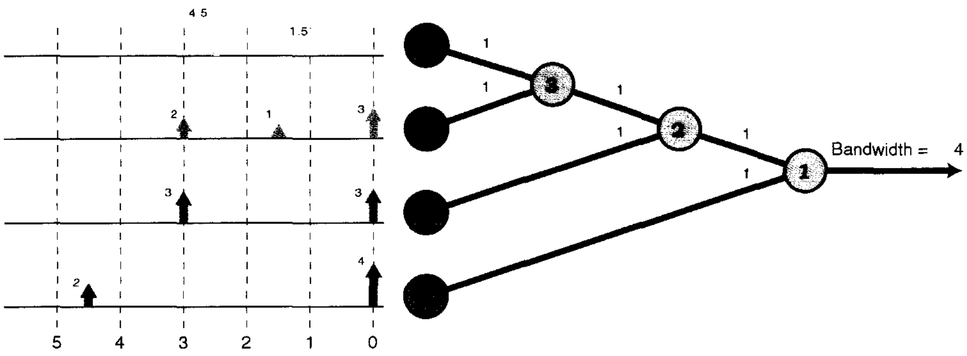
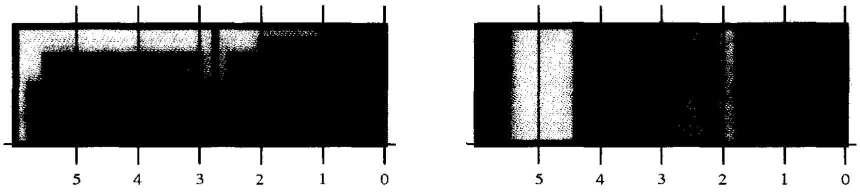


Fig. 2. An example scenario of hierarchical scheduling.



(a) Service progress in the ideal HPS model (b) Direct approximation of the HPS model

Fig. 3. Comparison of H-GPS model and its approximation.

the packet proceeds through layers of single-level servers. The information attenuation occurs because each single-level server in H-PFQ schedules packets based only on the "local" QoS requirement information, which represents QoS requirements of sessions in groups, but not on the individual basis.

In HFQ, we avoid the error accumulation problem by retaining the "original" QoS requirement information of sessions throughout the scheduling hierarchy. In this way, all packets are treated in a "globally-fair" manner to receive consistent performance guarantees over the entire system. In order to maintain the hierarchical bandwidth distribution capability in the system, we introduce novel mechanisms to manage spare bandwidth in an explicit manner. We exploit the concepts of distributed virtual time and service credit to accomplish two potentially conflicting goals of hierarchical bandwidth distribution and layer-independent delay performance in harmony.

II. HIERARCHICAL LINK SHARING METHODOLOGY

Before we introduce HFQ, we illustrate its design principle through an example in this section. By comparing the design principle with that of H-PFQ, we demonstrate how the task of hierarchical link sharing can be better modeled by a single-step modeling approach than by multi-step modeling approaches like H-PFQ.

A. Single-step Approach

Consider an ideal hierarchical link sharing system with 7 nodes given in Fig. 2. If the hierarchical link sharing rule, explained in the previous section, is captured for its entirety in a single step, then the hierarchical system will become a H-GPS system, and the bandwidth distribution among 4 sessions will take place as in Fig. 3 (a). This fluid model (i.e., H-GPS) of the hierarchical link sharing rule does not exhibit any performance degradation as the number of hierarchical levels increase. This is because H-GPS retains the information of fair bandwidth shares of individual sessions regardless of the locations of the sessions in the hierarchy of the link sharing structure.

If one tries to approximate the ideal H-GPS system just in the same way as WFQ approximates the GPS model, the packets will be transmitted in the order of Fig. 3 (b) according to their finish times. This direct approximation of H-GPS is not likely to exhibit serious or steep performance degradation due to the hierarchical link sharing structure. This is because the discrepancy between H-GPS and the approximation will always be caused only by the unavoidable restriction of a packet fair queueing system; i.e., "*at most one packet can be transmitted simultaneously while servers are not expected to idle whenever the system is backlogged.*" In other words, regardless of the number of hierarchical levels, the relationship of H-GPS and a direct approximation of H-GPS will be the same as that of GPS and a direct approximation of GPS, such as WFQ.

However, this direct approximation is achievable only if the information on the future packet arrival is available in advance. This is because unlike the case of GPS, the relative order of finish times of packets in a H-GPS system may be reversed by packets arriving in the future. Therefore, in developing a packet version of H-GPS, there exists another practical and fundamental restriction; i.e., "*a packet must be chosen for transmission based only on the history of the past packet arrival.*" In fact, the design principle of HFQ is to approximate H-GPS as closely as the direct approximation illustrated in Fig. 3 (b) without any information on the future packet arrival.

B. Multi-step Approach

Consider a H-PFQ system that approximates the task of hierarchical link sharing in multiple steps. In H-PFQ, each non-leaf

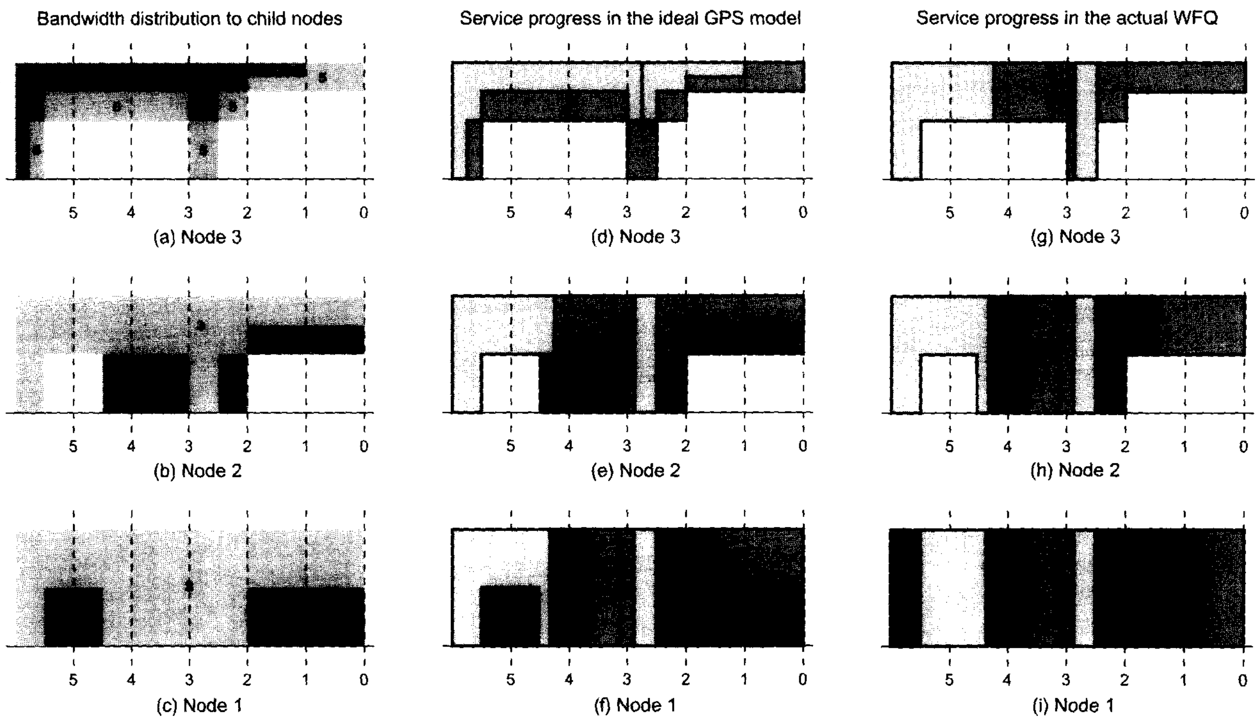


Fig. 4. Bandwidth distribution and service progress at nodes 1–3.

node (or local server) is operated based on a GPS-based *Packet Fair Queueing* (PFQ) model, such as WFQ, WF2Q, SCFQ, and SFQ (refer to [8] and references therein for various GPS-based packet scheduling algorithms) in an independent and isolated manner. Therefore, even non-leaf child nodes are treated by their parent nodes as if they were FIFO (First-In, First-Out) queues serving only a single packet stream, and only partial scheduling information (the relative service order of packets within the group) is transferred from one local server to the next. Although this loss of information (i.e., finish times) may not immediately damage the service fairness at the local server, it may cause fluctuations in packet delay by disturbing service fairness in the global scale.

This behavior of H-PFQ can be easily understood by examining the operations of nodes 3, 2, and 1 for the scenario given in Fig. 2. Consider the H-GPS system in Fig. 2, where non-leaf servers will be replaced by a WFQ one by one in the order of nodes 3, 2, and 1. Before any non-leaf node is replaced by a WFQ (i.e., in the original H-GPS system), the finish time of an incoming packet can be computed at each non-leaf node with the variation in the allocated bandwidth taken into account. One can easily track the bandwidth at each non-leaf node available for its child nodes as in Fig. 4 (a)–(c). If node 3 is a GPS server, the packets will be served (i.e., forwarded to node 2) as in Fig. 4 (d), where the finish times of packets arriving at node 3 can be computed. Now, if we replace node 3 with a WFQ, then packets will be forwarded to node 2 in the order of their finish times through the bandwidth available at node 3 as is illustrated in Fig. 4 (g). Once node 3 is replaced by a WFQ, packets from node 3 will appear to be a single stream of packets from the viewpoint of node 2, and hence, the service progress at node 2 will be given by Fig. 4 (e). Consequently, after node 2 is replaced by a WFQ, packets will be forwarded from node 2 to node 1 in the order of Fig. 4 (h).[1] Through a similar procedure, the service progress at node 1 can be determined as in Fig. 4 (f) and (i), before and after node 1 is replaced by a WFQ, respectively. Since node 1 is the root node, the order of packet service at node 1 corresponds to the packet service order of the entire system. In other words, Fig. 4 (i) represents the order of packet transmission from the H-PFQ system composed of layered WFQ servers. As expected, the order of packet transmission in the H-PFQ system is not the same as that of the single-step approximation given in Fig. 3 (b). The discrepancy in the packet transmission orders of the ideal H-GPS and the H-PFQ systems is caused not just by the inevitable restrictions of a packet fair queueing system, but also by the multi-step approximations of the hierarchical link sharing rule, implicitly carried out at every non-leaf node. For instance, if node 3 is a WFQ, node 2 will receive a series of packets from node 3 one by one as given in Fig. 4 (g), and node 2 will compute the finish times of these packets based only on the local service weight information (i.e., service weight of 1 for both child nodes of node 2) as illustrated in Fig. 4 (e). Therefore, once the order of packets being forwarded from node 3 to node 2 is determined, the service weights of individual child sessions of node 3 will be ignored at all parent nodes of node 3 including node 2. Similarly, the local service weight of node 3 at node 2 will also be disregarded once packets from sessions 4, 5, and 6 are forwarded to node 1. In other words, as a packet from a session travels through the hierarchy, its service weight information at lower levels of hierarchy gets lost, and the packet will be treated identically with packets from other sessions in the same group. The result is that H-PFQ reveals the problem of error accumulation, which degrades delay performance along

[1] At node 2, two packets are tied in their finish times, and one of them can be chosen randomly for transmission in such cases. In this specific example, the packet from session 4 was forwarded first to the next level.

the depth of hierarchy as indicated in [5].

In contrast, as will be illustrated in the following sections, HFQ as a single-step approximation of the ideal hierarchical link sharing rule bears no such problem. Consequently, HFQ can be considered as an improved fair hierarchical link sharing discipline over H-PFQ.

III. FLUID MODEL

The error accumulation problem or layer-dependent performance of H-PFQ is fundamentally induced by the multi-step approximation approach as illustrated in the previous section. This implies that the same problem will arise as long as the H-GPS model is viewed as a group of hierarchically "stacked" GPS servers and individual GPS servers are approximated by a packet version of GPS. Therefore, to take one step further for layer-independent performance, a different interpretation of the H-GPS model is required, in which the operation of the H-GPS model is described as a whole, but not as a set of GPS servers.

In this section, we introduce two fluid hierarchical link sharing models, which are identical to H-GPS (as defined in [5]) in function, but different in expression. We name the models as HPS and Explicit HPS (E-HPS), where the word "explicit" is used to emphasize that the management of guaranteed bandwidth (or service) and that of spare bandwidth are defined separately.

A. Hierarchical Processor Sharing (HPS)

A HPS server is characterized by a hierarchical scheduling structure, such as the one in Fig. 1, where each node i is characterized by a positive real number ϕ_i, the *local service weight* of the node. From now on, let us assume that $[t_1, t_2)$ represents an interval during which all sessions (i.e., leaf nodes) are either continuously backlogged or continuously empty. Also, let $p^h(k)$, $B_k(\tau)$, and A^L be the hth parent node of session k, the set of backlogged child nodes of node k at time τ, and the set of all sessions, respectively. Then, we define the *effective service weight* of node k at time τ as

$$\Gamma_k(\tau) = \prod_{h=1}^{H_k} \left(\frac{\phi_{p^{h-1}(k)}}{\sum_{b \in B_{p^h(k)}(\tau)} \phi_b} \right), \tag{1}$$

where H_k is the number of ancestor nodes of node (or session) k through (and including) the root node in the hierarchy. Now, the operation of the HPS model with service capacity R can be defined in terms of $W_i(t_1, t_2)$, the amount of service that session i receives during the interval $[t_1, t_2)$ as follows:

The HPS server is a work-conserving server such that

$$(t_2 - t_1)R = \sum_{i \in A^L} W_i(t_1, t_2) \tag{2}$$

if there is at least one continuously backlogged session during the interval $[t_1, t_2)$ *and*

$$\frac{W_j(t_1, t_2)}{\Gamma_j(t_1)} = \frac{W_i(t_1, t_2)}{\Gamma_i(t_1)} \tag{3}$$

holds for any continuously backlogged sessions i and j.

B. Explicit Hierarchical Processor Sharing (E-HPS)

The actual operation of the E-HPS model is identical to that of HPS or H-GPS, but in the E-HPS model, the service distribution rule is stated in terms of guaranteed service and excess service. Let $G_i(t_1, t_2)$ and $E_i(t_1, t_2)$ denote the amounts of guaranteed and excess service that session i receives during the interval $[t_1, t_2)$, respectively. Then, $W_i(t_1, t_2)$, the total amount of service that session i receives will be given as the sum of $G_i(t_1, t_2)$ and $E_i(t_1, t_2)$. As before, ϕ_k denotes the *local service weight* of node k, and R the service (or link) capacity. Also, let A_k and $B_k(\tau)$ be the set of child nodes of node k and the set of backlogged child nodes of node k at time τ, respectively. Now, we define the (global) *nominal service weight* of node k as

$$\varphi_k = \prod_{h=1}^{H_k} \left(\frac{\phi_{p^{h-1}(k)}}{\sum_{a \in A_{p^h(k)}} \phi_a} \right). \tag{4}$$

Then, $G_i(t_1, t_2)$, the amount of guaranteed service that session i receives from the E-HPS server during the interval $[t_1, t_2)$ is given by

$$G_i(t_1, t_2) = \begin{cases} r_i(t_2 - t_1) & \text{if session } i \text{ is backlogged,} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $r_i = R \times \varphi_i$ is the *nominal guaranteed service rate* of session i.

The amount of excess service that session i receives during the interval $[t_1, t_2)$, on the other hand, is defined as the sum of $E_{i,k}(t_1, t_2)$, the amount of excess service that session i receives at each ancestor node k: i.e.,

$$E_i(t_1, t_2) = \sum_{h=1}^{H_i} E_{i,p^h(i)}(t_1, t_2). \tag{6}$$

At non-leaf node k, excess service is provided in such a manner that

$$\frac{E_{j,k}(t_1, t_2)}{\Gamma_{j,k}(t_1)} = \frac{E_{i,k}(t_1, t_2)}{\Gamma_{i,k}(t_1)} \tag{7}$$

holds for any descendant session i and j of node k that are continuously backlogged, where

$$\Gamma_{i,k}(\tau) = \prod_{h=1}^{H_{i,k}} \left(\frac{\phi_{p^{h-1}(i)}}{\sum_{b \in B_{p^h(i)}(\tau)} \phi_b} \right) \tag{8}$$

is the *effective service weight* of session i at node k at time τ, and $H_{i,k}$ denotes the number of ancestor nodes of session i through (and including) node k in the scheduling hierarchy. At the same time, unless all child nodes of node k are empty, the total amount of excess service provided at node k is constrained by

$$\sum_{b \in B_k^L(t_1)} E_{b,k}(t_1, t_2) = (t_2 - t_1) \times \sum_{e \in A_k - B_k(t_1)} r_e, \tag{9}$$

where $B_k^L(\tau)$ is the set of backlogged descendant sessions (i.e., leaf nodes) of node k at time τ. Note that the right-hand side of equation (9) accounts only for the spare bandwidth of the empty child nodes of node k whose descendant nodes are all empty. Hence, equation (9) simply expresses the "law of service conservation" that the total amount of *service credit* (or excess service) produced by idling sessions is same as the total amount of service credit consumed by active sessions in the same group.

Although the definitions of the H-GPS model, the E-HPS model, and the HPS model may appear to be different, they are all equivalent in their operation (see [9] for the proof). However, it should be noted that the guaranteed service component and the excess service component are completely split in the definition of the E-HPS. As can be seen from equation (5), (given the nominal service weight) the guaranteed service component is not dependent to the service hierarchy nor to the state of other sessions, and hence, every session is explicitly guaranteed to receive its guaranteed service amount regardless of the system state and the relative location in the hierarchy. In contrast, the excess service component is governed by the rules (i.e., equations (6)–(9)) for the hierarchical service redistribution process. This separate and protected control for the guaranteed and excess service components provides the theoretical basis for the development of a hierarchically-fair packet scheduling system with layer-independent delay bound performance.

IV. PACKETIZED MODEL

In this section, we propose a hierarchical packet scheduling algorithm to approximate the E-HPS (or equivalently HPS) model described in the previous section.

A. Design Principles

The design goal is to achieve layer-independent delay performance without compromising the global fairness of service distribution. In the following, we introduce the concepts of local virtual time and virtual time offset, and then explain how layer-independent service guarantee and globally fair bandwidth distribution mechanisms can be built upon these concepts.

In a single-level fair queueing system such as WFQ, the concept of system virtual time [10] is used as a measure of the progress of system service. When a session becomes newly backlogged, the start time of the *Head-of-Line* (HOL) packet of the session is set to the system virtual time. In this way, both newly-backlogged and continuously-backlogged sessions are treated fairly from any instant and onward regardless of the service histories of the sessions.

For tight management of service fairness over the entire system, a similar mechanism is also required in a multi-level (i.e., hierarchical) fair queueing system. The difficulty in a hierarchical system, however, is that a single value cannot characterize the state of the system as a whole. This is because the rate of service progress at a local server may deviate from the rate of service progress at others nodes as per the hierarchical link sharing rule. In a hierarchical fair queueing system, a *local virtual time* (LVT) can be defined at each local scheduling server in the same manner as done for single-level packet fair queueing systems, and can be used to sustain local-scale service fairness. However, as we have seen in Section II, local-scale service fairness at every local server does not immediately yield global-scale service fairness.

What we need is a systematic method to merge LVTs at all non-leaf nodes into a *distributed virtual time system* that effectively represents the service progress in the system in its entirety. For this purpose, we introduce the concept of *virtual time offset* (VTO) representing the difference between the LVTs of a child-parent pair, and refer to a virtual time system comprising LVTs and VTOs at all non-leaf nodes as the *distributed virtual time system*. The significance of the distributed virtual time system is that it represents the system state consistently throughout the entire system. Having the precise system state information in hand, newly-backlogged session(s) can start receiving fair service together with continuously-backlogged session(s) regardless of their relative locations in the hierarchy.

For computationally efficient implementation of E-HPS and its packet version, we need to develop practical means to realize hierarchically-fair spare bandwidth distribution. This is because naive implementation of the E-HPS model may result in a computationally expensive system. In particular, direct computation of $E_i(t_1, t_2)$ based on (6), (7), and (9) would entail evaluation of $E_{i,k}(t_1, t_2)$ at each ancestor node k of each "recipient" session i. Moreover, for the computation of effective service weights (i.e., $\Gamma_{i,k}(\tau)$), the complete state information of every single node in the system would be required as stated in equation (8).

As opposed to this "centralized" and "active" approach to compute excess service for every backlogged session, the distributed virtual time system allows us to take a "distributed" and "passive" approach by managing *service credit*, the amount of service unused by idling sessions at individual non-leaf nodes. Being passive and distributed, the new approach eliminates the need for per-node computation and system-wide state information, which will significantly reduce the computational complexity of hierarchical bandwidth distribution.

The mechanism for service credit management can be outlined as follows. Instead of actively distributing spare bandwidth of an idling session to an appropriate set of backlogged sessions one by one, each non-leaf node simply accumulates service credit generated from idling child node(s). Distribution of service credit is then performed in a passive manner by subtracting the amount of service credit accumulated at each non-leaf node from the finish times of packets arriving from its child nodes, just before they are forwarded to its parent node. From the fact that LVTs render the service progress at non-leaf nodes in a HPS or E-HPS system, one may expect that the accumulated amount of service credit at non-leaf nodes should be related to VTOs in the distributed virtual time system. The relationships among LVTs, VTOs, and service credit will be discussed in the following section together with the formal definition of the distributed virtual time system.

B. Distributed Virtual Time System

The distributed virtual time system simply represents the aggregation of the LVT and the VTO defined for every non-leaf node. $V_k(\tau)$, the LVT of non-leaf node k, is the measure of ser-

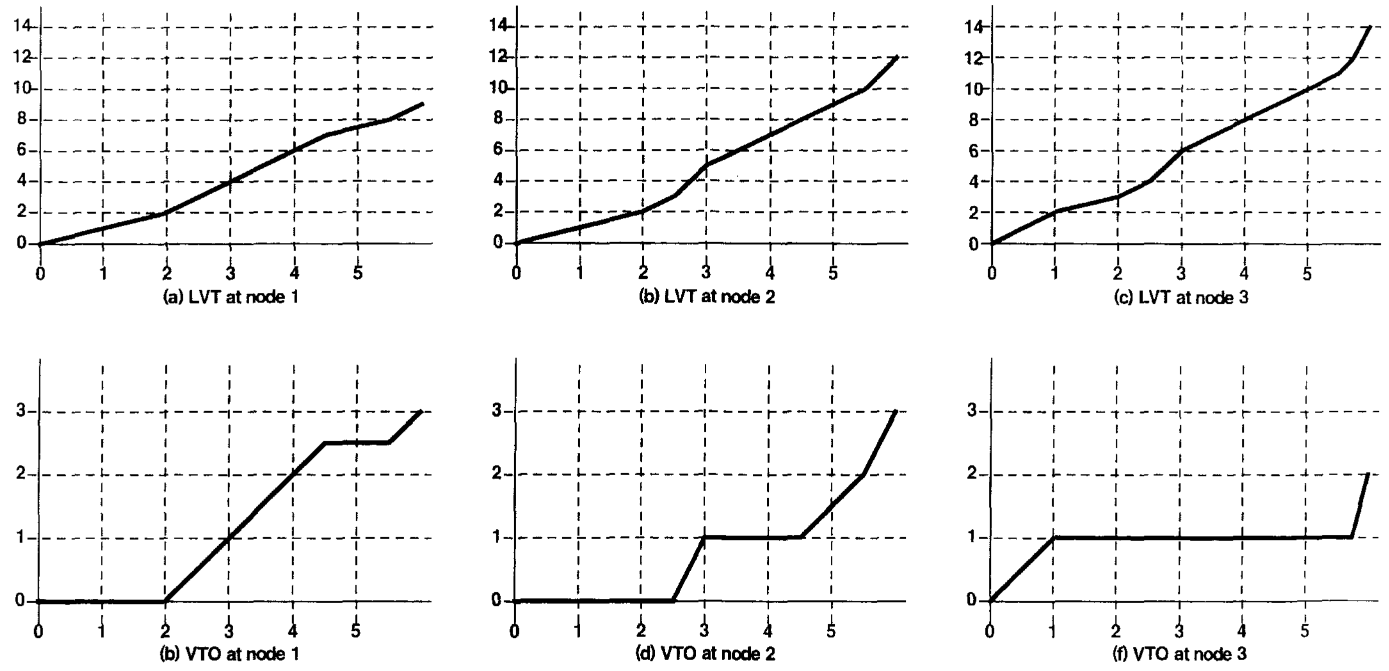


Fig. 5. Progress of the LVTs and the VTOs at nodes 1–3.

vice progress at node k, such that for any backlogged child node $b \in B_k(\tau)$ of node k, the bandwidth (i.e., service rate) that will be allocated to node b at time τ, is given by $R \times \varphi_b \times \frac{d}{d\tau}V_k(\tau)$. From this property of LVT, it follows that

$$\frac{dV_k(\tau)}{d\tau} = \frac{\sum_{a \in A_k} \varphi_a}{\sum_{b \in B_k(\tau)} \varphi_b} \times \frac{dV_{p(k)}(\tau)}{d\tau}, \tag{10}$$

or equivalently,

$$\frac{dV_k(\tau)}{d\tau} = \prod_{h=0}^{H_k} \frac{\sum_{a \in A_{p^h(k)}} \varphi_a}{\sum_{b \in B_{p^h(k)}(\tau)} \varphi_b}. \tag{11}$$

From (10), it is clear that the LVT of non-leaf node k and that of its parent node $p(k)$ increase at a same rate, only if all child nodes of node k are backlogged, and otherwise, the LVT of node k increases faster. The difference between the LVT of node k and that of its parent node is denoted by $\beta_k(\tau)$, the VTO at node k; i.e.,

$$\beta_k(\tau) = V_k(\tau) - V_{p(k)}(\tau). \tag{12}$$

By unrolling (12), the LVT at node k can be expressed only in terms of the VTOs and the LVT at the root node as follows:

$$V_k(\tau) = V_0(\tau) + \sum_{h=0}^{H_k - 1} \beta_{p^h(k)}(\tau). \tag{13}$$

Now, assume that the distributed virtual time system is synchronized by resetting LVTs and VTOs at all non-leaf nodes to 0 at the beginning of each system busy period. Then, LVTs at all non-leaf nodes can be tracked and remain synchronized during a busy period in many different ways. For the precise tracking of LVTs, one may integrate (11) starting from the beginning of a busy period, or the VTOs (instead of LVTs) may be tracked at every non-leaf node using the relationship

$$\begin{aligned}\frac{d\beta_k(\tau)}{d\tau} &= \left(\frac{\sum_{e \in A_k - B_k(\tau)} \varphi_e}{\sum_{b \in B_k(\tau)} \varphi_b}\right) \times \frac{dV_{p(k)}(\tau)}{d\tau} \\ &= \frac{\sum_{e \in A_k - B_k(\tau)} \varphi_e}{\sum_{a \in A_k} \varphi_a} \times \frac{dV_k(\tau)}{d\tau},\end{aligned} \tag{14}$$

which is an immediate result of (10) and (12). An example of such tight tracking is shown in Fig. 5 for the hierarchical link sharing scenario given in Fig. 2. Although these brute-force approaches would suffer from high computational complexity, we will henceforth assume that LVTs and VTOs are tracked exactly at all non-leaf nodes. This is for the sake of lucid illustration of finish-time computation and comparison mechanism in HFQ, which is far more critical than the VTO update mechanism in achieving layer-independent performance.

Now that LVT and VTO are formally defined, we illustrate how the service credit concept (introduced in Section IV-A) fits into the distributed virtual time system. In other words, it will be shown in the context of the distributed virtual time system that the total amount of service credit produced by idling child nodes is same as the amount consumed by backlogged child nodes, all seen at a non-leaf node k. Consider a node k, and assume that no child node of node k changes its state during a time period $[t_1, t_2)$. Also, let Φ_k^b be the sum of nominal service weights of backlogged child nodes of node k, and Φ_k^e be the sum of nominal service weights of empty child nodes. From the definition of the LVT, the total amount of service that node k will receive from its parent node (as long as there is at least one busy child

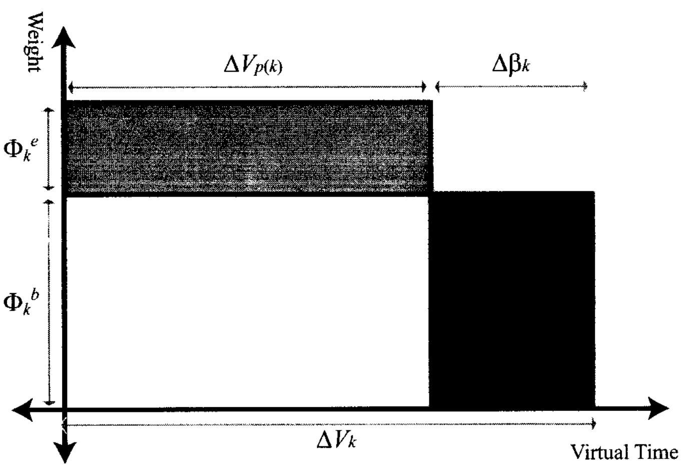


Fig. 6. Credit distribution via virtual time offset.

node) is given by $R\varphi_k \Delta V_{p(k)}(t_2, t_1)$, where $\Delta f(t_2, t_1)$ denotes the difference $f(t_2) - f(t_1)$. Besides, the total amount of service received by the busy child nodes B_k during the interval $[t_1, t_2)$ is $R\Phi_k^b \Delta V_k(t_2, t_1)$. Since the states of child nodes remain unchanged during the interval, it follows from (10) that $\Phi_k^b \Delta V_k(t_2, t_1) = \varphi_k \Delta V_{p(k)}(t_2, t_1)$.[2] Therefore, one can see that node k is work-conserving, and it will distribute the entire amount of service it can claim, to its busy child nodes.

If all child nodes of node k were busy, then V_k would increase at the same rate as $V_{p(k)}$ over the interval $[t_1, t_2)$, and thus, the amount of service that the set $B_k(t_1)$ of child nodes would receive, will diminish to $R\Phi_k^b \Delta V_{p(k)}(t_2, t_1)$. Hence, we can conclude that

$$R\Phi_k^b \Delta V_k(t_2, t_1) - R\Phi_k^b \Delta V_{p(k)}(t_2, t_1) = R\Phi_k^b \Delta\beta_k(t_2, t_1), \quad (15)$$

which corresponds to the extra amount of service that the busy child nodes (i.e., $B_k(t_1)$) receive because of the inactive child nodes $A_k - B_k(t_1)$. For this reason, $\beta_k(\tau)$ may be interpreted as the cumulated service credit (or extra service) per unit nominal weight that can be claimed by busy child nodes. The service credit produced (or spared) by idling child nodes is the amount of service that would be received by idling child nodes (i.e., $A_k - B_k(t_1)$) if they were busy; i.e.,

$$R(V_{p(k)}(t_2) - V_{p(k)}(t_1)) \sum_{b \in A_k - B_k(t_1)} \varphi_b = R\Phi_k^e \Delta V_{p(k)}(t_2, t_1). \quad (16)$$

Here, note from (14) that

$$\Delta\beta_k(t_2, t_1) = \frac{\Phi_k^e}{\Phi_k^b} \Delta V_{p(k)}(t_2, t_1). \quad (17)$$

From (15)–(17), one can easily see that in the context of the distributed virtual time system, the amount of service credit consumed by the busy child node is exactly the same as the amount of service credit produced by idling child nodes. This indicates that the equations (i.e., (10)–(14)) describing the dynamics of the distributed virtual time system fulfill the service conservation law stated in (2) or (9). This service conservation principle is illustrated graphically in Fig. 6, where the light-shaded and dark-shaded areas represent the amount of service credit produced by idling child nodes and claimed by busy child nodes, respectively. As was shown in (17), the sizes of these areas are always supposed to be identical.

[2]Remember that $\varphi_k = \sum_{i \in A_k} \varphi_i$.

C. HFQ Algorithm

In HFQ, the transmission order of packets is determined through a tournament competition (representing the service distribution hierarchy) of packets, where different scoring systems will be employed at different matches. Matches will take place at individual non-leaf nodes, and the scores of packets participating in a match will be calibrated with respect to the LVT at the corresponding non-leaf node, which represents the scoring system for the match. More precisely, to determine a winner packet at a non-leaf node (i.e., a packet that will be forwarded to the next hierarchical level), scores of HOL packets of backlogged child nodes will be compared, and the packet with the highest score (i.e., the smallest finish time) will become the winner packet. The winner packet will then be forwarded to the next level of the hierarchy, and a winner packet at the parent node will be determined in the same manner but under a different scoring system. This tournament competition will continue until a winner packet is determined at the root node, which will also become the packet chosen for transmission. The detailed time-stamp calculation and conversion algorithm used in HFQ can be formally presented as follows.

For any session i and its ascent node k, let C_i^m and $F_{i,k}^m$ denote the m^{th} packet of session i and its finish time at node k, respectively. If we further define a_i^m and L_i^m as the arrival time and the length of C_i^m, respectively, the finish time of the packet at the immediate parent node $p(i)$ is calculated as

$$F_{i,p(i)}^m = S_i^m + \frac{L_i^m}{\varphi_i R}, \quad (18)$$

where $S_i^m = \max\{F_{i,p(i)}^{m-1}, V_{p(i)}(a_i^m)\}$ is the start time of the packet. Here, recall that $V_{p(i)}$ reflects the service progress at the immediate parent node $p(i)$. Therefore, from the definition of $F_{i,p(i)}^m$, $F_{i,p(i)}^m - V_{p(i)}(\tau)$ can be interpreted as the amount of virtual time left until the packet C_i^m departs from the ideal E-HPS (or equivalently, HPS or H-GPS) server, which is estimated at real time τ under the assumption that all sessions will receive services proportional to their nominal service weight φ_i after time τ. Henceforth, we will refer to the difference between the finish time of a packet and the LVT at the immediate parent node as the *Nominal Residual Sojourn Time* (NRST) of the packet. Based on the interpretation of NRST, the ideal fluid models can be approximated in one step. In other words, one may derive a packet version of H-GPS in the same way as WFQ is derived from the GPS model, by *sorting all packets in the entire system in the order of their NRSTs and choosing a packet with the smallest NRST for transmission.* In fact, the resulting packet version H-GPS is equivalent to HFQ.

The physical meaning of transmitting packets in the order of their NRSTs can be explained as follows. As discussed in Section II-A, the order of the actual finish times of backlogged packets is dependent to the future packet arrival in a H-GPS system. This means that in order to sort packets according to their anticipated finish times, a certain assumption must be made on the future packet arrival. In the case of the HFQ, the NRSTs of packets are used to compare their anticipated departure time, and this implies that all sessions are assumed to receive services proportional to their nominal service weights after the time τ, at which the NRSTs are computed. Since such service distribution will take place if all sessions will be backlogged after time τ, *in a sense, the worst-case scenario is assumed for the estimation of the anticipated departure time of all backlogged packets in a HFQ system.* Therefore, remembering the fundamental and practical limitations in approximating the ideal fluid models by packet-based schedulers, one can see that transmitting packets in the order of their NRST will be one of the most sensible and fairest approaches, and is not expected to result in serious performance degradation due to hierarchical service structure as discussed in Section II-A.

While sorting (and transmitting) backlogged packets in the order of their NRSTs is the main idea in HFQ, maintaining a single queue of all packets sorted according to their NRSTs is not a simple task. In the case of a GPS system, once packets are sorted according to their NRSTs (or finish times), the relative order of packets in the queue will remain unchanged. On the other hand, in the case of a H-GPS system, depending on the session states, the relative order of existing packets' NRSTs may be reversed because, in the hierarchical processor sharing model, service rates at non-leaf nodes may not always be proportional to their nominal service weights.

A way to get around this problem is to compare the NRSTs of backlogged packets in the form of a tournament only when a packet must be chosen for transmission. Now, let node k be a non-leaf node, and consider two packets, C_i^m and C_j^n from its child sessions i and j, whose NRSTs need to be compared to decide the packet that will be forward to the next level. In other words, assuming that τ denotes the time when the comparison is made, if $F_{i,p(i)}^m - V_{p(i)}(\tau) > F_{j,p(j)}^n - V_{p(j)}(\tau)$, then C_j^n will be forwarded to node $p(k)$ for the next match, and C_i^m will be forwarded otherwise. On the other hand, it follows from (12) that

$$V_{p(i)}(\tau) = V_k(\tau) + \sum_{h=1}^{H_{i,k}-1} \beta_{p^h(i)}(\tau) \quad \text{and}$$

$$V_{p(j)}(\tau) = V_k(\tau) + \sum_{h=1}^{H_{j,k}-1} \beta_{p^h(j)}(\tau),$$

where $H_{i,k}$ denotes the number of ancestor nodes of session i through (and including) node k in the scheduling hierarchy. From these equations, note that the NRSTs of C_i^m and C_j^n can be rewritten in terms of $V_k(\tau)$ as

$$F_{i,p(i)}^m - \sum_{h=1}^{H_{i,k}-1} \beta_{p^h(i)}(\tau) - V_k(\tau) \quad \text{and}$$

Table 1. HFQ algorithm applied to Fig. 2 (bold-faced finish times represents the winner of each competition).

Time	Packets compared	F.T. at Node 3	F.T. at Node 2	F.T. at Node 1	Transmitted packet
t=0	None	-			C_7^1
	C_5^1	**0+6**	6-0		
	C_6^1		**0+3**	3-0	
	C_7^1			**0+2**	
t=1	C_4^1	**2+3**	5-1		C_6^1
	C_5^1	6			
	C_6^1		**3**	**3-0**	
	None			-	
t=1.75	C_4^1	**5**	**5-1**	**4-0**	C_4^1
	C_5^1	6			
	None		-		
	None			-	
t=2.125	None	-			C_5^1
	C_5^1	**6**	**6-1**	**5-0**	
	None		-		
	None			-	
t=2.875	C_4^2	5+9			C_5^2
	C_5^2	**6+2**	**8-1**	**7-3/4**	
	None		-		
	None			-	
t=3.125	C_4^2	14			C_6^2
	C_5^3	**8+4**	12-1		
	C_6^2		**5+3**	**8-1**	
	None			-	
t=3.875	C_4^2	14			C_5^3
	C_5^3	**12**	**12-1**	**11-1**	
	None		-		
	None			-	
t=4.375	C_4^2	**14**	**14-1**	**13-1**	C_4^2
	None	-			
	None		-		
	None			-	
t=5.5	None	-			C_7^2
	None		-		
	None			-	
	C_7^2			**8**	

$$F_{j,p(j)}^n - \sum_{h=1}^{H_{j,k}-1} \beta_{p^h(j)}(\tau) - V_k(\tau),$$

respectively. Therefore, we can only compare

$$F_{i,p(i)}^m - \sum_{h=1}^{H_{i,k}-1} \beta_{p^h(i)}(\tau) \quad \text{and}$$

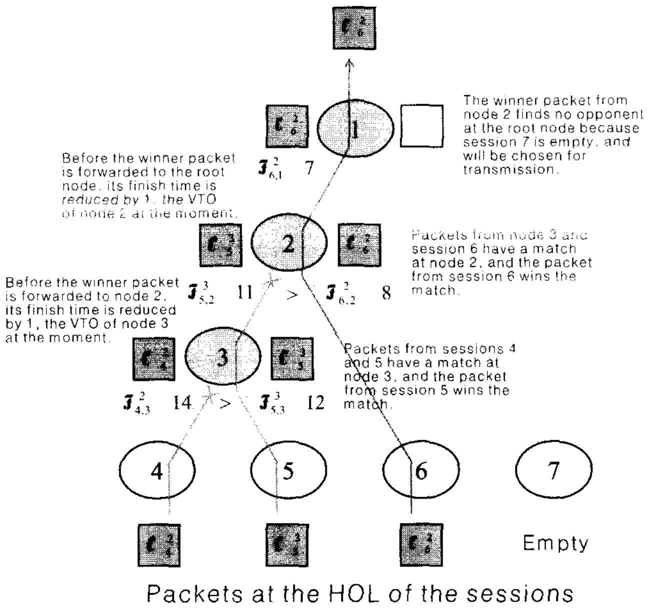
$$F_{j,p(j)}^n - \sum_{h=1}^{H_{j,k}-1} \beta_{p^h(j)}(\tau),$$

to decide which packet to forward to $p(k)$. This implies that $V_k(\tau)$, the LVT at node k is not really required for the comparison, if the finish time of a packet C_i^m at node k is defined as

$$F_{j,k}^m = F_{i,p(i)}^m - \sum_{h=1}^{H_{i,k}-1} \beta_{p^h(i)}(\tau).$$

One may think of $F_{i,k}^m$ as the score of the packet C_i^m adjusted for the match at node k, which has its own scoring system represented by $V_k(\tau)$. Further, the finish time of a packet from a session i at its ancestor nodes can be computed recursively as

$$F_{i,p^h(i)}^m = F_{i,p^{h-1}(i)}^m - \beta_{p^{h-1}(i)}(\tau), \tag{19}$$



The winner packet from node 2 finds no opponent at the root node because session 7 is empty, and will be chosen for transmission.
1
Before the winner packet is forwarded to the root node, its finish time is reduced by 1, the VTO of node 2 at the moment.
$\mathfrak{F}^2_{6,1}$ 7
2
Packets from node 3 and session 6 have a match at node 2, and the packet from session 6 wins the match.
Before the winner packet is forwarded to node 2, its finish time is reduced by 1, the VTO of node 3 at the moment.
$\mathfrak{F}^3_{5,2}$ 11 > $\mathfrak{F}^2_{6,2}$ 8
3
Packets from sessions 4 and 5 have a match at node 3, and the packet from session 5 wins the match.
$\mathfrak{F}^2_{4,3}$ 14 > $\mathfrak{F}^3_{5,3}$ 12
4
5
6
7
Empty
Packets at the HOL of the sessions

Fig. 7. Packet selection procedure at time $t = 3.125$ in the HFQ system given in Fig. 2.

Table 2. Simulation configuration.

Session	Inactive Period	Avg Rate (Mbps)	Packet Size (Bytes)	Nominal Weight	Guaranteed Rate (Mbps)
1		0.5	1000	0.0625	0.5
2	0.5 - 2.5	2.0	900	0.0625	0.5
3		2.0	200	0.2500	2.0
4	1.0 - 3.0	2.0	700	0.0625	0.5
5	1.5 - 3.5	2.0	950	0.0625	0.5
6		2.0	200	0.2500	2.0
7	2.0 - 4.0	4.0	1000	0.1250	1.0
8		3.0	750	0.1250	1.0

and this computation will be necessary only if the packet is the winner of the match at node $p^{h-1}(i)$.

(18) is similar to the equation for computing finish times of packets in WFQ except that the nominal service weight φ_i is used instead of the local service weight ϕ_i. This change is necessary to reflect the hierarchical nature of HFQ and thereby to maintain service fairness in the global scale. On the other hand, (19) states how the VTOs can be used to adjust the finish time along the hierarchy. Since VTOs are derived from spare bandwidth of idling sessions as illustrated in (14) or (17), the finish time adjustment can be thought of as a means to distribute spare bandwidth at a non-leaf node to its busy child nodes. For instance, if all child nodes of node k are backlogged over a time interval, then $\beta_k(\tau)$ will remain unchanged through out the interval, and no extra service will be provided to the child nodes. Also, when there is no spare bandwidth in the system (i.e., all sessions are backlogged), the VTO of every intermediate node will remain unchanged, and the HFQ system degenerates to a simple WFQ system. This is an expected result because a H-GPS, HPS, or E-HPS server degenerates to a single-level GPS server when all sessions are backlogged, and HFQ approximates these ideal fluid models in the same way as WFQ approximates the single-level GPS model. On the contrary, a H-PFQ server does not generally degenerate to a simple single-level packet fair queueing algorithm when all sessions are backlogged.

For an illustration of the HFQ algorithm, it is applied to the hierarchical packet scheduling scenario in Fig. 2, and the detailed procedure to determine the order of packet transmission is described in Table 1. Note that a tournament competition among the HOL packets of active sessions is initiated when a packet needs to be chosen for transmission; i.e., when a new busy period starts or when a packet transmission completes. The packet selection procedure taking place at time $t = 3.125$ is also schematically illustrated in Fig. 7.

V. SIMULATION RESULTS

In this section, we present simulation results for delay bound and hierarchical link sharing characteristics of the HFQ algorithm. The simulation is undertaken in NS [11]. For performance comparisons, the H-WFQ and H-WF2Q algorithms [5] are also simulated under identical conditions.

Table 2 describes the simulation configuration parameters of sessions 1 through 8 for the scheduling hierarchy shown in Fig. 1. Every session is associated with a CBR (Constant Bit Rate) traffic source with "randomizing dither" on the inter-packet departure intervals (refer to the NS manuals [11] for details). Packet sizes are kept constant. The traffic sources of sessions 1, 3, and 6 are leaky-bucket constrained in order to regulate their average input rates same as their guaranteed rates, and maximum allowable burst sizes are twice as long as their packet sizes. Other traffic sources generate significantly more traffic than their guaranteed rates to take advantage of spare capacity in the system. The simulation is carried out for 5 seconds, during which sessions 2, 4, 5, and 7 have inactive periods as shown in Table 2. The link rate is set to 7.0 Mbps, the sum of the guaranteed rates of sessions.

A. Hierarchical Link Sharing

Fig. 8 illustrates link sharing characteristics of H-WFQ, H-WF2Q and HFQ, where each bandwidth-time graph depicts throughput responses for sessions 1 through 8 all together. As can be seen in the figure, HFQ distributes bandwidth in the same manner that H-WFQ and H-WF2Q do. When session 2 stops sending traffic at time 0.5, priority to utilize the unused bandwidth of session 2 is first given to its sibling sessions 1 and 3. However, since both sessions 1 and 3 are not capable of absorbing the whole unused bandwidth, generating only as much traffic as their guaranteed rates, the unused bandwidth is spilled over to sessions 4 and 5. When session 4 stops at time 1.0, session 5 becomes the sole beneficiary of the unused bandwidth of both sessions 3 and 4. At time 1.5 session 5 goes to sleep, and the unused bandwidth of sessions 3, 4, and 5 are shared between sessions 7 and 8. Then, session 8 takes all the spare bandwidth after session 7 becomes inactive at time 2.0.

When session 2 wakes up at time 2.5, the unused bandwidth of sessions 4 and 5 becomes available exclusively to session 2. Then, at time 3.0 when session 4 wakes up, session 2 surrenders the entire spare bandwidth of session 5 and some of its share for the spare bandwidth of session 7 to session 3. Note that after session 5 restarts sending traffic at time 3.5, the unused bandwidth

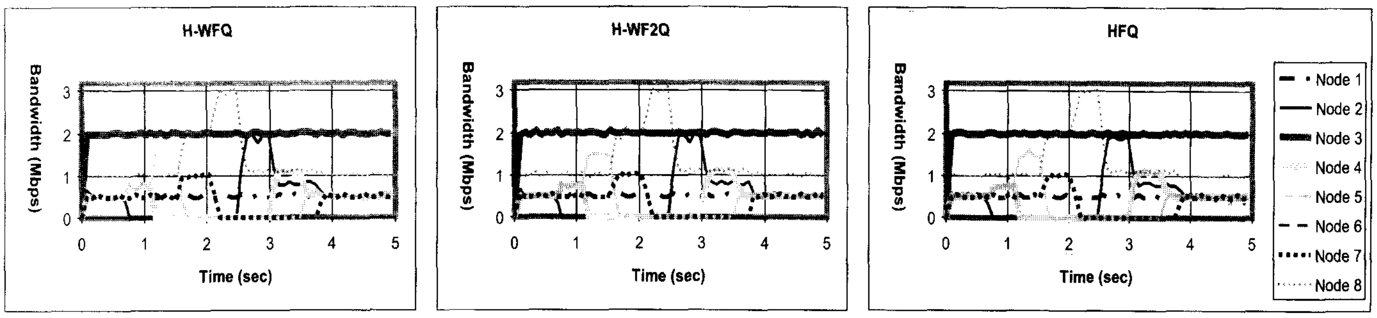


Fig. 8. Link sharing characteristics.

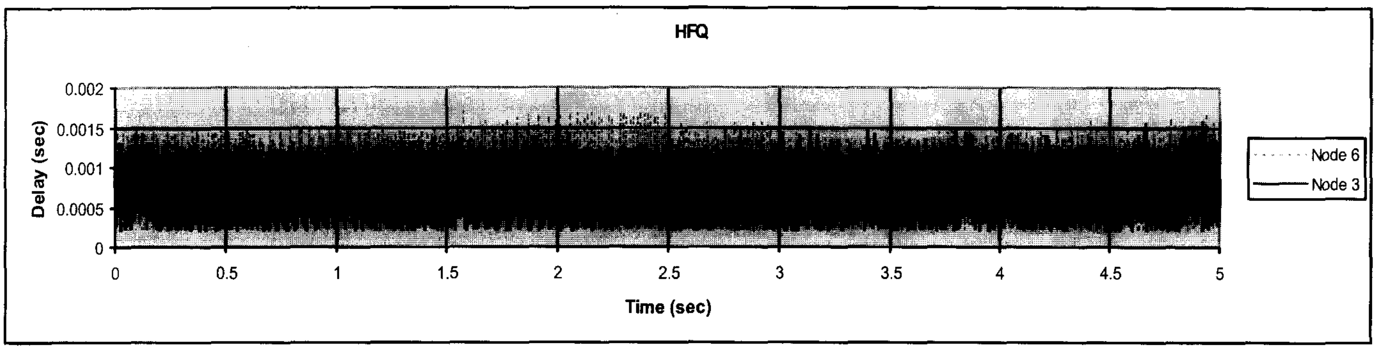


Fig. 9. Delay responses of sessions 3 and 6 for HFQ.

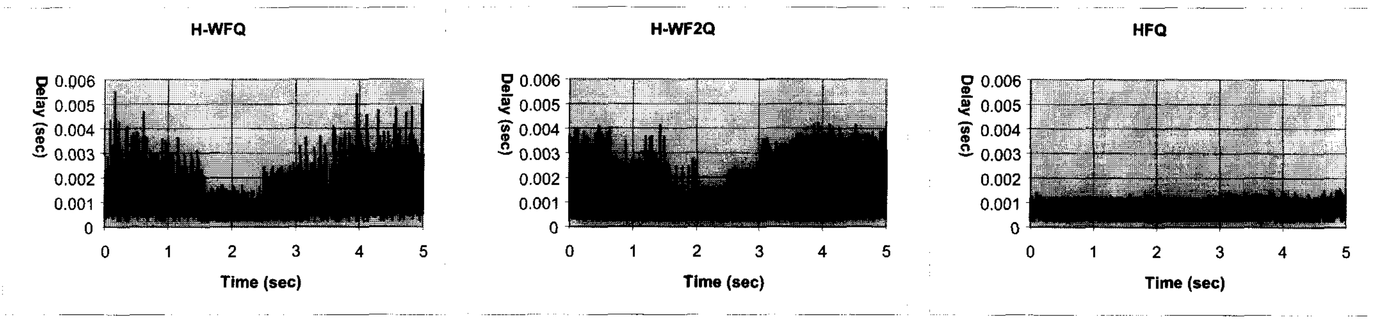


Fig. 10. Delay response of session 3 for H-WFQ, H-WF2Q, and HFQ.

of session 7 is shared among sessions 2, 4, 5, and 8 in proportion to their effective service weights. Finally, when session 7 wakes up at time 4.0, every session gets as much throughput as one's guaranteed rate.

B. Maximum Delay Bound

Layer-independent delay characteristics of HFQ are illustrated in Fig. 9, where delay responses of packets for sessions 3 and 6 are depicted together in a delay-time graph. The traffic sources of sessions 3 and 6 are conditioned exactly the same. Since both sessions are leaky-bucket constrained with the same parameters, their maximum delay bounds are expected to be the same regardless of their relative locations in the scheduling hierarchy. Indeed, as can be seen from the figure, the delay response of session 3, which is three generations away from the root node, is no worse than that of session 6, which is a direct child of the root node. In fact, the delay response of node 3 is slightly better than that of node 6 for many occasions. This is because of the "hierarchical statistical multiplexing" effect that the idling periods of sessions 2, 4, and 5 help enhancing the delay response of node 3. Session 6 benefits less from them due to the hierarchical link sharing policy of HFQ. Nevertheless, the maximum delay statistics for both nodes 3 and 6 are of comparable level. In order to demonstrate the improved layer-independent delay characteristics of HFQ, delay responses of node 3 for H-WFQ, H-WF2Q, and HFQ under the same simulation settings, are compared side by side in Fig. 10. Note that delay responses of session 3 for H-WFQ and H-WF2Q are clearly worse than the one for HFQ. Although the delay performance of H-WF2Q is slightly better than that of H-WFQ, their delay characteristics reveal serious inter-session dependence, which is not the case with HFQ.

VI. RELATED WORK AND IMPLEMENTATION ISSUES

Most closely related work to HFQ is H-PFQ [5]. As have been presented so far, the very motivation for the development of HFQ was to improve the layer-dependent delay characteristics of H-PFQ while retaining the hierarchical link sharing capability.

H-FSC [12] is another hierarchical packet scheduling algorithm that is capable of providing layer-independent delay performance. The approaches used in H-FSC, however, are very different from those used in HFQ. While HFQ is based only on a single discipline, H-FSC employs two independent scheduling disciplines to complement each other's shortcomings. Under normal circumstances, the rate-based discipline performs scheduling operation to achieve hierarchical link sharing. In case of a risk for violation of performance guarantees, the real-time discipline takes over the operation. In contrast, HFQ is a "pure-bred" rate-based scheduling algorithm. Besides the layer-independent delay performance capability, H-FSC is capable of decoupling delay and bandwidth allocation. Our approach is to separate the two issues for independent handling. The issue of decoupled delay and bandwidth allocation for non-hierarchical packet scheduling is addressed in CPS [13]. The integration of HFQ and CPS is currently under progress.

Another well-known hierarchical packet scheduling algorithm is CBQ [4]. While CBQ can specify flexible hierarchical link sharing policies, its capability to support delay performance is limited. Like H-FSC, CBQ consists of two separate scheduling mechanisms: The link-sharing scheduler for hierarchical bandwidth distribution and the general scheduler for priority-based packet scheduling. Unlike H-FSC, however, the link-sharing scheduler is not a real scheduler that makes a direct decision for packet selection. Rather, it provides a feedback to the general scheduler to condition the eligibility of packets. Due to the complexity of the link sharing rules and the nature of the priority-based scheduler, it is hard to formulate the performance characteristics of CBQ.

Although HFQ is expected to provide improved delay performance, its straightforward implementation (as illustrated in Section IV) is likely to entail increased computational complexity. This is mainly because each non-leaf node requires the global system state information to track its LVT. For instance, as one can see from (11), the increase rate of the LVT at a non-leaf node k will be influenced by the states of all nodes (or sessions) either directly or indirectly. Consequently, difficulties in exactly tracking virtual times, such as the classical iterated deletion problem [8], will become more critical for the implementation of HFQ than for the implementation of H-PFQ where the virtual times can be computed only based on the local state information (i.e., the states of child nodes). However, on the bright side, it is the hierarchical and recursive finish-time computation/comparison mechanism built upon the notion of the distributed virtual time system but not the distributed virtual time system itself that enables HFQ to achieve the layer-independent delay performance. In other words, the accuracy of LVTs may not be as critical factor of the layer-independent delay performance as one may expect, and can be compromised to reduce the computational complexity and the implementation cost of HFQ. In fact, we are devoting considerable portion of our research effort to the development of computationally efficient HFQ algorithms, and obtained a HFQ algorithm, the computational complexity of which is comparable to that of H-PFQ while exhibiting delay performance indistinguishable from that of the genuine HFQ.[3] Further, it turned out that there exist numerous ways to alleviate the computational burden for tracking LVTs in a HFQ system. In particular, most of the techniques, such as SCFQ and SFQ, that reduce the computation for tracking the virtual time in a single-level PFQ system, can be extended to the HFQ system. Therefore, we are continually investigating all possible approaches to extend single-level PFQ algorithms to multi-level PFQ algorithms based on HFQ, and it is still too early to make any conclusive remark on the complexity of HFQ.

VII. CONCLUSION

In this paper, we have introduced HFQ, a new fair hierarchical packet scheduling algorithm, which can achieve tighter maximum delay bounds than H-PFQ. HFQ closely approximates H-GPS, delivering consistent worst-case delay performance regardless of relative locations in the scheduling hierarchy. "Global-scale" comparison of time-stamps, and "local-scale" maintenance of "service credits" are two distinct features of HFQ, which allow "globally-fair" performance and "hierarchically-fair" link sharing.

As data network services develop into complex structures, the role of hierarchical packet scheduling becomes more and more essential. For various technological, administrative and management purposes, network traffic needs to be organized into layers of scheduling groups for differentiated treatments. Should real-time applications such as video be supported, layer-independent performance characteristics of a hierarchical packet scheduling algorithm becomes extremely important for scalable provisioning of hierarchical network services. That is, no matter which traffic management group(s) a real-time application belongs to, QoS requirements should be met in absolute terms. In such an environment, HFQ can make a major contribution for opening up a new way to implement hierarchical packet scheduling in network switches and routers.

REFERENCES

[1] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," IETF RFC-1633, June 1994.

[2] S. Blake *et al.*, "An architecture for differentiated services," IETF RFC-2475, Dec. 1998.

[3] A. D.-S. Jun, "A soft network approach for network programming: Node model," Technical Report, ECE Department, University of Toronto, Dec. 1999.

[4] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

[5] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queuing algorithms," *IEEE/ACM Trans. Networking*, vol. 5, no. 5, pp. 675–689, Oct. 1997.

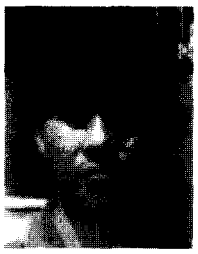
[6] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Internetworking: Research and Experience*, vol. 1, pp. 3–26, 1990.

[7] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

[8] S. Keshav, *An Engineering Approach to Computer Networking*, Addison-Wesley, 1997.

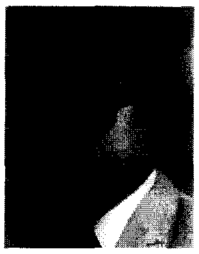
[3]In fact, the simulation results for HFQ systems presented in Section V is obtained through this modified HFQ algorithm. Therefore, the simulation results also tell us that tracking the exact values of LVTs is not a necessary condition for providing layer-independent delay performance.

[9] A. D.-S. Jun, J. Choe, and A. Leon-Garcia, "Hierarchical fair queueing: A credit-based approach for hierarchical link sharing," preprint. Available at http://htnl.sogang.ac.kr/publications/preprints/aja1.pdf.
[10] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," in *Proc. SIGCOMM'90*, Sept. 1990, pp. 19–29.
[11] The Network Simulator - version 2, Available at http://www.isi.edu/nsnam /ns/.
[12] I. Stoica, H. Zhang, and T. S. Eugene Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time and priority services," in *Proc. SIGCOMM'98*, Canada, Aug. 1998.
[13] A. D.-S. Jun, J. Choe, and A. Leon-Garcia, "Credit-based processor sharing for decoupled delay and bandwidth allocation," *IEEE Commun. Lett.*, vol. 5, no. 4, pp. 178–180, Apr. 2001.



Andrew Do-Sung Jun received the B.S. and M.S. degrees from the Departments of Engineering Science and Electrical and Computer Engineering, respectively, all at the University of Toronto. Currently, he is pursuing his Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Toronto, while he is also working as a research scientist at Telcordia Technologies. His research areas include hierarchical VPN architecture, network resource management, and packet scheduling. His job responsibilities are data modeling, and architecture and prototype development in the areas of 2G/3G cellular networks and wireless LANs.

Jinwoo Choe received the B.S. and the M.S. degrees from Seoul National University in 1990 and 1992, respectively, and the Ph.D. degree from Purdue University in 1998. From 1998 to 2001, he was with the Edward S. Rogers, Sr. Department of Electrical and Computer Engineering at University of Toronto as an assistant professor. He is currently with the Department of Electronic Engineering at Sogang University, and his research area of interest encompasses network traffic modeling, performance analysis, optical LAN, multimedia caching, and packet scheduling algorithms.



Alberto Leon-Garcia received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Southern California, in 1973, 1974, and 1976 respectively. He is a full professor in the Department of Electrical and Computer Engineering of the University of Toronto, and became an IEEE fellow for "For contributions to multiplexing and switching of integrated services traffic" in 1999. He was editor for Voice/Data Networks for the IEEE Transactions on Communications from 1983 to 1988 and editor for the IEEE Information Theory Newsletter from 1982 to 1984. He was guest editor of the September 1986 Special Issue on Performance Evaluation of Communications Networks of the IEEE Selected Areas on Communications. He is also author of the textbooks Probability and Random Processes for Electrical Engineering, and Communication Networks: Fundamental Concepts and Key Architectures.