This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

A Practical and Efficient Implementation of WF^{2}Q+

George N. Rouskas, Zyad Dwekat
Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206

***Abstract*— The WF^{2}Q+ scheduler combines all three properties that are important to a fair queueing algorithm: a tight delay bound, a small worst-case fair index value, and a relatively low worst-case complexity of $O(\log n)$ for a link with n flows. We present a new implementation of WF^{2}Q+ in which both the number of packet sorting operations and the computation of the virtual time function are independent of the number n of flows. Our implementation exploits two widely observed characteristics of the Internet, namely that service providers offer some type of tiered service with a small number of service levels, and that a small number of packet sizes dominate. Our scheduler combines provably good performance with amenability to hardware implementation in high-speed routers.**

I. Introduction

In packet-switched networks, the scheduling algorithm is central to the QoS architecture. A scheduler is desirable to possess three important properties [8]: *fairness*, to provide isolation among competing flows and ensure that each flow receives its fair share of the link bandwidth; *bounded delay*, so as to guarantee a bounded end-to-end delay to interactive applications; and *low complexity*, so as to be possible to implement in hardware and operate at wire speeds.

In general, packet schedulers can be classified as either *timestamp-based* or *frame-based*. Timestamp schedulers emulate the ideal but unimplementable generalized processor sharing (GPS) algorithm by maintaining a virtual time function. Packets are assigned a timestamp based on the virtual time value at the time of their arrival, and are transmitted in increasing order of timestamp. In general, timestamp schedulers have good delay and fairness properties, but high implementation complexity, hence there has been limited deployment of such schedulers in high-speed routers.

The complexity of timestamp schedulers arises from two factors. First, it is necessary to select among the head-of-line packets of the n backlogged flows the one with the smallest timestamp to serve next; this operation takes time $O(\log n)$ using a priority queue. Whereas current router technology makes it possible to support millions of flows, each with its own queue, the logarithmic complexity and the fact that the priority queue structure is not suited to hardware implementation pose significant challenges. Second, the worst-case complexity of computing the virtual time function of GPS, as required by weighted fair queueing (WFQ) [7] and worst-case fair weighted fair queueing (WF^{2}Q) [3], is $O(n)$ [5]. WF^{2}Q+ [2] uses a different virtual time function that can be computed in $O(\log n)$ time [2]; since it also provides tight delay bounds and achieves worst-case fairness, WF^{2}Q+ is the best known packet fair algorithm. Other schemes, including self-clocked fair queueing [5], use simplified virtual time functions that are more efficient to compute, but still require that packets be sorted in increasing order of timestamp. Furthermore, it has been shown that achieving a delay bound relative to GPS that is independent of the number of flows is impossible if the scheduler has a complexity below $O(\log n)$ [13].

Frame-based schedulers typically operate by dividing time into frames. Within each frame, flows are mapped to time slots of fixed or variable length and are served in a round-robin manner. By eliminating the need for packet sorting, schedulers such as deficit round robin (DRR) [9], are easy to implement and have been widely deployed in high-speed routers. However, frame-based schedulers have poor delay bound and output burstiness properties. More recently, schedulers such as stratified round robin [8] incorporate some elements of timestamp schedulers into a frame-based scheme, so as to improve the delay and output burstiness while maintaining low implementation complexity.

The main contribution of our work is a new implementation of the WF^{2}Q+ scheduler which ensures that the two main operations, namely, computing the virtual time function and selecting the next packet to be transmitted, can be performed in time that is independent of the number n of flows. Our work is motivated by two important observations. First, most Internet service providers offer some type of *tiered service*, in which users may select only from a small set of service *tiers*. Service tiers are either based on the bandwidth hierarchy of the underlying infrastructure (e.g., DS-1, DS-3, OC-3, etc.), or are determined in some *ad-hoc* manner (e.g., the various ADSL tiers available through different providers). The practical implication of a tiered-service is that the rates assigned to flows (equivalently, the flow weights in the fair queueing system) are not arbitrary, but are limited to a small set of predetermined values. We emphasize that tiered-service is different from the approach in [8]; there, flows with arbitrary weights are "stratified" into classes using exponential grouping, while we assume that *all flows* within a service tier are assigned the *same* weight. The second observation is that in the Internet, the vast majority of packets have a fixed length that takes one of a small number of values [10], [12]. As we explain later, the low sorting complexity of this scheduler does not contradict the findings of [13] regarding the tradeoffs between complexity and delay bounds of scheduling algorithms.

Following a brief review of important concepts in Section II, in Section III we introduce the intra-tier and inter-tier structure of our scheduler. In Section IV, we first consider a network with fixed size packets and present an intra-tier scheduler

This work was supported by the NSF under grant CNS-0434975.

1-4244-0353-7/07/$25.00 ©2007 IEEE

of constant-time complexity. In Section V, we generalize the scheduler structure to variable-size packets by exploiting the observed Internet packet length distribution to keep complexity low. We conclude the paper in Section VI.

II. The WFQ, WF^{2}Q, and WF^{2}Q+ Schedulers

A virtual time function $V(t)$ was proposed in [7] to track the progress of GPS. The rate of change of $V(t)$ is:

$$\frac{\vartheta V(t+\tau)}{\vartheta \tau} = \frac{1}{\sum_{i \in B(t)} \phi_i} \quad (1)$$

where $B(t)$ denotes the set of backlogged flows at time t and ϕ_i is the weight assigned to flow i. Let r be the rate of the link (server). In GPS, if flow i is backlogged at time t, it receives a rate of $\frac{\vartheta V(t+\tau)}{\vartheta \tau}\phi_i r$; in other words, $V(t)$ is the marginal rate at which backlogged flows receive service in GPS.

Suppose that the k-th packet of flow i arrives at time a_i^k, and has length L_i^k. Let S_i^k and F_i^k denote the virtual times at which this packet begins and completes service, respectively Letting $F_i^0 = 0$ for all flows i, we have [7]:

$$S_i^k = \max\{F_i^{k-1}, V(a_i^k)\} \quad (2)$$

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i} \quad (3)$$

The WFQ scheduler serves packets in increasing order of their virtual finish times F_i^k, a policy referred to as "smallest virtual finish time first (SFF)" [2]. This policy can be implemented in time $O(\log n)$, where n is the number of flows, using a priority queue data structure. In addition, there is the cost of maintaining the virtual time function $V(t)$. The worst-case complexity of computing $V(t)$ can be $O(n)$, although the average-case complexity is $O(1)$ [5]. WFQ provides a delay bound that is within one packet transmission time of that provided by GPS [7]. However, WFQ may introduce substantial unfairness relative to GPS in terms of the worst-case fairness index (WFI), a metric introduced in [3] to represent the maximum time a packet arriving to an empty queue will have to wait before receiving its guaranteed service rate. Specifically, GPS has a WFI of zero, but the WFI of WFQ increases linearly with the number of flows n.

The WF^{2}Q algorithm introduced in [3] improves on WFQ in terms of WFI by employing a "smallest eligible virtual finish time first (SEFF)"policy for scheduling packets. A packet is *eligible* if its virtual start time is no greater than the current virtual time; hence, the WF^{2}Q scheduler only considers the packets that have started service in GPS to select the one to be transmitted next. It has been shown [3] that WF^{2}Q is work-serving, provides the same tight delay bound as WFQ, and is worst-case fair in the sense that its WFI is a constant independent of the number n of flows. However, the worst-case complexity of WF^{2}Q is $O(n)$, identical to that of WFQ, as both schedulers need to compute the function $V(t)$.

A lower-complexity scheduler, WF^{2}Q+ was introduced in [2]. WF^{2}Q+ provides the same delay bound and WFI as WF^{2}Q, but uses a different virtual time function which can be computed in $O(\log n)$ time. The new function is [2]:

$$V_{WF^2Q+}(t+\tau) = \max\left\{V_{WF^2Q+}(t), \min_{i \in B(t)}\left\{S_i^{h_i(t)}\right\}\right\} \quad (4)$$

where $h_i(t)$ is the sequence number of the packet at the head of flow i's queue at time t, and $S_i^{h_i(t)}$ is the virtual start time of that packet. The minimum operation in the right-hand side of (4) can be performed in time $O(\log n)$ in the worst-case using a priority queue structure, hence the overall complexity of WF^{2}Q+ is $O(\log n)$, significantly lower than the $O(n)$ complexity of WFQ and WF^{2}Q.

III. Tiered Service Fair Queueing (TSFQ)

We consider a link which serves n flows and employs per-flow queueing, i.e., it allocates a FIFO buffer to each flow. The link supports p distinct tiers of service, where $p \ll n$ is a small constant (e.g., $p \approx 10 - 15$). The l-th tier is characterized by a positive real weight $\phi_l, l = 1, \ldots, p$. Each flow i is mapped to one of the p service levels, i.e., it is assigned one of the p weights ϕ_l; we assume that this assignment remains fixed throughout the duration of the flow. The mapping of flows to service tiers is performed at the time the flow is admitted to the network by taking into account the QoS requested by the user. In this paper we assume that the link is configured with the number p of service tiers and the associated weights ϕ_j; these parameters are determined in advance by the network operator using information about the user demands. Our research group has developed techniques based on discrete location theory to select an optimal set of service tiers for a given stochastic distribution of user requests [1], [6].

Similar to WF^{2}Q+, our tiered service fair queueing (TSFQ) scheduler uses the virtual time function in (4), and also employs the SEFF policy to serve packets. The TSFQ scheduler logically consists of two parts, as shown in Figure 1. The first part comprises of p identical *intra-tier schedulers*, while the second part is a single *inter-tier scheduler*. The l-th intra-tier scheduler is responsible for using the SEFF policy to select *the eligible packet with the minimum virtual finish time within the flows in the l-th service tier*, $l = 1, \ldots, p$. The inter-tier scheduler simply serves the packet with the smallest virtual finish time among the p packets selected by the corresponding intra-tier schedulers. Since p is a small constant for the given link, the packet to be transmitted next can be determined in time that is independent of the number of flows, and in fact, this operation can be performed in constant time in hardware. Hence, the implementation of the inter-tier scheduler is straightforward and does not require any priority queue data structure to be maintained.

IV. Intra-Tier Scheduler: The Fixed-Packet Case

The l-th TSFQ intra-tier scheduler, $l = 1, \ldots, p$, serves flows belonging to the l-th service tier and have been assigned the same weight ϕ_l. The p intra-tier schedulers are identical and operate independently of each other. Therefore, in this and the next section we make the assumption that all flows have

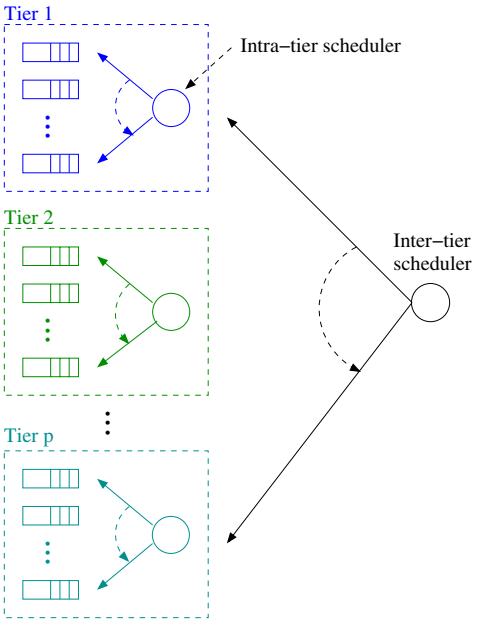

Fig. 1. Logical diagram of the TSFQ scheduler with p service tiers

identical weights. For simplicity, we let ϕ denote the weight assigned to all the flows served by the scheduler.

In this section we make the additional assumption that *all packets of all flows have constant size* L (i.e., $L_i^k = L \ \forall \ i,k$). We will remove this assumption in the next section; however, we note that the implementation we present in this section is of practical importance to ATM networks.

In a system with fixed-size packets and flows of identical weight, sorting packets according to their virtual start times produces an identical order to sorting them according to their virtual finish times. This property is formally expressed in the following lemma (note that the property is trivially true for packets belonging to the same flow).

Lemma 4.1: Consider flows i and j with $\phi_i = \phi_j = \phi$ and packets of fixed size L. Let S_i^k be the virtual start time of the k-th packet of flow i, and S_j^l be the virtual start time of the l-th packet of flow j. Then:

$$S_i^k \ \leq \ S_j^l \ \Leftrightarrow \ F_i^k \ \leq \ F_j^l \qquad (5)$$

Proof. Under the assumption of fixed packet size and identical weights, the second term of expression (3) is constant, hence:

$$S_i^k \ \leq \ S_j^l \ \Leftrightarrow \ S_i^k + \frac{L}{\phi} \ \leq \ S_j^l + \frac{L}{\phi} \ \Leftrightarrow \ F_i^k \ \leq \ F_j^l \qquad (6)$$

■

Queue structure and operation. The intra-tier scheduler for fixed-length packets consists of a simple FIFO scheduler, as illustrated in Figure 2, and operates as follows. Arriving packets of new or previously idle flows are inserted at the tail of the FIFO immediately upon arrival; from (2), the virtual time at the arrival instant is the virtual start time of such packets. A head-of-line packet of a backlogged flow, however, is not released to the FIFO queue at the instant the previous packet of the same flow leaves the system. Instead, the packet

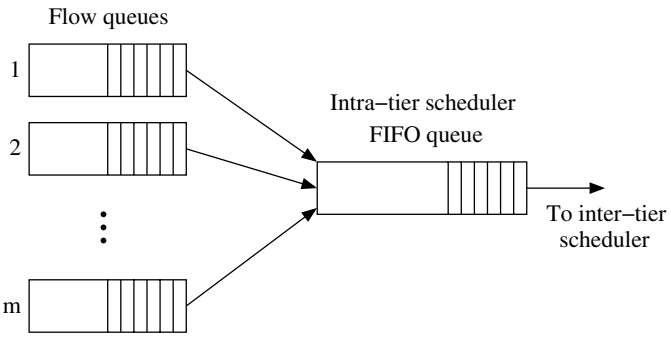

Fig. 2. Queue structure of the intra-tier scheduler for fixed-size packets

is held in the flow queue until the instant the virtual time becomes equal to its virtual start time, and then inserted into the FIFO queue. This operation is identical to that of the WF^{2}Q+ scheduler [2], and ensures that no packet will start service before it is eligible for service under GPS.

We have the following results.

Lemma 4.2: For the flows of a given tier, the intra-tier scheduler of Figure 2 is identical to the WF^{2}Q+ scheduler [2].

Proof. Since packets are released to the FIFO queue at their virtual start times, packets in the FIFO queue are sorted in increasing order of their virtual start times. Because of Lemma 4.1, the queue is sorted in increasing order of the packet virtual finish times, which is the order in which packets are served under WF^{2}Q+. Since arrivals to the FIFO queue take place at exactly the same instants these arrivals take place under WF^{2}Q+ and the order of service is identical, the two schedulers are identical under the assumption of flows with fixed-size packets and identical weights. ■

Lemma 4.3: The TSFQ scheduler consisting of p intra-tier schedulers and one inter-tier scheduler is identical to WF^{2}Q+.

Proof. Each of the p intra-tier schedulers is identical to WF^{2}Q+ and the inter-tier scheduler serves the p packets at the head of the p intra-tier FIFO queues in increasing order of their virtual start (equivalently, finish) times. Consequently, the TSFQ scheduler is identical to WF^{2}Q+. ■

Lemma 4.4: In the TSFQ scheduler, the virtual time function (4) can be computed in $O(1)$ time.

Proof. Since virtual finish time order is equivalent to virtual start time order, the packet with the smallest start time within a given tier is the one at the head of the respective FIFO queue. Thus, the minimum operation in (4) reduces to selecting the packet with the smallest finish time among the ones at the head of the p intra-tier queues; as we mentioned earlier, this operation can be performed in constant time. ■

Hence, in addition to possessing the worst-case fairness and delay properties of WF^{2}Q+, the TSFQ (intra- and inter-tier) scheduler has algorithmic complexity of $O(1)$. This result does not contradict the findings of [13] which suggest that the $O(\log n)$ time complexity is fundamental to achieving good delay bounds. The analysis in [13] assumes that flow weights and packet sizes can take arbitrary values, whereas the result of Lemma 4.3 only holds under the specific assumptions of

fixed flow weights and packet lengths.

V. Intra-Tier Scheduler: Variable-Packet Case

We now remove the assumption we made in the previous section that all packets have a fixed size. As in the previous section, we consider the problem of scheduling flows within a given service tier, therefore we assume that all flows are assigned the same weight ϕ. In a network with variable-size packets, the statement of Lemma 4.1 is no longer true, since the second term in the right-hand side of (3) is not constant. Hence, in such a network, fair queueing schedulers in general require some form of packet sorting.

In the Internet, however, it is well known that certain packet sizes dominate [10], [12]. Specifically, the study in [12] found that packets of one of three common sizes make up more than 90% of all Internet traffic; the three common packet sizes identified in the study were 40, 576, and 1500 bytes, corresponding to TCP acknowledgments, the default IP datagram size, and Ethernet frames, respectively. A more recent study [10] shows that (1) Internet traffic is mostly bimodal at 40 and 1500 bytes, (2) there is a shift away from 576 bytes due to the proliferation of Ethernet, and (3) a new mode is forming around 1300 bytes which the authors theorize is due to widespread use of VPNs. Similar studies, which can be found on CAIDA's web site (http://www.caida.org), confirm that the length of the vast majority of Internet packets takes one of a small number of constant values. We can exploit these facts regarding the Internet packet length distribution to modify the intra-tier TSFQ scheduler we presented in the previous section so that it handle Internet traffic efficiently, i.e., by performing a number of packet sorting operations that is independent of the number of flows.

Queue structure. Instead of maintaining a single FIFO queue, as is the case for fixed-size packets shown in Figure 2, the intra-tier scheduler for variable packet size networks maintains a small number k of queues. The queue structure of this scheduler is illustrated in Figure 3 for the trimodal packet length distribution reported in [12]; the queue structure can be modified in a straightforward manner to reflect any similar distribution. In this case, the scheduler maintains $k = 7$ queues. Three of the queues are dedicated to packets of a common size, i.e., 40, 576, and 1500 bytes, respectively, which define the three modes of the distribution in [12]. The other four queues are for packets of size between the common values; as seen in Figure 3, there is one queue for packets of size less than 40 bytes, one for packets of size 41-575 bytes, one for packets of size 577-1499 bytes, and one for packets of size greater than 1500 bytes).

Operation. The operation of the intra-tier scheduler is identical to the one we described in Section IV, with one difference: when a head-of-line packet moves from some flow's queue to a scheduler queue, it is inserted in the queue corresponding to its size. Since each of the p inter-tier schedulers maintains k queues, the inter-tier scheduler selects the packet to serve next as the one with the smallest virtual finish time among the pk candidate packets at the head of the pk queues. Since

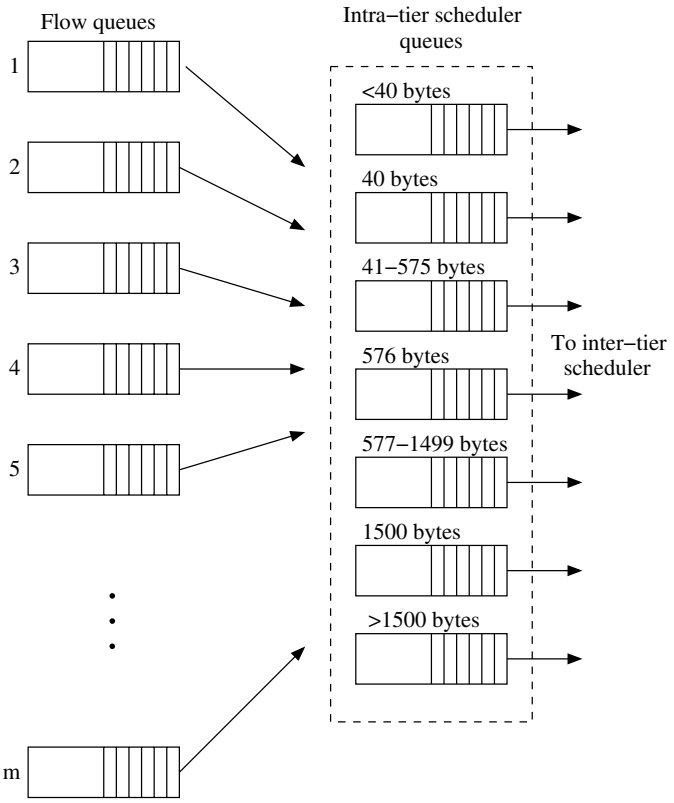

Fig. 3. Queue structure of the intra-tier scheduler for Internet packet traffic

both p and k are small integers and their values are constant for a given system, this operation takes constant time as in the fixed-size packet case.

Packet sorting operations. Note that Lemma 4.1 holds true for packets of a common size. Hence, the queues dedicated to these packets operate in a FIFO manner, and packets are simply inserted at the tail of these queues. Since packets of a common size make up more than 90% of Internet traffic [12], no sorting operations are necessary for the large majority of packets. On the other hand, queues dedicated to packets of size between the common values must be sorted appropriately at the time of a packet insertion. These sorting operations take place infrequently (less than 10% of the time), and involve relatively short queues (since less than 10% of the packets are spread over several such queues at p different service levels). Moreover, the time complexity of the sorting operations is *independent* of the number m of flows in the given service tier, and is a function only of the network load and the ratio of packets with a non-common size.

We have the following results.

Lemma 5.1: The TSFQ scheduler for variable packet sizes, consisting of p intra-tier schedulers as in Figure 3 and one inter-tier scheduler, is identical to WF^{2}Q+.

Proof. The proof of Lemmas 4.2 and 4.3 also holds in this case, hence the scheduler is equivalent to WF^{2}Q+. (Note also that, although consecutive packets of the same flow i may be inserted to different queues in Figure 3, they will always

be transmitted in order, since the second packet has a larger virtual finish time.) ■

Lemma 5.2: In the TSFQ scheduler for variable packet sizes, the virtual time function (4) can be computed in $O(1)$ time.

Proof. Each intra-tier scheduler needs to maintain a variable that represents the minimum virtual start time over all packets of the flows in the given tier. This variable is updated every time an eligible packet is inserted to any of the scheduler's queues, or whenever a packet is selected by the inter-tier scheduler for transmission; updating this variable takes constant time. Hence, the minimum operation in (4) reduces to selecting the minimum of p such variables, and can be performed in constant time in hardware. ■

Elimination of packet sorting. The operation of the intra-tier scheduler may be further simplified by eliminating packet sorting even for queues holding packets of size between the common values. Doing so may cause some packets to be served in incorrect order of virtual finish time, hence introducing a small degree of unfairness. However, the overall impact is likely to be small. Indeed, observe that packets of a non-common size represent only a small fraction of the overall traffic seen by the server, and are distributed over a number of different queues across p service tiers. Consequently, the arrival rate to each of these queues is likely to be low, especially under typical operating conditions when the load offered to the server is not too high. Now note that, since all flows within a service tier have the same weight ϕ in expression (3), the order of packets in such a queue will depend on the relative values of their virtual start time and length. Therefore, even when a small packet arrives to find larger packets in the queue (i.e., packets with a larger value for the second term in the right-hand side of (3)), the elapsed time since the previous arrival (which affects the first term of (3)) may be sufficiently large so that the queue remain sorted. This intuition is further supported by the coarse manner in which the leap forward virtual clock [11] algorithm computes timestamps, and the mechanism employed by bin sort fair queueing [4] to sort packets. The results in [4], [11] indicate that approximate sorting can be as good as exact sorting; moreover, in the case of our TSFQ scheduler, approximate sorting is limited to a small fraction of all packets. To verify this intuition, we have simulated the operation of the two variants of the TSFQ scheduler (with and without sorting the packets of a non-common size) using the *ns-2* simulator. We have indeed found that the effect of eliminating the sorting operations has negligible effect on the delay bound and fairness properties of the scheduler; due to space constraints, the results of the simulations are omitted.

We emphasize that the queue structure shown in Figure 3 is for illustration purposes only and is simply meant to convey the idea underlying the structure of the scheduler for Internet packet traffic; we do not imply that routers have to be configured in exactly this manner. Network operators may configure this queue structure to reflect the specific packet distribution observed in their networks, and update it over time as traffic conditions evolve. Similarly, they may optimize the number of service tiers and the flow weights associated with them (e.g., with the techniques developed by our research group [1], [6]) by taking into account the prevailing user demands. Therefore, our framework of fair queueing schedulers for tiered-service networks is quite flexible. Network providers may adapt the specific elements of the framework to differentiate their offerings, and to provide users with a menu of customized services.

VI. Concluding Remarks

We have proposed a new tiered service fair queueing TSFQ scheduler. Our work was motivated by two key observations: that providers typically offer a small number of service levels, and that the Internet packet length distribution exhibits a small number of prominent modes. Within each tier, the schedulers employ a fixed number of queues to handle packets with few or no sorting operations. The intra-tier scheduler simply serves the packet with the smallest timestamp among a constant number of packets at the front of the intra-tier queues. The simple structure and operation of the schedulers are practically realizable and especially attractive for hardware implementation. The TSFQ scheduler is equivalent to WF^{2}Q+ with the additional property that the virtual time function can be computed in $O(1)$ time. Therefore, we believe that employing TSFQ scheduling within high-speed routers will enable network operators to enhance significantly their ability to offer and guarantee a wide range of services.

References

[1] Nikhil Baradwaj. Traffic quantization and its application to QoS routing. Master's thesis, North Carolina State University, Raleigh, NC, August 2005. (2006 Graduate School Nancy G. Pollock MS Thesis Award).

[2] J. C. R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *Proc. ACM SIGCOMM '96*, pages 143-156, Aug. 1996.

[3] J. C. R. Bennett and H. Zhang. WF^{2}Q: worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM '96*, pages 120–128, 1996.

[4] S. Cheung and C. Pencea. BSFQ: bin sort fair queueing. In *Proceedings of IEEE INFOCOM '02*, 2002.

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

[6] Laura E. Jackson. *The Directional p-Median Problem with Applications to Traffic Quantization and Multiprocessor Scheduling*. PhD thesis, North Carolina State University, Raleigh, NC, December 2003. (2004 College of Engineering Nancy G. Pollock PhD Dissertation Award).

[7] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[8] S. Ramabhadran and J. Pasquale. Stratified round robin: A low complexity packet scheduler with bandwidth fairness and bounded delay. In *Proceedings of ACM SIGCOMM '03*, pages 239–249, August 2003.

[9] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *Proceedings of ACM SIGCOMM '95*, 1995.

[10] R. Sinha, C. Papadopoulos, and J. Heidemann. Internet packet size distributions: Some observations. http://netweb.usc.edu/˜rsinha/pkt-sizes/, October 2005.

[11] S. Suri, G. Varghese, and G. Chandranmenon. Leap forward virtual clock: An $O(\log \log N)$ queueing scheme with guaranteed delays and throughput fairness. In *Proceedings of IEEE INFOCOM '97*, 1997.

[12] K. Thompson, G. J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, Nov/Dec 1997.

[13] J. Xu and R. Lipton. On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms. In *Proceedings of ACM SIGCOMM '02*, 2002.