An Efficient Network-Switch Scheduling for Real-Time Applications

Caimu Tang, *Student Member, IEEE*, Anthony T. Chronopoulos, *Senior Member, IEEE*, and Ece Yaprak, *Member, IEEE*

***Abstract*—Bursts consist of a varying number of asynchronous transfer mode cells corresponding to a datagram. Here, we generalized weighted fair queueing to a burst-based algorithm with preemption. The new algorithm enhances the performance of the switch service for real-time applications, and it preserves the quality of service guarantees. We study this algorithm theoretically and via simulations.**

***Index Terms*—Burst preemptive scheduling, fair queueing, quality of service (QoS).**

I. Introduction

We are interested in scheduling bursts (of cells) instead of individual cells for asynchronous transfer mode (ATM) networks. Previous results in cell scheduling and burst scheduling have been reported in [1], [2], [5], [6], and references therein. Here, we propose a scheduler called burst-based weighted fair queueing with preemption (PBWFQ) for real-time applications, which is an extension of a previously developed burst-based weighted fair queueing (BWFQ) [7]. Burst-based scheduling offers performance advantages over nonburst counterparts, as elaborated in [7]. To meet real-time application requirements, bursts from a low-priority application have to yield services to bursts from high-priority applications. PBWFQ introduces preemption in BWFQ (to accommodate real-time applications) without explicitly introducing priority queueing, and thus, it is integrated into the fair-queueing scheduler directly. Buffer management is critical for any burst-level fair-queueing scheduler based on finishing time instead of arrival time, due to cell aggregation. This situation can be mitigated to some degree by limiting the maximal burst size across all sessions and by admission control. Cell-based packetized generalized processor sharing (CPGPS), BWFQ, and PBWFQ are all work-conserving (PBWFQ to be shown in Section II). This implies that the overall buffer requirement of each switch remains the same under CPGPS, BWFQ, and proposed PBWFQ. This claim can be proved for all work-conserving disciplines, based on the fact that the added buffer for some of the active sessions is exactly offset by the reduced buffer on the rest of the active sessions (details are omitted due to space limitation). Since we do not include the proof here, we assume that switches have infinite buffer size. The terminology is similar to the algorithms for scheduling packets (e.g., see [2]–[4]). In particular, we follow the model and the notation in [7].

Paper approved by T. T. Lee, the Editor for Wireless Communication Theory of the IEEE Communications Society. Manuscript received June 19, 2003; revised May 31, 2004. The work of A. T. Chronopoulos was supported in part by the National Science Foundation under Grant CCR-0312323.

C. Tang is with the Computer Science Department, University of Southern California, Los Angeles, CA 90089 USA (e-mail: caimut@cs.usc.edu).

A. T. Chronopoulos is with the Computer Science Department, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: atc@cs.utsa.edu).

E. Yaprak is with the College of Engineering, Wayne State University, Detroit, MI 48202 USA (e-mail: yaprak@eng.wayne.edu).

Digital Object Identifier 10.1109/TCOMM.2005.843434

II. PBWFQ

We consider the virtual time function used in WFQ [3]. Let $V(t)$ be the virtual time function, which is defined to be zero when the server is idle. Let ϕ_i be the bandwidth weight (i.e., service rate) for the backlogged session i. Rate change (for session i) is controlled by

$$\frac{\phi_i}{\sum_{j\in B_t}\phi_j}$$

where B_t is the set of backlogged sessions at time t. We apply the virtual time function to a burst instead of a single cell. Let i and k be the session index and burst number, respectively, and let $c(i,k)$ be the burst size.

Proposed Algorithm: Starting system busy period (at physical time t_{start})
$S(i,0) = F(i,0) = 0$ for arbitrary session index i, and $V(t_{\text{start}}) = 0$.
Burst arrival

1) Burst start

$$c(i,k) = 0 \quad \text{and} \quad S(i,k) = \max\{F(i,k-1), V(a(i,k))\}.$$

2) For each new cell arrival, the burst size is incremented

$$c(i,k) = c(i,k) + 1.$$

3) End of a burst

$$F(i,k) = S(i,k) + c(i,k)\frac{\tau_0}{\phi_i}$$

where τ_0 is the cell slot.

Burst departure **Simple Scheme (BWFQ)**: For a given session i, the scheduler serves the bursts according to the ascending order of the $F(i,k)$ backlogged in the server so far. The results for the simple scheme have been presented in [7].

Burst departure **Preemptive Scheme (PBWFQ)**: The scheduler chooses the burst with the smallest $F(i,k)$ among the backlogged sessions in the server. If a burst with a smaller virtual finish time arrives, then the scheduler will preempt the currently served burst at the closest boundary of a cell, and it will schedule this burst prior to the session being served.

Remark: In the PBWFQ scheme, starvation will not happen because the virtual finish time of a burst is based on dynamic priority. The preempted burst will definitely be scheduled at time

$$t_{\text{current}} + C(i,k)\frac{\tau_0}{\phi_i}$$

where (i,k) is the preempted burst from session i. Therefore, the session's throughput is guaranteed.

It is well known that the system busy periods of two work-conserving schedulers are identical, because the virtual time function can be reset to zero at the end of a system busy period. Therefore, we only need to consider the behavior of one system busy period in order to study the scheduling of the system.

Claim 1: PBWFQ is a work-conserving scheme.

Proof: Let set B_t contain the session indexes that are in service at time t in a BWFQ server. Due to possible preemption, for a given t, the cardinality of B_t does not change under PBWFQ, because any session being preempted is replaced by a session which is not in B_t. Since BWFQ is work-conserving [7], it follows that PBWFQ is also work-conserving. □

We next cite two theorems on quality of serive (QoS) of the simple BWFQ proved in [7].

Theorem 1: For all bursts (i,j) in the simple BWFQ, we have

$$\hat{f}(i,j) - f(i,j) \leq C_{\max}\tau_0 \tag{1}$$

where $C_{\max}$ is the maximal burst size to the server.

Theorem 2: For any time τ and session i, let $NS_i(\tau,t)$ and $\widehat{NS}_i(\tau,t)$ be the number of cells of session i served under CPGPS and BWFQ, in the interval $[\tau,t]$, respectively. Then we have $NS_i(0,\tau) - \widehat{NS}_i(0,\tau) \leq C_{\max}$.

Since, for a PBWFQ server burst, when preempted can be treated as two separate bursts, the following theorem can be derived similarly to *Theorem 1*. Here, we sketch the proof, and a detailed proof is given in [8].

Theorem 3: If the scheme is preemptive, then for all bursts (i,j), we have

$$\hat{f}(i,j) - f(i,j) \leq C_{\max}\tau_0 \tag{2}$$

and the burst service order is the same in both CPGPS and PBWFQ servers.

Proof: We first show that an equivalent simple-BWFQ priority queue system can be constructed for the preemptive queue system by splitting preempted bursts into two parts, with the first part being scheduled and the second part being preempted. The proof of the first part of the theorem is via induction on the preemption instances. The base step has only one splitting. The induction step can have more than one splitting, and it can transformed to the base-step case. The first part follows by applying *Theorem 2*.

Given any two bursts $(n,i),(m,j)$. It follows that $f_{\text{CPGPS}}(n,i) \geq f_{\text{CPGPS}}(m,j)$ if $\hat{f}_{\text{PBWFQ}}(n,i) \geq \hat{f}_{\text{PBWFQ}}(m,j)$ because PBWFQ is an approximation to CPGPS. If $f_{\text{CPGPS}}(n,i) \geq f_{\text{CPGPS}}(m,j)$, there are two cases: Case 1 $a(n,i) < a(m,j)$ and Case 2 $a(n,i) \geq a(m,j)$. If $\hat{f}_{\text{PBWFQ}}(n,i) \geq \hat{f}_{\text{PBWFQ}}(m,j)$ under Case 2, no preemption could occur and the second part holds. Inequality $\hat{f}_{\text{PBWFQ}}(n,i) < \hat{f}_{\text{PBWFQ}}(m,j)$ (under Case 1) is not possible by noticing the fact that $f_{\text{CPGPS}}(n,i) \geq f_{\text{CPGPS}}(m,j)$ and $a(m,j) \leq f_{\text{CPGPS}}(m,j)$. Finally, by noticing the fact that the last cell from burst (m,j) finishes before the services of cells in burst (n,i), it can be shown (using a proof by the process of elimination) that the preemption cases, $\hat{f}_{\text{PBWFQ}}(n,i) \geq \hat{f}_{\text{PBWFQ}}(m,j)$ under Case 1, and $\hat{f}_{\text{PBWFQ}}(n,i) < \hat{f}_{\text{PBWFQ}}(m,j)$ under Case 2, yield a situation with a burst split from burst (m,j) and burst (n,i), respectively, having a starting time greater than its own finishing time. □

From this theorem, we not only know that the discrepancy between CPGPS and PBWFQ is bounded by the maximum length of the bursts, but also that the serving order is preserved. Obviously, this is a better approximation than is in the BWFQ case.

Theorem 4: Assume the scheme is preemptive, and assume burst A [of size $c(A)$] from session i arrives at time $a(A)$ with size $c(A)$. If A finishes at $f_{\text{CPGPS}}(A)$ under CPGPS, and at time $f_{\text{PBWFQ}}(A)$ under PBWFQ, then

$$(f_{\text{CPGPS}}(A) - f_{\text{PBWFQ}}(A)) \leq \left(\frac{c(A)}{\phi_i} - \triangle\right)\tau_0$$

where $\triangle$ is the sum of lengths of bursts, which arrive after the arrival of A and depart before the departure of A under CPGPS (i.e., all the bursts which preempt A under PBWFQ), and ϕ_i is the bandwidth portion for session i.

Proof: Since PBWFQ is work-conserving and preemptive

$$f_{\text{PBWFQ}}(A) \geq a(A) + (\triangle + c(A))\tau_0. \tag{3}$$

Let set $P = \{p_i : i = 1,\ldots,m$, with bandwidth proportion $\phi_i\}$ be the burst set which arrives before arrival of burst A and departs after arrival of A, and let set $Q = \{q_i : i = 1,\ldots,n$, with bandwidth proportion $\hat{\phi}_i\}$ be the burst set which arrives after the arrival of A and departs after departure of A. We notice that the $P \cap Q = \emptyset$ and the bursts in set P are preempted bursts, and the bursts in set Q cannott preempt burst A. Since the interval

$$\left[\frac{1}{\tau_0}, \frac{\phi_i}{\phi_i + \sum_{j=1}^{m}\phi_j}\right]$$

is the rate for burst A during time interval $[a(q_1), a(A)]$, and by the definition of CPGPS, the following holds:

$$\begin{aligned} c(A) = NS_i(a(A), f_{\text{CPGPS}(A)}) &\geq \frac{1}{\tau_0}\frac{\phi_i(a(q_1) - a(A))}{\phi_i + \sum_{j=1}^{m}\phi_j} \\ &+ \frac{1}{\tau_0}\frac{\phi_i(a(q_2) - a(q_1))}{\phi_i + \hat{\phi}_1 + \sum_{j=1}^{m}\phi_j} \\ &\vdots \\ &+ \frac{1}{\tau_0}\frac{\phi_i(f_{\text{CPGPS}}(A) - a(q_n))}{\phi_i + \sum_{j=1}^{n}\hat{\phi}_j + \sum_{j=1}^{m}\phi_j}. \end{aligned} \tag{4}$$

By (4), we have

$$f_{\text{CPGPS}}(A) \leq a(q_n) + \frac{(\tau_0 c(A) - \oplus)}{\ominus} \tag{5}$$

where

$$\ominus = \frac{\phi_i}{\phi_i + \sum_{j=1}^{m}\phi_j + \sum_{j=1}^{n}\hat{\phi}_j}$$

$$\oplus = \sum_{k=1}^{n}\frac{\phi_i(a(q_k) - a(q_{k-1}))}{\phi_i + \sum_{j=1}^{m}\phi_j + \sum_{j=1}^{k}\hat{\phi}_j}.$$

TABLE I
SOURCE TRAFFIC PARAMETERS

Session No.	1	2	3	4	5	6	7	8
Bandwidth Portion	0.1	0.1	0.1	0.1	0.2	0.5	0.5	1.0
Pk Size Args	8988	1581	1581	8988	1240	1240	2400	2400
Source Interarrival	200	200	200	200	100	100	100	100
Source Start Time	100.0	0.0	200.0	0.0	100.0	100.0	20.0	0.0
Source End Time	2000	1800	1700	1500	2000	2000	1800	1700

Notice that $q_0 = A$ in the above. Denote

$$\Lambda_0 = \phi_i + \sum_{j=1}^{m} \phi_j + \sum_{j=1}^{n} \hat{\phi}_j.$$

Let

$$\Phi(k) = \frac{\Lambda_0}{\phi_i + \sum_{j=1}^{m} \phi_j + \sum_{j=1}^{k} \hat{\phi}_j}.$$

Since

$$\sum_{k=1}^{n} \Phi(k) \geq \sum_{k=1}^{n} (a(q_k) - a(q_{k-1})) \geq a(q_n) - a(A) \qquad (6)$$

we have

$$f_{\mathrm{CPGPS}}(A) \leq \frac{\tau_0}{\phi_i} \Lambda_0 c(A) + a(A). \qquad (7)$$

By (3) and (7), we have

$$f_{\mathrm{CPGPS}} - f_{\mathrm{BWFQ}} \leq \left(\frac{c(A)}{\phi_i} - \triangle \right) \tau_0$$

because $\phi_i + \sum_{i=1}^{m} \phi_i + \sum_{j=1}^{n} \phi_j \leq 1$, if we use a normalized bandwidth portion. However, if we do not use a normalized bandwidth portion, we need to replace the ϕ_i by the bandwidth percentage available to the session i. □

III. IMPLEMENTATION AND COMPARISONS WITH OTHER SCHEMES

At first, we discuss the difference in the virtual time functions of PBWFQ and self-clocked fair queueing (SCFQ). The virtual time function used in SCFQ [1], [2] is very attractive, due to its simplicity. Since PBWFQ is work-conserving, and virtual time computation is done at the level of cell aggregates (with much less frequency), and per-cell processing uses cell counting which can be done fast, a finish-time-based virtual function performs also very well, because the buffer requirement is the same under both PBWFQ and SCFQ disciplines. *Example*: Assume that CPGPS needs to compute the virtual time 1 000 000 (1M) per second on one output link with a bandwidth of 53 Mb/s. Since BWFQ or PBWFQ are working on burst level, for example, on average, the burst size may be 50 cells. The total virtual function invocations 1) under BWFQ is 1 M/50 = 20 K, and 2) under PBWFQ it might be greater than 20 K; however, it will be much less than 1 M. A detailed study on the efficiency of BWFQ is presented in [7]. Our simulation model is based on a two-level ATM network environment consisting of backbone switches and access switches. The simulation is done in

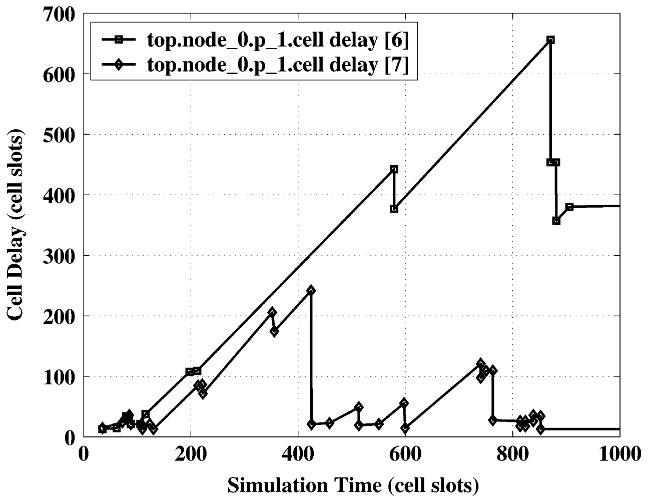

Fig. 1. Cell delay under PBWFQ.

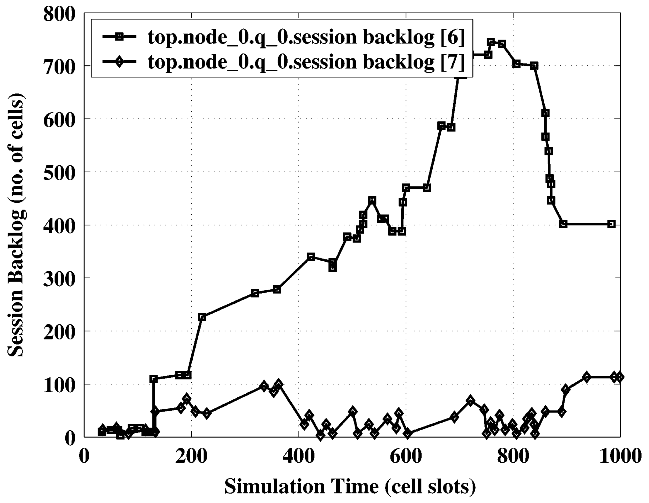

Fig. 2. Session backlog under PBWFQ.

an OPNET simulation environment. The traffic parameters are given in Table I.

We compare PBWFQ with BWFQ and CPGPS, where CPGPS has the finest granularity in terms of approximation to a GPS server. The performance comparison results between BWFQ and CPGPS is presented in [7]. For lack of space, we only present the results on cell backlog/delay and session throughput with one scenario at a subset of the nodes only. The cell delay, backlog, and session throughput under PBWFQ are shown in Figs. 1–3. We refer to [7, Figs. 13, 14, and 17 (Figs. 14, 16, and 18)] for the corresponding cell backlog, ses-

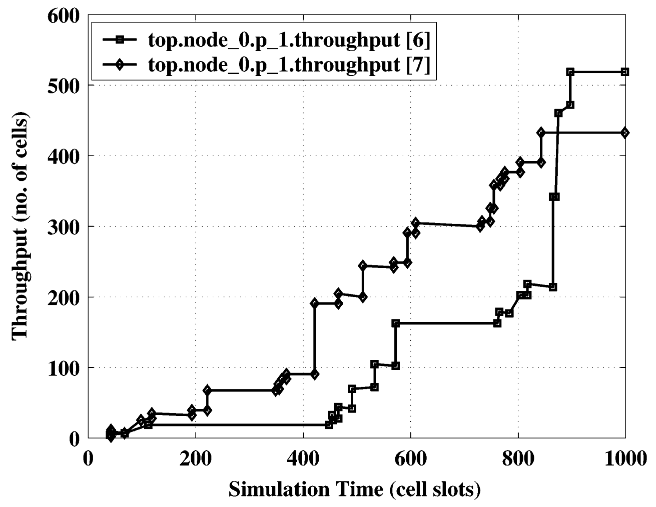

Fig. 3. Session throughput under PBWFQ.

sion throughput, and delay of BWFQ (CPGPS), respectively. These comparisons reveal that the backlog variation in PBWFQ servers is relatively larger than those in CPGPS and BWFQ servers. It is reasonable because the cells from the same session are scheduled at closer times in bursts in a PBWFQ server. As for QoS indexes, the PBWFQ server outperforms both CPGPS and BWFQ servers.

IV. Conclusion

In this letter, a novel scheduling scheme called PBWFQ is proposed. Since it allows high-priority sessions to preempt low-priority sessions, it can support real-time applications to meet the latency requirements. Analytic results, as well as experimental results, show that it can provide service guarantees to both high-priority and low-priority sessions.

References

[1] S. J. Golestani, "Network delay analysis of a class of fair queueing algorithms," *IEEE Trans. Sel. Areas Commun.*, vol. 13, pp. 1057–1070, Aug. 1995.

[2] P. Goyal and H. M. Vin, "Generalized guaranteed rate scheduling algorithms: A framework," *IEEE/ACM Trans. Netw.*, vol. 4, pp. 561–571, Aug. 1997.

[3] 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. Netw.*, vol. 1, pp. 344–357, Jun. 1993.

[4] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Trans. Netw.*, vol. 6, pp. 175–185, Apr. 1998.

[5] G. Xie and S. Lam, "Real-time block transfer under a link-sharing hierarchy," *IEEE/ACM Trans. Netw.*, vol. 6, pp. 30–41, Feb. 1998.

[6] T. Mizuike, Y. Ito, D. J. Kennedy, and L. N. Nguyen, "Burst scheduling algorithms for SS/TDMA systems," *IEEE Trans. Commun.*, vol. 39, pp. 533–539, Apr. 1991.

[7] A. T. Chronopoulos, C. Tang, and E. Yaprak, "An efficient ATM network switch scheduling," *IEEE Trans. Broadcast.*, vol. 49, pp. 278–292, Sep. 2003.

[8] C. Tang, A. T. Chronopoulos, and E. Yaprak, "A cell burst based scheduling for ATM networking: Part I," in *Proc. 3rd IEEE Symp. Computers, Commun.*, Jun.–Jul. 1998, pp. 455–461.