An Efficient Per-VC Accounting-Based Buffer Management Scheme for GFR Services in TCP/IP Networks

Po-Chou Lin and Chung-Ju Chang, *Senior Member, IEEE*

***Abstract*— An efficient per-virtual connection (per-VC) accounting-based scheme with steepest decent weight updating (PASD) algorithm is proposed for buffer management in TCP/IP networks supporting guaranteed frame rate (GFR) services. It can achieve good fairness, high utilization, and low delay. Also it can be implemented by a genuine FIFO architecture and thus easily support up to 10 Gbps STM-64 rate.**

***Index Terms*— Guarantee frame rate, buffer management, TCP/IP.**

I. Introduction

Guaranteed frame rate (GFR) services intend to provide committed access rate (CAR) or minimum cell rate (MCR) guarantee at the frame level and receive a fair share of any unused system capacity for TCP/IP networks over MPLS or ATM. A simple and efficient buffer management scheme is necessary toward successful development of GFR services.

Methods proposed to support GFR services can be classified into three types. The first one is the simplest, which relies only on the tagging of frames performed by a frame-based generic cell rate algorithm (F-GCRA) policing at the ingress node of the network. However, the fair share of available bandwidth cannot be achieved. The second one belongs to the *per-virtual connection (per-VC) accounting* class, which maintains individual counters for every VC. This type is the most suitable for single FIFO implementation. Several schemes have been proposed, such as the differential fair buffer allocation (DFBA) [1] and the packet-discard push-out (PDPO) [2]. The third one belongs to the *per-VC queueing* class, such as the dynamic threshold - early packet discard [3] and the selective weighted fair allocation (SWFA) [4], which maintains multiple FIFO queues and relies on a weighted fair queueing (WFQ)-like scheduler to provide MCR guarantee, fair share of available bandwidth, and QoS differentiation. However, the WFQ scheduler is difficult to implement at high transmission speed due to its high complexity. The complexity of WFQ arises from two main sources: updating the virtual clock and sorting of packet tags to schedule the new transmission. The per-VC queueing class is much more complicated than the per-VC accounting class.

Manuscript received October 20, 2004 The associate editor coordinating the review of this letter and approving it for publication was Prof. Jinwoo Choe. This work was supported by the National Science Council, Taiwan, under contract No. 91-2219-E-009-034, and by the Lee and MTI Center for Networking Research at the National Chiao Tung University, Taiwan, under Grant No. Q.528.

Po-Chou Lin is with the Wireless Communication Technology Lab., Telecommunication Laboratories, Chunghwa Telecom Co., Ltd., Taoyuan 326, Taiwan (e-mail: pochou@cht.com.tw).

Chung-Ju Chang is with the Department of Communication Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: cjchang@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/LCOMM.2005.06009.

II. The PASD Algorithm

The letter presents an efficient **p**er-VC **a**ccounting-based scheme with **s**teepest **d**ecent weight updating (PASD) algorithm for buffer management of GFR services. The PASD algorithm is mainly constituted by two schemes: the prepaid early packet discard (PEPD) and the nonlinear random early detection (RED) with steepest descent weight updating (SDWU). The former is used to guarantee frame level throughput; it decides whether a frame can be accepted, based on the FIFO residual queue length. The latter, with a probability generated by the embedded SDWU, decides to discard a frame of the corresponding VC due to fair share.

The PASD algorithm originally proposes the SDWU method, embedded in the RED scheme [5], to determine a nonlinear dropping probability for link utilization enhancement and fairness improvement. Let X_i be the cell number of i-th $VC(VC_i)$ in the FIFO buffer, N be the total number of VCs, B be the FIFO buffer capacity, and $X = \sum_{i=1}^{N} X_i$. Denote MCR_i to be the MCR requested by VC_i and F_i to be the fair index of VC_i, where F_i is initially set to be $(MCR_i / \sum MCR_i)$. The nonlinear dropping probability for tagged cells of VC_i is given by

$$P = \begin{cases} \min\{[(\frac{X_i}{X \cdot F_i} - 1) \cdot \frac{X}{B}], 1\} & \text{if } X_i > (X \cdot F_i), \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where the $(X_i/(X \cdot F_i))$ term indicates occupancy of VC_i fair share, and the (X/B) reflects total buffer occupancy.

The TCP window management algorithm fast decreases the window size to half if congested or increases the window size step by step otherwise. It is easier for low MCR VCs to recover to their corresponding fair share bandwidth than for high MCR VCs. Therefore, low MCR VCs tend to overutilize the available bandwidth and then the unfairness exists. The SDWU method innovatively adopts the basic idea of neural networks to adaptively adjust the fair index F_i of VC_i in (1). Define ρ_i as the actual utilization for VC_i in a specified time interval,which can be measured from the output link; denote $f_i = \frac{MCR_i}{\sum MCR_i}$ to be the fair share of the VC_i throughput in the ideal case; and let η be a weighting factor to adjust F_i stepwise. The adjustment of F_i in (1), based on the

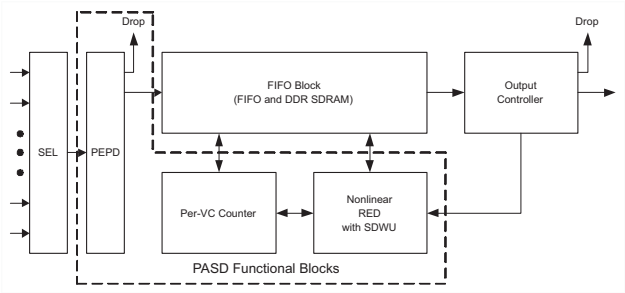

Fig. 1. Functional block diagram of the PASD algorithm.

difference between the VC_i fair share f_i and the measured VC_i utilization ρ_i, is given by

For every fixed update interval /*SDWU Block */
Repeat for all $1 \le i \le N$
$\Delta F_i = \eta\,(f_i - \rho_i)\,;\ F_i = F_i + \Delta F_i\,;$ End. ■

Notice that the resulted ρ_i will not approach f_i if F_i is set linearly proportional to MCR_i, due to the nonlinear TCP sliding window management algorithm. However, the SDWU method can make ρ_i approach its fairness index f_i by adaptively updating F_i. If the measured ρ_i is less than its fair share f_i, the SDWU method increases F_i by $\eta \cdot (f_i - \rho_i)$. For the larger F_i, the SDWU method reserves more FIFO space and thus yields a lower value of nonlinear dropping probability P in (1). A lower value of P will result in a higher ρ_i, which makes ρ_i closer to its fair share f_i. The proposed PASD can dynamically reserve the FIFO space based on the measured ρ_i; unlike traditional algorithms, such as DFBA, which fixedly reserves the FIFO space. The updating process repeats for every fixed interval. If the nonlinear RED scheme with SDWU detects an over-utilization of VC_i, a drop-from-front strategy [6] is adopted to drop a whole VC_i frame from head of the buffer. The drop-from-front strategy can effectively reduce the round-trip delay and resolve the congestion fast. Also, it is the best suitable for FIFO implementation.

The functional block diagram of the PASD algorithm is shown in Fig. 1. The *SEL* block selects one of incoming cells from input modules of a switch to the *PEPD* block of a specific output module. The PEPD block decides to accept or reject a frame, based on the FIFO residual queue length, and pushes cells to the *FIFO* block if accepted. Acceptance of the first cell of a frame by the PEPD block also triggers the *nonlinear RED with SDWU* block to justify its VC's fair share. The *per-VC counter* block keeps all the counters for active VCs. Assume the maximum frame size is M (cells). The PASD algorithm sets a counter B_t to track the FIFO residual queue length used by PEPD and a counter L_i to record the incoming frame length for VC_i. Define $A1_i$ to mark acceptance or rejection of the present VC_i frame for FIFO input port and $A2_i$ to mark acceptance, retagging, or rejection of VC_i cells for FIFO output port. Let U_i count the number of unrealized retagging VC_i frames by the PEPD and V_i count the number of discarded VC_i frames by the nonlinear RED with SDWU. As the first cell of a VC_i frame arrives at the FIFO, if $B_t > 0$, the cell is accepted with unmarking $A1_i = 0$ and the available buffer space is prepaid by setting B_t to be $B_t - M$; otherwise, the PEPD block marks $A1_i = 1$. Based on $A1_i$, the PEPD block decides to accept or discard the rest cells of a VC_i frame. If the discarded frame is untagged (cell loss priority (CLP)=0), the updated counter U_i is increased. The *output controller* block will compensate an untagged frame loss by retagging a tagged (CLP=1) frame to an untagged frame and decreasing the U_i counter. When the switch receives the last cell of a frame, it gets the exact frame's length L_i and updates B_t to be $B_t + M - L_i$. The counter L_i is reset to zero for next incoming frames. Also, whenever a cell is served out from the FIFO block, B_t is increased. The detailed implementation of the PASD algorithm by *Pseudo Code* is shown below.

[*Pseudo Code of PASD*]

B_t=B; $X_i = L_i = U_i = V_i = 0$, $F_i = \frac{MCR_i}{\sum_{i=1}^{N} MCR_i}$;
for $1 \le i \le N$; /**Initialization* */
For every cell time do
/**FIFO Input Port* */
Get VC index i from the VPI/VCI of an incoming cell;
If (First Cell)
{ If ($B_t \le 0$) /**Check for PEPD* */
$A1_i = 1$; If (Untagged cell) U_i++; /**Add Untagged Counter* */
Else
{ $A1_i = 0; B_t = B_t - M;$
If (Tagged cell) /**Check for RED with SDWU* */
If (($((X_i/(X * F_i)) - 1) * (X/B)$) $> Random(0,1)$)
V_i++; } } /**Add Discard Counter* */
Switch $A1_i$:
{ Case 0: Accept Cell; L_i++; X_i++;
If (Last Cell) { $B_t = B_t + (M - L_i); L_i = 0;$ } break;
Case 1: Discard Cell; break; }
/**FIFO Output Port* */
Label: While ((FIFO not Empty))
{ If (First Cell)
{ $A2_i = 0;$
If (Low Priority)
If ($V_i > 0$) {$A2_i = 2$; V_i−−; } /**Mark Discard Frame* */
Else if ($U_i > 0$) {$A2_i = 1$; U_i−−; }
/**Mark Retagged Frame* */ }
Switch $A2_i$:
{ Case 0: Transmit cell; B_t++; X_i−−;break;
Case 1: Retagged to High Priority; Transmit cell;
B_t++; X_i−−;break;
Case 2: Discard cell; goto Label; break; } }; End.

/**SDWU Block* */
For every fixed update interval
Repeat for all $1 \le i \le N$
$\Delta F_i = \eta\,(f_i - \rho_i)$; $F_i = F_i + \Delta F_i$; End. ■

The PASD algorithm can support 10 Gbps STM-64 applications by using FIFOs for the VC header processing and double data rate (DDR) SDRAM for the payload precessing in the FIFO block. As noted, an FIFO chip can easily operate at 166 MHz, and a DDR SDRAM module can operate at 400

TABLE I
PERFORMANCE COMPARISONS OF PASD, PDPO, SWFA AND DFBA

	Ideal	PASD	PDPO	SWFA	DFBA
VC_1 MCR	0.0667	0.0796	0.0798	0.1537	0.1788
VC_2 MCR	0.1333	0.1484	0.1436	0.1745	0.1821
VC_3 MCR	0.2000	0.2032	0.2044	0.2058	0.1959
VC_4 MCR	0.2666	0.2527	0.2589	0.2164	0.2054
VC_5 MCR	0.3333	0.3122	0.3130	0.2309	0.2168
UI	1	0.9961	0.9997	0.9813	0.9790
FI	1	0.9911	0.9924	0.8193	0.7535
D		432	987	231	64
σ		75	96	285	121

MHz clock rate. The maximum throughput of the 64-bit DDR SDRAM is then 25.6 Gbps; this easily makes the architecture support STM-64 applications.

III. SIMULATION RESULTS AND DISCUSSIONS

The simulated network configuration used in [3],[4] is adopted here, where there are two backbone switches connected via STM-1. Each backbone switch carries 5 GFR VCs, and each GFR VC comes from an edge switch. Every VC handles 10 TCP connections and all connections are greedy sources. The TCP version used is New-Reno and the retransmission mode is Go-Back-N. The propagation delays are assumed to be 12 cell time (about 10 km) between adjacent edge and backbone switches and 1200 cell time (about 1000 km) between the two backbone switches. The MCR values for VC1 to VC5 are 5, 10, 15, 20, and 25 Mbps, respectively, giving a total of MCR allocations 50% of total GFR capacity. The TCP frame length is fixed at 12 cells. The FIFO buffer space is set to be 1000 cells. The L(low) and H(high) thresholds of both DFBA and SWFA are set to be 0.1 and 0.9 of the buffer capacity, respectively.

Define the system utilization index $UI = \frac{\sum_{i=1}^{N} \rho_i}{\sum_{i=1}^{N} f_i}$ and the system fairness index $FI = \frac{(\sum_{i=1}^{N} \rho_i/f_i)^2}{N \cdot \sum_{i=1}^{N} (\rho_i/f_i)^2}$. Table I shows the allocated MCRs of VC_1 to VC_5, the system utilization index UI, the system fairness index FI, and the average cell delay D with its standard deviation σ, of PASD, PDPO, SWFA, and DFBA under *per-VC accounting* constraint. Since all TCP connections are greedy sources, the actually generated traffic is limited by the TCP sliding window algorithm, which is controlled by the buffer management scheme. It can be found that DFBA and SWFA show comparable utilization but poor fairness. The unfairness is due to the reason that the PASD can adaptively adjust F_i and dynamically reserve the FIFO space, while both DFBA and SWFA statically set F_i linearly proportional to MCR_i and reserve a fixed FIFO space. Thus the fastly decreasing and slowly increasing TCP window management algorithm results in low MCR VCs over-utilizing the fair share bandwidth and high MCR VCs under-utilizing the fair share bandwidth. Besides, the SWFA belongs to the *per-VC queueing* class, which relies on a complicated WFQ-like scheduler to provide fairness. The complexity of WFQ arises from two operations of updating the virtual clock and sorting of packet tags to schedule the new transmission. Although DFBA and SWFA have smaller delay, it is because they attain lower system utilization. Their delay would increase exponentially if the system utilization approaches one.

TABLE II
COMPARISONS OF PASD AND PDPO

PASD	PDPO
Maximum UI & FI	Maximum UI & FI
Genuine FIFO Architecture	Special Pushout Architecture
First Come First Serve	Complex Three Pushout Phases
10Gbps STM-64 rate	Within 2.5Gbps STM-16 rate
Use half buffer space	Use almost all buffer space

Moreover, PASD and PDPO can achieve maximum UI and FI. Simulations of variable TCP sizes on different network architectures show similar results. However, there are two shortages in PDPO. First, implementation cost of the push-out process is too high. Commercial FIFOs cannot be applied to the memory architecture of PDPO, since PDPO needs to pushout frame from any place of an FIFO. To pushout a VC_i frame from an FIFO, three phases are involved. (i) The VC_i connection, which uses the most extra bandwidth than its fair share, should be correctly chosen. (ii) All of the cell addresses of the last VC_i frame should be determined. Note that a frame of cells may be interleaving; so all of the cell addresses of the corresponding VC_i frame should all be memorized. (iii) The last phase, content of memorized cell addresses should be pushed out through shift operations. The PDPO uses four major function components, a packet-discard controller, a cell dispatcher, a stack controller, and a push-out controller, to complete the operations described above. Due to the complex architecture, the operation speed of the PDPO scheme is limited within 2.5 Gbps STM-16 line rate. Second, the PDPO tends to use up all buffer space, hence increasing the mean delay and delay variation. Besides, the PDPO algorithm in the average occupies 98.7% of the FIFO space, while the PASD algorithm uses only 43.2%. This will result in a higher cell loss probability for PDPO, when new setup VCs or bursty traffic arrives. Table II summarizes comparisons of PASD and PDPO. The proposed PASD algorithm can be applied to TCP/IP networks supporting GFR services and implemented by a genuine FIFO-based architecture, while the PDPO scheme [2] is in a complicated configuration. The PASD algorithm can easily facilitate hardware implementation; it is more compelling: simpler and more efficient.

REFERENCES

[1] R. Goyal, R. Jain, S. Fahmy, and B. Vandalore, "Buffer management for the GFR services," *ATM Forum* 98-0405, 1998.

[2] C. T. Chan, S. C. Hu, P. C. Wang, and Y. C. Chen, "A FIFO-based buffer management approach for the ATM GFR services," *IEEE Commun. Lett.*, vol. 4, pp. 205-207, June 2000.

[3] D. Wu and H. J. Chao, "Buffer management and scheduling for TCP/IP over ATM-GFR," in *Proc. IEEE GLOBECOM'98*, pp. 519-524, Nov. 1998.

[4] W. K. Lai and C. C. Liu, "SWFA: a new buffer management mechanism for TCP over ATM-GFR," *IEEE Trans. Commun.*, vol. 51, pp. 356-358, Mar. 2003.

[5] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397-413, Aug. 1993.

[6] T. V. Lakshman, A. Neidhardt, and T. J. Ott, "The drop from front strategy in TCP and in TCP over ATM," in *Proc. IEEE INFOCOM'96*, pp. 1242-1250, Mar. 1996.