ELSEVIER

Computer Networks 31 (1999) 1927–1933

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Dynamic buffer allocation in an ATM switch

Ece Yaprak [a,1], Anthony Theodore Chronopoulos [b,*], Kleanthis Psarris [b,2], Yi Xiao [c,3]

[a] *Wayne State University, Detroit, MI, USA*
[b] *University of Texas at San Antonio, San Antonio, TX, USA*
[c] *Microcrafts Inc., Redmond, WA 98052, USA*

Abstract

Efficient and fair use of buffer space in an Asynchronous Transfer Mode (ATM) switch is essential to gain high throughput and low cell loss performance from the network. In this paper a shared buffer architecture associated with threshold-based virtual partition among output ports is proposed. Thresholds are updated based on traffic characteristics on each outgoing link, so as to adapt to traffic loads. The system behavior under varying traffic patterns is investigated via simulation; cell loss rate is the quality of service (QoS) measure used in this study. Our study shows that the threshold based dynamic buffer allocation scheme ensures a fair share of the buffer space even under bursty loading conditions. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: ATM switch; Dynamic threshold updating; Cell loss; Quality of service

1. Introduction

The recent emergence of ATM technology makes available a unique driving technology for high-speed communication platforms, with line speeds that will scale to interfaces in the gigabit range. An ATM-based switch can intermix all classes of traffic (voice, video, data), with different latency and delay constraints, on the same transmission and switching fabric, with a guaranteed QoS. It thus provides an ideal platform for supporting multimedia-based services.

To achieve such a QoS, a *traffic contract* is secured between the host and the network. At an initial set-up, the host informs the network of its anticipated traffic characteristics, such as a peak and an average data rate, a maximum delay, and a cell loss probability in each direction of the requested connection [5]. The network then takes these traffic parameters and available resources into account in allocating its resources to satisfy the host's requests without adversely affecting existing connections [2–4,8,9]. Current ATM networks even with traffic shaping, do not automatically meet the performance requirements set forth in the traffic contract. A potential problem in the ATM networks may be the contention in the switch caused by either long-term or short-term congestion. The first is caused by more incoming traffic than the network can handle, and the second is caused by burstiness in the traffic. Congestion control contributes greatly to a stable and efficient operation of an ATM network.

There are several different mechanisms for avoiding congestion such as admission control, rate-based

[*] Corresponding author. E-mail: atc@cs.utsa.edu
[1] E-mail: yaprak@et1.eng.wayne.edu
[2] E-mail: psarris@cs.utsa.edu
[3] E-mail: yix@microcrafts.com

control, and resource reservation. *Congestion control through adequate buffering*, is becoming particularly significant in minimizing the probability of cell loss and cell delay. Unlike deterministic multiplexing, whereby each connection is allocated its peak bandwidth, statistical multiplexing allows several connections that may be very bursty at times to share the same link based on their traffic characteristics, so that statistically they will not all burst at the same time. Congestion still occurs, however, when multiple cells are blasting away at the peak rate simultaneously through different incoming links and attempt to reach the same outgoing link at the same (cell slot) time. In this case, only one cell is allowed to go through the network, and the others must be stored in buffers. A switch buffering strategy and the buffer size then become important, because buffers are required to secure low cell loss rate by providing a place to guard against cell loss when the switch is overloaded with *bursty traffic*. The choice of strategy or size can have a dramatic effect on the performance of the switch. If cell loss is experienced due to overflowing buffers, then overall system performance is degraded.

This paper proposes the design of a dynamic buffer allocation through virtual partitioning. In Section 2, some major issues about designing dynamic buffer allocating ATM switch are addressed; in Section 3, the dynamically buffer management scheme is presented; simulation results are shown in Section 4; finally, we draw our conclusions on this study in Section 5.

2. Design objectives of a dynamic buffer allocation

Under bursty traffic, statistical multiplexing of ATM cells will lead to severe cell loss. Large buffers are costly, create excessive cell delay and increase switch cost. Thus an appropriate number of buffers must be allocated in a switch and an optimal utilization of them must be made. The goal is to accommodate more incoming traffic cells from various sources and smooth out the burst arrival rate while limiting the overhead of the switch to a predetermined size. A feasible solution is a dynamic buffering strategy, which applies a *threshold-based sharing policy* to the buffers and a *priority-based cell push-out policy* to either buffered or incoming cells.

Choice of a threshold: A number of buffer allocation schemes have been investigated. (1) *Complete sharing* accommodates all cells if the storage space is not full. (2) *Complete partitioning* divides permanently the entire space to all output ports (this is equivalent to output queuing). (3) *Partial sharing* or *partial partitioning* reserves a certain number of buffers and the rest are partitioned among the output ports [1]. When traffic is bursty and the aggregated cell arrival rate exceeds the link transmission rate then the shared buffers in the switch are quickly filled, which makes cell loss unavoidable. This raises the question of how to choose the cells to be dropped, while maintaining fairness of cell loss among all output ports, when the switch buffer is full. Under *symmetric traffic* (equal traffic on all ports), it has been proved that the optimal policy in shared memory switches with two output ports is the ‘‘push-out type with threshold’’ (POT) [1]. In other words, whenever the buffer is not full, an incoming cell should be accepted. Once the buffer is full, a cell of type 1 (or of type 2) is allowed in the switch and a cell of type 2 (or of type 1) is pushed out, if the number of type 1 (or type 2) cells is below some threshold, such as T1 (or T2 for type 2), where $T1 + T2 = \text{buffer size}$. Under *asymmetric traffic*, however, for a system with N ports, the optimal policy is still an open question. Thus, we are motivated to impose a threshold on buffer allocation to logical output queues, thus preventing one queue from draining out all the space while throughput is very low due to smaller buffers on other outgoing links [4].

Intuitively, in order to adapt to different mixes of incoming traffic, the threshold should be changing dynamically, unlike static partitioning. Some kinds of traffic applications, such as video and audio, are very sensitive to delay and cell loss. Under a bursty condition, the more traffic of such types the switch can accommodate, the better the QoS will be. Yet, the statistical nature of the significant part of the traffic, its burstiness, and its variability combine to pose difficulties in distributing buffers of fixed size. Therefore, in a statistical multiplexing environment with heterogeneous traffic, before arriving at an optimal threshold, it is quite important to define *conges-*

tion detection criteria for dynamic threshold updating. The complexity and the performance of the switch mechanism play a critical role in preventive congestion control. Our algorithm detects the congestion and updates the corresponding threshold. The threshold updating processes obtain and use the following *attributes*:

- *in-use bandwidth* (i.e. the effective bandwidth of the arriving cells),
- *in-use bandwidth of distinct QoS* (e.g. real-time traffic),
- *available buffer space* on the corresponding logical queue, and
- *the latest updated cell arrival rate* destined to the corresponding output port.

The first two attributes relate to anticipated volume as well as the QoS requirement of the incoming traffic on each outgoing link in general, while the last two involve the real-time evolution state of the switch. This information gives a heuristic measure of the traffic in the near future and threshold updating decisions will be based on this prediction. The updating processes for thresholds are efficient in the sense that all four attributes can be easily retrieved, and no complicated calculations are required. Therefore, the procedure causes little latency over the network.

3. Dynamic buffer threshold updating

We designed a simulation model to investigate our proposed method using the OPNET package of Mil 3, Inc. [6]. OPNET is a very sophisticated software package for supporting simulation and performance evaluation of communication networks. An example of a typical source mode model in an ATM is shown in Fig. 1. Our focus is on the ATM switch process residing in the ATM switch module, and our goal is to explore a *heuristic buffer management control scheme* to ensure maximum throughput within a limited buffer capacity. Without loss of generality we restrict our study to a buffer shared by real-time and nonreal-time traffic. To accommodate more cells of the traffic, our solution dynamically shares the limited buffer space among all output ports while maintaining logically separated output queues in the switch. The OPNET standard ATM model component, that is the ATM switch process,

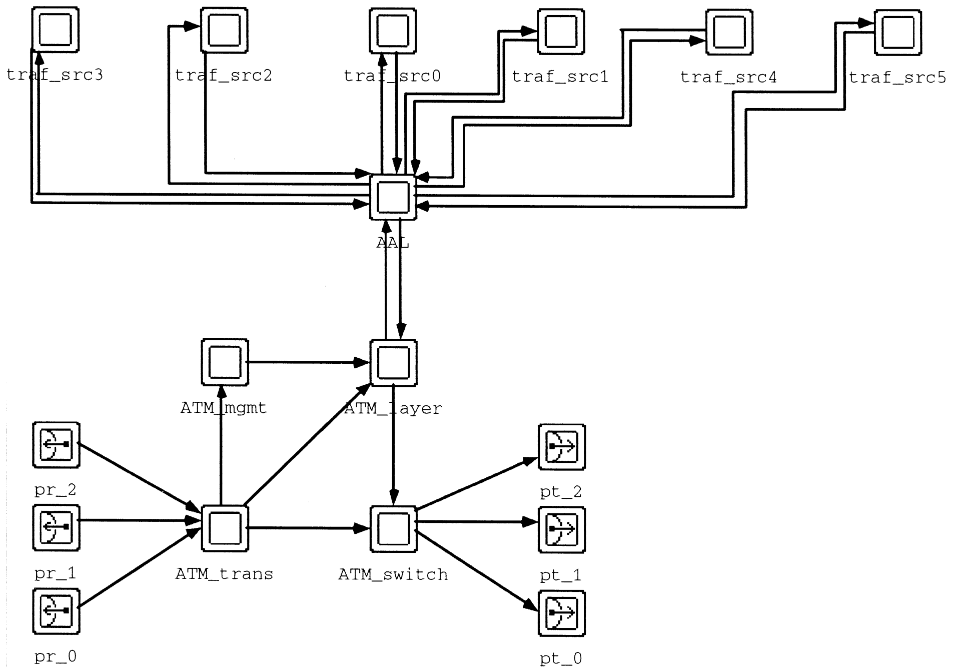

Fig. 1. A typical source node model in an ATM network.

which uses a static buffer allocating strategy, is obviously not suitable for our purposes.

The general policies of the threshold adaptation strategy are:

(i) First in first out (FIFO) policy is enforced at all times.
(ii) Real-time and nonreal-time cells have an equal chance to be transmitted over an outgoing link from the corresponding output port.
(iii) Thresholds of the logical queues for all output ports are equally assigned during the initialization phase.
(iv) Thresholds are updated every M time slots.
(v) Any excess of buffer space from any logical queue goes to the shared buffer pool.
(vi) A cell accommodation rule (CAR) is applied whenever a cell arrives at the switch.

A CAR is needed to make use of the dynamic thresholds. When the buffer pool is not full, any incoming cell should be accepted; otherwise a selective cell admission/drop mechanism is enforced. Based on assumptions (iv) and (v), our algorithm adapts to traffic and maintains fair cell loss among all output ports. It differs from the static algorithm in that cell buffering depends on thresholds of output ports, not on the fixed length of each output queue. Because any space in the buffer pool is available to cells destined to various outgoing links, there is no dedicated use of buffer space.

To measure and predict the incoming traffic the following four attributes are identified on any outgoing link, i:

1. m_{ai} = in-use bandwidth of the nonreal-time traffic, and $A_i = m_{ai}/\sum_{j=1}^{n} m_{aj}$, where n is the total number of links.
2. m_{ui} = in-use bandwidth of the real-time traffic, and $U_i = m_{ui}/\sum_{j=1}^{n} m_{uj}$, where n is the total number of links.
3. m_{bi} = the reciprocal of the available buffer space of the corresponding logical queue and $B_i = m_{bi}/\sum_{j=1}^{n} m_{bi}$. (Note that $m_{bi} = 0$ if available buffer space is less than or equal to 0.)
4. m_{li} = the updated short-term cell arrival rate destined to the corresponding output port, and $L_i = m_{li}/\sum_{j=1}^{n} m_{li}$.

For $i = 1,\ldots,n$, the weights of each traffic attribute are $W_{i1},\ldots,W_{i4}$, where $\sum_{j=1}^{4} W_{ij} = 1$. Thus, the new threshold (R_i) of logical queue i for outgoing link i is adjusted by the following formula: $R_i = S_{\text{buff}} * (X_i/\sum_{j=1}^{n} X_j)$, where S_{buff} is the size of the shared buffer space, and $X_i = A_i * W_{i1} + U_i * W_{i2} + B_i * W_{i3} + L_i * W_{i4}$.

Exceptions occur when there is no incoming traffic or no call is being set up during a certain threshold updating interval at all output ports (or over all outgoing links). The following polices are used to deal with these cases:

1. Let K_i be any one of A, U, B, and L, and let k be any one of a, u, b, or l. If $K_i = \sum_{j=1}^{n} m_{kj} = 0$, then the corresponding K_i is assigned the value 0.
2. If $\sum_{j=1}^{n} X_j = 0$, then the thresholds of the logical queues for all output ports are equally assigned.
3. If any $X_i = 0$, then the threshold R_i is assigned a default share of S_{buff}, and the rest of the buffer space, $S_{\text{buff}} = S_{\text{buff}} - R_i$, is distributed by the above formula.

If the length of the logical queue to which the cell is headed, is within its threshold, then we term it a conforming queue (Qc) (else nonconforming queue (Qnc)). The cell delay variation (CDV) is a parameter showing how much the (cell) delay exceeds the nominal interval [5]. When the entire buffer pool is full, a new cell is buffered by dropping one buffered cell from either Qnc or Qc. The selection policy is as follows, in descending priority:

(1) A cell is dropped from either Qnc or Qc if its CVD exceeds the tolerable limit.
(2) A nonreal-time cell from a Qnc queue is dropped if its position exceeds the threshold of its queue.
(3) If there are nonreal-time cells in the same queue, then the last-in nonreal-time cell is dropped.
(4) A real-time cell is dropped from a Qnc queue if its position exceeds the threshold of its queue.
(5) If the new arrival is a nonreal-time cell, it is dropped immediately.
(6) If the new arrival is a real-time cell, then it is dropped immediately if no nonreal-time cell exists in any queue, or it replaces the last-in nonreal-time cell in some other Qnc queue.

4. Simulation study

Fig. 2 shows the topmost layer of an ATM network model constructed by integrating one traffic source node (*f_0*) and three sink nodes (*f_1*, *f_2*,

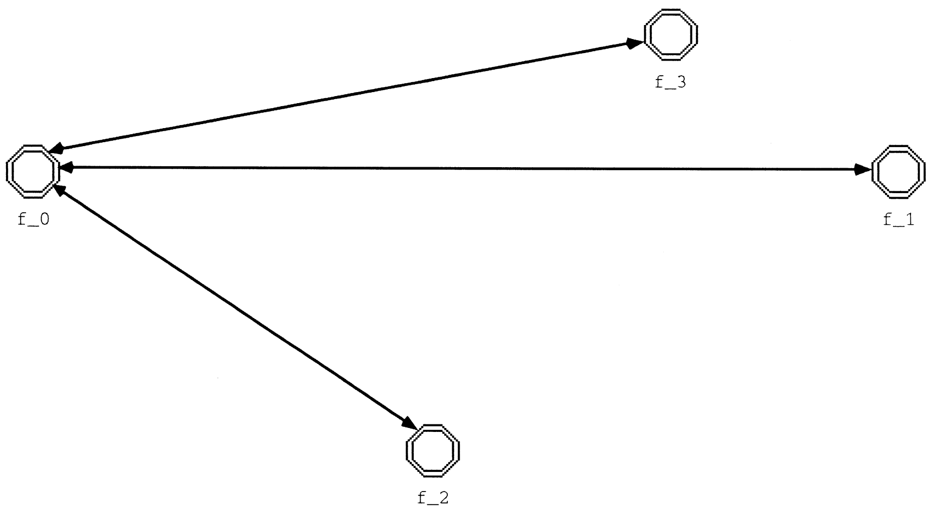

Fig. 2. An ATM network model used in the simulation.

f_3). Each sink node has its own unique ATM network address as its node serial number in the network. The traffic source generates either real-time or nonreal-time traffic destined to one sink node. Like the parameter used in Ref. [7], the link rates are set to 51.84 Mbits/sec, which corresponds to *STS-1*. The *segmentation and assembly rate* (SAR) is kept very high so that there is no traffic shaping due to the AAL5 layer. Since packet size is set equal to 320 bits, segmentation is not required.

We next discuss the effect of threshold updating frequency on switch performance. As noted in the algorithm, there are a number of parameters, such as the frequency of updating the threshold and the weight of each traffic factor. These are used to determine new shares of the buffer space. Table 1 tabulates the traffic parameters for studying the effect of threshold updating frequency on the switch. Let TSi = traffic source i, TC = traffic characteristics, MT = mean peak inter-arrival time (in 10^{-5} units), CD = call duration time, CW = call wait time, TT = traffic type, D = destination, R = real-time, and NR = nonreal-time.

Let SBS = switch buffer size, RT = real-time traffic, NRT = nonreal-time traffic, OP_ST = buffer space for output port ($i = 1,2,3$) in static buffering, and SBP_DB = shared buffer pool in dynamic buffering.

As shown in Table 2, the total buffer size for the switch in the simulation of the dynamic buffering strategy is 5022 cells. For the static buffering strategy, the 5022-cell buffer space is equally assigned to three output port queues. From each of these 1674 cells, 950 cells are used for accommodating real-time traffic in each output port, and the rest are for nonreal-time traffic. Since the packet inter-arrival time is exponentially distributed, by selecting different seeds in our simulations, the mixed pattern of bursty traffic is obtained.

We show simulations to compare the dynamic versus the static algorithms in terms of cell loss. We chose the simulation time of 1.5 s and the seed of 5000. Fig. 3 shows the cell loss in the dynamic

Table 1
Traffic parameters

TC	TS0	TS1	TS2	TS3	TS4	TS5
MT	0.9	0.1	0.1	0.9	0.12	0.12
CD	0.23	0.2	0.2	0.2	0.156	0.2
CW	0.2	0.415	0.51	0.4	0.5	0.32
TT	NR	R	NR	R	R	NR
D	1	2	3	1	3	2

Table 2
Static/dynamic buffer allocation in simulation

SBS	RT	NRT
OP_ST	950	724
SBP_DB	5022	5022

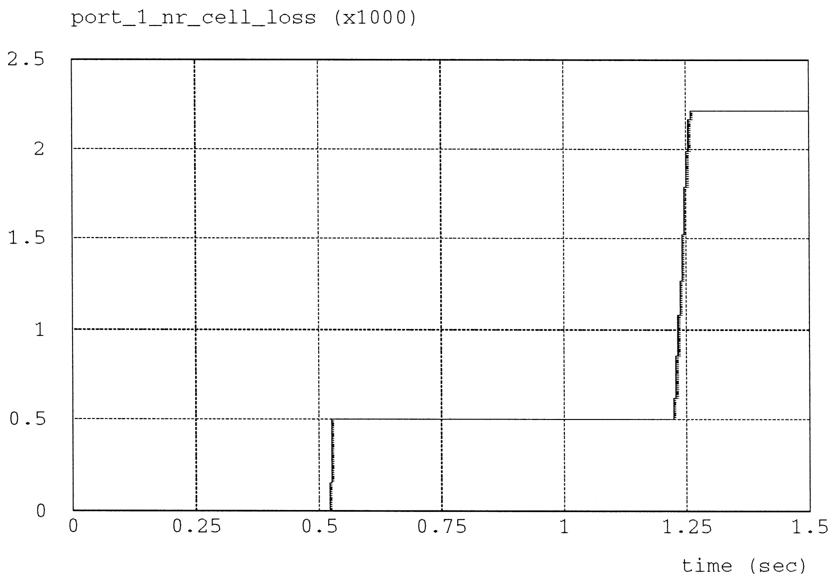

Fig. 3. Cell loss in the dynamic buffering algorithm.

buffering strategy switch, with threshold updating frequency $M = 500$. Cell loss occured only at output port 1. The nonreal-time cell loss began at time 0.5244 s. When the simulation ended the total cell loss was 2212, and no real-time cell was dropped. Fig. 4 shows the cell loss in the static buffering algorithm. Cell loss occured at output ports 0 and 1 and began at time 0.450014 s. In total, 1811 real-time

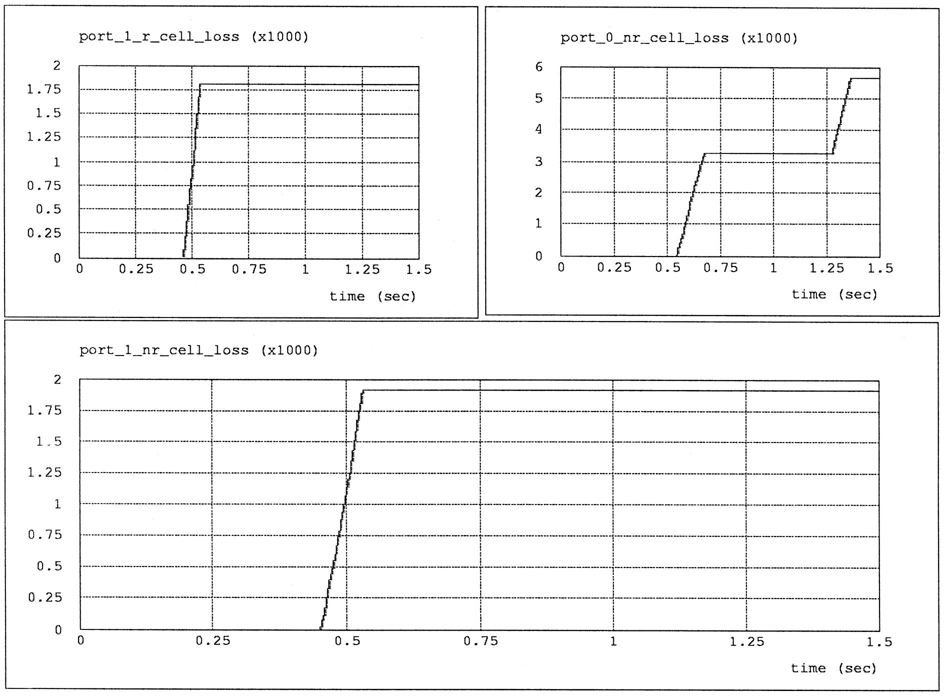

Fig. 4. Cell loss in the static buffering algorithm.

cells at output port 1 were lost, and 1914 and 5643 nonreal-time cells were lost at output port 1 and 0, respectively. Thus, this simulation shows that the dynamic buffering algorithm is superior to the static algorithm.

5. Conclusion

To achieve efficient and fair use of buffer space in ATM switches, we have proposed a threshold-based dynamic buffer allocation scheme. The system behavior under varying on-off bursty traffic patterns was investigated via simulation. The results show that the dynamic strategy for updating thresholds is superior to static complete partitioning in the sense of low cell loss. When traffic is heavy, the frequent updating plays a critical role in gaining higher throughput and in maintaining fair cell loss among all output ports.

Acknowledgements

Some comments of anonymous referees, which helped improve the presentation of some parts of the paper are gratefully acknowledged. Dr. Chronopoulos work was supported by NSF Grant ASC-9634775 and Dr. Psarris work was supported by NSF grants CCR-9528330, CDA-9529541, CDA-9633299 and by AFOSR grant F 49620-96-10472.

References

[1] I. Cidon, L. Georgiadis, R. Guérin, A. Khamisy, Optimal buffer sharing, IEEE Journal on Selected Areas in Communications 13 (1995) 1229–1239.

[2] A. Elwalid, D. Mitra, R.H. Wentworth, A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node, IEEE Journal on Selected Areas in Communications 13 (1995) 1115–1127.

[3] R. Jain, Congestion control and traffic management in ATM networks: recent advances and a survey, Computer Networks and ISDN Systems 28 (1996) 1723–1738.

[4] M. Katevenis, P. Vatsolaki, A. Efthymiou, Pipelined memory shared buffer for VLSI switches, Computer Communication Review 25 (4) (1995); Proceedings of ACM SIGCOMM'95, pp. 39–48.

[5] D.E. McDysan, D.L. Spohn, ATM Theory and Application, McGraw-Hill, New York, 1994.

[6] OPNET Modeler: Modeling and Simulation Kernel, Mil 3, Inc., 1996.

[7] S. Roy, E. Yaprak, J. Liu, E. Chow, R. Markley, ATM switch modeling and simulation, The International Association of Science and Technology for Development (IASTED) on Networks, January 1996, pp. 187–190.

[8] L. Tassiulas, Y.C. Hung, S.S. Panwar, Optimal buffer control during congestion in an ATM network node, IEEE/ACM Transactions on Networking 2 (1994) 374–386.

[9] G.-L. Wu, J.W. Mark, A buffer allocation scheme for ATM networks: complete sharing based on virtual partition, IEEE Transactions on Networking 3 (1995) 660–669.

Theodore Chronopoulos obtained his Bachelor's Degree at the University of Athens, Greece and his Ph.D. in Computer Science at the University of Illinois in Urbana-Champaign in 1987. He is a senior member of IEEE. He has performed research under 10 different research grants and he has given over 50 conference and invited research lectures. He has advised over 10 graduate students. He has co-authored 25 refereed journal papers and 25 conference papers in the areas of Computer Networks, Parallel and Distributed Systems and Applications. Dr. Chronopoulos is an Associate Professor in Computer Science at the University of Texas San Antonio.

Kleanthis Psarris received his B.S. degree in Mathematics from the National University of Athens, Greece in 1984. He received his M.S. degree in Computer Science in 1987, his M.Eng. degree in Electrical Engineering in 1989 and his Ph.D. degree in Computer Science in 1991, all from Stevens Institute of Technology. He is currently an Associate Professor in the Division of Computer Science at The University of Texas at San Antonio. His research interests are in the areas of Compilers and Programming Languages for Parallel Computing. He is an Associate Editor for the Journal of Computing and Information and has served on the Program Committees for the 9th ACM International Conference on Supercomputing in 1995 and for the International Conference on Computing and Information in 1994, 1995, 1996 and 1998. He is a member of ACM and IEEE.

Yi Xiao received her bachelor's degree in computer science in 1991 from Peking University, Beijing, China. In 1995, she went to Wayne State University, Detroit, MI, USA for graduate study in computer science. Her research interest was in ATM technology. In December 1996, she graduated with M.S. degree and moved to Seattle, WA. Now she is a senior software engineer at Microcrafts Inc.