FAIR QUEUEING FOR INPUT-BUFFERED SWITCHES WITH BACK PRESSURE

Shizhao Li†, Jian-Guo Chen‡ and Nirwan Ansari†

†Dept. of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102, U.S.A.
+1-973-596-3670(tel) +1-973-596-5680(fax)
emails: sxl3576@megahertz.njit.edu, ang@njit.edu

‡Lucent Technologies
Murray Hill, NJ 07974, U.S.A.
+1-908-582-8034(tel) +1-908-582-3662(fax)
email: jianguo11@lucent.com

ABSTRACT

The output-buffered switching architecture, though is able to offer high throughput, guaranteed delay and fairness, is not practical owing to its lack of scalability, i.e., the memory size, speed, and control logic have to be scaled up proportionally to the number of input links, thus becoming infeasible for large switches. The commercial and research trend is to adopt architecture with input buffering which is scalable, but yields lower throughput and lacks the quality-of-service features such as delay bound and fairness. Although the problem of low throughput owing to head of line blocking in input-buffered switches can be resolved by adopting per-output-port queueing in each input port, the contention among input ports still limits the throughput. Existing schedulers designed for input-buffered switches attempt to improve throughput by imposing back pressure to the contending cells, and scheduling cells free of contention for transmission, at the expense of delay and fairness.

In this paper, we modeled and analyzed the back pressure with independent Bernoulli traffic load, and showed that back pressure occurs with high probability under loaded traffic. We also derived the average queue length at the input buffer. To address the above issues in input-buffered switches, we proposed a new algorithm, referred to as min-max fair input queueing (MFIQ), which minimizes the additional delay caused by back pressure and at the same time provides fair service among competing sessions.

Keywords: input-buffered switch, fair queueing, back pressure

1. INTRODUCTION

High performance switching technology is a key component in the competition to meet today's booming demands for Internet services and applications. ATM switches which are designed to deliver small fixed length packets can provide fast connection between the input and output as well as guaranteed quality-of-service(QoS). An ATM switch typically consists of three parts: input queues, output queues and a switch fabric. Input queues buffer cells coming from input links, while output queues buffer cells going out to output links. The fabric routes cells from arbitrary input links to arbitrary output links. Different approaches to building such a switch have been proposed. In general, two categories are broadly classified according to where the buffer is placed: the input-buffered switch and the output-buffered switch.

It was shown [1] that the throughput of an input-buffered switch is limited to 0.586 when a single first-in-first-out (FIFO) queue is used in each input port. This is mainly owing to the head of line (HOL) blocking, i.e., if cells in the front of the input queue are blocked, the cells stored behind them cannot be transmitted even if their destination output ports are open. Since the publication of the seminal paper by Karol *et al.* [1], many works [2]-[4] have indicated that the throughput of the input-buffered switch can be improved by using well designed scheduling algorithms. Maximizing the throughput of a switch is equivalent to maximizing the number of connections between the input and output ports with the constraint of unique pairing. This is similar to the problem of bipartite graph matching, which has a high computational complexity. Parallel Iterative Matching (PIM) will reach 100 % throughput if a sufficiently large number of iterations are used. Iterative round-robin matching with slip (*i*SLIP) [4] was proposed later which has similar performance with PIM at much lower hardware complexity. The above algorithms aim at maximizing throughput at the expense of QoS features such as delay and fairness.

Output-buffered switches are not afflicted by the same throughput problem. A large effort has been made to design scheduling algorithms to provide guar-

0-7803-4982-2/98/$10.00©1998 IEEE

anteed end-to-end delay and fairness services among competing sessions for output-buffered switches. However, output buffering is not practical for building high speed switches with large aggregate switching capacity owing to the speedup problem. Since more than one cells can arrive at an output port during one time slot and only one of them can be transmitted in the same time slot, the fabric and the output queue should have the capability to accommodate the excess cells. At the worst case, the fabric has to run N times faster than the input and output link when all N input ports send cells to the same output port. Thus, the memory size and the control logic have to be scaled up proportionally to the number of input links. Owing to this scalability problem, the commercial and research trend is to adopt input buffering architecture, even though it yields lower throughput and lacks the QoS features.

While scheduling algorithms designed for output-buffered switches, like Packet Fair Queueing (PFQ), can provide end-to-end delay bound and fairness among sessions, they are not directly applicable to input-buffered switches without causing performance degradation. When the scheduler at an input port schedules one cell for transmission, schedulers at other input ports could also schedule cells for transmission to the same output port. Only one cell can get through the fabric, and the other cells are back pressured and stored in the input ports. Instead of wasting bandwidth, the scheduler in a back pressured port should schedule another cell for transmission. As a result, the back pressured sessions suffer extra delay and lose their fair share of the bandwidth while other sessions get more services than they should. The key issue is what type of actions should a scheduler executes when the output port is open for the back pressured sessions after a certain time lapse. Instead of only aiming at maximizing the throughput , we propose an algorithm, called min-max fair input queueing(MFIQ), to minimize the additional delay and to arbitrate fair service among all competing sessions in an input port.

The rest of the paper is organized as follows. The model of the back pressure problem under independent Bernoulli traffic is presented in Section 2, and it is shown that the effect of back pressure is significant. Section 3 presents our proposed scheduler, and simulation results are shown in Section 4. Remarks are concluded in Section 5.

2. BACK PRESSURE MODEL AND QUEUEING ANALYSIS

Consider a system with N input ports, N output ports, and an $N \times N$ fabric. The cell arrival processes at the input ports are assumed independent and identical Bernoulli. Let p be the traffic load, i.e., in any given

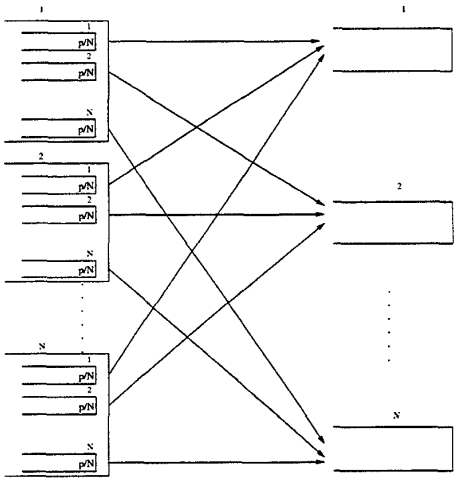

Figure 1: Per-output-port queueing

time slot, the probability that a cell arrives at a particular input port is p. The cell has equal probability of $1/N$ to go to any given output port, and successive cells are independent. To eliminate HOL blocking, per-output-port queueing is adopted, i.e., N separated queues, each of which is associated with a corresponding output port, are maintained in an input port. The probability of a cell appears at a per-output-port queue is p/N. There are N^2 per-output-port queues in the system as shown in Figure 1. Associated with an output port is a queue group with N per-output-port queues, each of which is located in one of the N input ports. The cell arrival processes in these queues are also independent. Let A be the total number of cells appeared at the head of each per-output-port queue in a queue group in one time slot. Thus, A is a Binomial random variable and has the following distribution:

$$P(A=i) = \binom{N}{i} (p/N)^i (1-p/N)^{N-i}. \quad (1)$$

Contention occurs when more than one of the per-output-port queues have cells in the same time slot. Only one out of the competing cells can get through the fabric, and the others are back pressured for later transmission. Thus, back pressure occurs with probability

$$\begin{aligned} P_B &= \sum_{i=2}^{N} \binom{N}{i} (p/N)^i (1-p/N)^{N-i} \\ &= 1-(1-\frac{p}{N})^{N-1}(1-\frac{p}{N}+p). \end{aligned} \quad (2)$$

When the number of input ports increases, the proba-

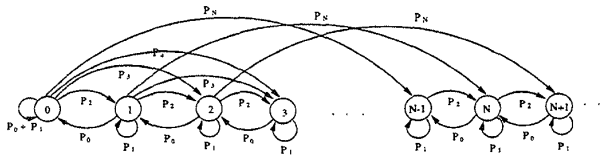

Figure 2: The discrete time Markov chain for the total queue length of one queue group

bility of back pressure becomes

$$\lim_{N\to\infty} P_B = 1 - e^{-p}(1+p). \tag{3}$$

Note that Equation (3) is monotonically increasing with the load p. The probability of back pressure reaches its maximum value of $1 - 2/e$ when the load reaches one.

Consider the total queue size of one queue group. With the assumption that one cell can always be transmitted during the time slot if there are cells at the head of the N per-output-port queues, the Markov chain for the total queue length of the queue group can be obtained as shown in Figure 2, where

$$p_i = \binom{N}{i} (P/N)^i (1 - P/N)^{N-i},$$

and the state value of the chain indicates the queue size of the queue group. The transition probability matrix of the Markov chain is

$$\mathbf{P} = \begin{bmatrix} p_0+p_1 & p_2 & p_3 & \cdots & p_N & 0 & \cdots \\ p_0 & p_1 & p_2 & \cdots & p_{N-1} & p_N & \cdots \\ 0 & p_0 & p_1 & p_2 & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}.$$

It is difficult to derive the close form distribution of the queue length, but the generating function of the queue length distribution can be readily derived:

$$Q(z) = \frac{(1-p)(z-1)}{z - A(z)}, \tag{4}$$

where

$$A(z) = (1 - \frac{p}{N} + z\frac{p}{N})^N \tag{5}$$

is the generating function of random variable A. Thus, the mean steady-state queue length can be obtained as follows:

$$\overline{Q} = \lim_{z\to 1} Q'(z) = \frac{N-1}{N}\frac{p^2}{2(1-p)}, \tag{6}$$

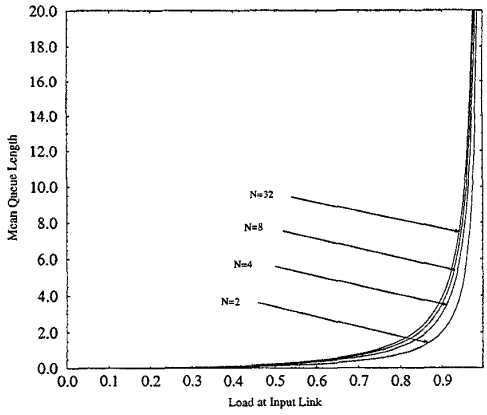

Figure 3: Mean queue length of a queue group with ideal throughput

where $Q'(\cdot)$ is the derivative of $Q(\cdot)$. Figure 3 shows the average total queue length in a queue group associated with one output port.

Note that Equation (6) is applicable to every queue group. If queue groups corresponding to different output ports are independent, the average queue length at the input buffer of the switch is $N\overline{Q}$. Since the traffic in the N input ports are independent, the average queue length in one input port is also $\overline{Q}$. The Markov chain is derived based on the assumption that there is always a cell to be transmitted to a given output port whenever there are cells in the queue group. In fact, if more than one output ports schedule the same input port to transmit cells, only one cell can be transmitted. As a result, the bandwidths of the other output ports are not utilized. Thus, the actual throughput can only be lower than what is assumed here. Hence, the average queue length at the input buffer in a realistic situation (Figure 4) is longer than the ideal case (Figure 3). Thus, back pressure can potentially cause adverse effect on the delay and fairness of the competing sessions.

3. MIN-MAX FAIR INPUT QUEUEING ALGORITHM

Generalized Processor Sharing (GPS) [5] was proposed to guarantee end-to-end delay and to provide fair service among competing sessions. Associated with each session is a real number r_i, which represents its service share of the server. The normalized service time of a session in a time interval (t_1, t_2) is defined as the amount of service $W(t_1, t_2)$ the session receives during

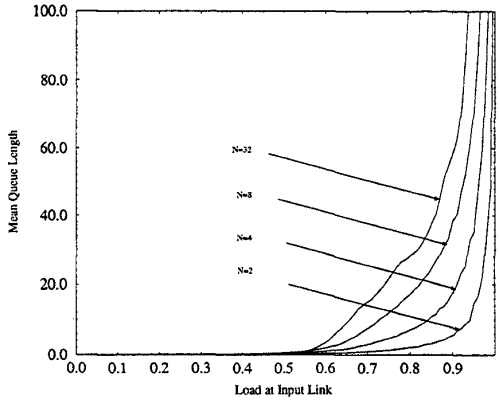

Figure 4: Mean queue length of a queue group simulation results

that interval divided by its rate r_i. Thus, fairness between two sessions can be measured by the difference between their normalized service times:

$$F = \left| \frac{W_i(t_1,t_2)}{r_i} - \frac{W_j(t_1,t_2)}{r_j} \right|. \quad (7)$$

GPS, with the assumption that data unit can be infinitely divisible (infinitesimally small), and that all sessions can be served simultaneously, is a perfectly fair scheduler in which $F = 0$. In reality, the GPS's assumptions do not hold, and thus the fairness of a server can be quantized by the upper bound of the fairness between any two sessions, i.e.

$$F_S = \max_{\forall i,j,\ j \neq i} \left| \frac{W_i(t_1,t_2)}{r_i} - \frac{W_j(t_1,t_2)}{r_j} \right|. \quad (8)$$

Many Packet Fair Queueing (PFQ) algorithms [5],[7]-[9] were proposed to emulate the GPS model for scheduling output-buffered switches. Each PFQ algorithm maintains a system potential $V(t)$ which represents the normalized service time that each session should receive by time t. In addition, associated with each session i are a virtual start time $S_i(t)$ and a virtual finish time $F_i(t)$. $S_i(t)$ keeps track of the normalized service received by session i by time t. The virtual time is updated according to the following rule [5]:

$$S_i(t) = \begin{cases} max(V(t), S_i(t-)) & \text{session } i \text{ becomes backlogged} \\ S_i(t-) + \frac{l}{r_i} & \text{current packet finishes service} \end{cases} \quad (9)$$

where l represents the length of the current packet of session i. Thus, the virtual finish time for session i (i.e., for the current packet) is

$$F_i(t) = S_i(t) + \frac{l}{r_i}. \quad (10)$$

PFQ algorithms approximate GPS performance by selecting cells with minimum virtual finish time for transmission. Different policies for choosing the system potential lead to different PFQ algorithms. For example, choosing real time as the system potential leads to Virtual Clock [6], and choosing the virtual finish time of current session in service as the system potential leads to Self-Clocked Fair Queueing [7].

Though PFQ algorithms can provide bounded end-to-end delay and fair service for output-buffered switches, they cannot be directly applied to input-buffered switches to achieve the same performance. Before we proceed to describe our algorithm which will overcome the shortcomings of applying PFQ directly to input-buffered switches, we first define the following.

Definition 1 *A reference scheduler of a system is an ideal scheduler which operates without back pressure but has the same configuration as the real scheduler in the system.*

The reference scheduler maintains its own virtual time. The virtual time of a session scheduled by the reference scheduler is updated no matter whether the session is back pressured in the real system. Thus, virtual time of sessions in the reference system keeps track of the service that the sessions should receive in the real system.

Definition 2 *The additional delay of a cell is the time interval between the time when the cell is transmitted in the real system and the time when the cell is scheduled in the reference system.*

Note that the additional delay is negative when the cell is transmitted before it is scheduled in the reference system.

Definition 3 *The normalized service lag of a session is the difference between the normalized service time the session should receive in the reference scheduler and the normalized service time it has received in the real system.*

For input-buffered switches, the schedulers of input ports are not independent from each other. When more than one input port schedule cells to the same output port, contention occurs. Only one of the competing cells can get through the fabric, and the others are back pressured in the input ports. To increase throughput, the scheduler of the back pressured input port needs to schedule another cell that is free of contention. Thus,

the back pressured cell experiences additional delay and loses its fair share of service. On the contrary, the being served session receives earlier and more service than its fair share. There could be more than one session back pressured at the same time. When the output ports for the back pressured cells are open for transmission, which cell should be transmitted? Within the context of GPS which is the perfectly fair scheduler, the back pressured session with the largest normalized service lag is the one that has been back pressured longest, thus experiencing the largest additional delay. It is therefore intuitively fair to transmit the cell of the session that has been back pressured the longest. This is the essence our algorithm as shown in Figure 5.

In the algorithm, a reference virtual time system and a real virtual time system are maintained. The virtual time of each session is updated in the reference system, independent of the status of the real system, to keep track of the normalized service the session should receive. Normalized service lags are maintained in the real system. The system potential $V(t)$ can be updated by using any PFQ algorithms like WFQ [5], $SCFQ$ [7], WF^2Q [8], and WF^2Q+ [9]. For example, if WFQ is selected to update the system potential, the rule is

$$V(t+\tau) = V(t) + \frac{\tau}{\sum_{i \in B} r_i}, \qquad (11)$$

where B represents all backlogged sessions and τ is the time increment. The session with the smallest virtual finish time F_i in the reference scheduler is updated regardless of the status of the real system, i.e., the virtual finish time of the selected session i in the reference system is updated no matter whether it is back pressured or not. The session with the largest normalized lag is scheduled for transmission in the real system. If the transmitted session j is not the session selected in the reference system, the selected session is deferred for transmission. Thus, the normalized service lag of the selected session is increased by l/r_i, and that of the transmitted session is decreased by l/r_j.

```
Cell_reaches_head_of_queue(session i, packet *cell)
  if i ∈ B
    if queue_length(i) == 0
      S_i ← max(V, S_i)
    F_i ← S_i + l/r_i
  else
    S_i ← V
    lag_i ← 0
    B ← B ∪ i
  enqueue(session i, packet *cell)

Transmit_cell
  q_to_update ← min_i F_i
  q_to_send ← max_i lag_i {i | i ∈ not BP }
  if q_to_send ≠ −1
    dequeue(q_to_send)
    Update_system_potential()
    S_q_to_update ← S_q_to_update + l/r_q_to_update
    F_q_to_update ← S_q_to_update + l/r_q_to_update
    if q_to_update ≠ q_to_send
      normalized_lag[q_to_update]
        ← normalized_lag[q_to_update] + l/r_q_to_update
      normalized_lag[q_to_send]
        ← normalized_lag[q_to_send] − l/r_q_to_send

Session_leave(session i)
  B ← B \ i
```

Figure 5: The pseudo-code of the min-max fair input queueing algorithm

4. SIMULATION RESULTS

Consider a system with eight input ports and eight output ports. To eliminate HOL blocking, per-output-port queueing is used. Each per-output-port queue has three sessions with transmission rates of 1, 5 and 10 Mbps. WFQ was selected to update the system potential. Simulations were conducted for a load of 0.8, 0.9 and 0.95. Each simulation lasted through 2 seconds.

Two schedulers were simulated: our proposed MFIQ algorithm, and the reference scheduler defined earlier. Sessions were scheduled only based on their virtual times in the reference scheduler. When the scheduled session was back pressured, its virtual time was updated and another session which was free of contention was selected for transmission. The scheduler did not keep track of the service time lost by the back pressured sessions. Thus, the lost service time could not be compensated.

The normalized service times received by three sessions belonging to different per-output-port queues are shown in Figures 6 and 7. Since our proposed algorithm always selected the session with the largest normalized service lag for transmission, the differences of the received normalized service times among sessions were smaller than that of the reference scheduler. The instantaneous fairness, which is the difference between the largest normalized service lag and the smallest normalized service lag experienced by all sessions, is compared in Figure 8. The instantaneous fairness of MFIQ is much smaller than that of the reference scheduler, even though they both experience randomness owing to the back pressure. The proposed MFIQ has better performance in terms of maximum normalized service lag and maximum additional delay, as shown in Figures 9-11.

Table 1 illustrates the statistics of the simulations for different loads. Four results can be derived from the table.

1. The two algorithms have similar performance in terms of average delay and average queue length;
2. The proposed MFIQ has better performance in terms of fairness and additional delay;
3. The total delay of a session is inversely proportional to its rate;
4. Average instantaneous fairness, average maximum additional delay, and average queue length increase as the load increases.

5. CONCLUSION

We have modeled and analyzed the back pressure occurred in input-buffered switches for independent Bernoulli traffic. The probability of back pressure and the average queue length have been derived. In reality, the average queue length at the input could be worse than the analytical results, and thus back pressure is an important issue. We have proposed a new algorithm, MFIQ, and demonstrated through simulations its effectiveness in minimizing the additional delay caused by back pressure and in providing fair service among competing sessions.

6. REFERENCES

[1] M. J. Karol, M. G. Hluchyj and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, pp. 1347-1356, Dec. 1987.

[2] M. J. Karol, K. Y. Eng and H. Obara, "Improving the performance of input-queued ATM packet switches," *INFOCOM '92*, pp.110-115.

[3] H. Obara and T. Yasushi, "An efficient contention resolution algorithm for input queueing ATM cross-connect switches," *International Jour. of Digital & Analog Cabled Systems*, vol. 2, no. 4 pp. 261-267. Oct.-Dec. 1989.

[4] N. McKeown, P. Varaiya and J. Walrand, "Scheduling cells in an input-queued switch," *IEE Electronics Letters*, pp. 2174-2175, Dec. 9th, 1993.

[5] 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. on Networking*, vol. 1, no.3, pp. 344-357, June, 1993.

[6] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching," *ACM Trans. on Computer Systems*, vol. 9, no.2 pp.101-124, May, 1991

[7] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," *INFOCOM'94*, pp.636-646, June, 1994.

[8] J.C.R. Bennett and H. Zhang, "WF^2Q: worst-cast fair weighted fair queueing," *INFOCOM'96*, pp.120-128, March, 1996.

[9] J.C.R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *ACM SIGCOMM'96*, pp.143-156, Aug, 1996.

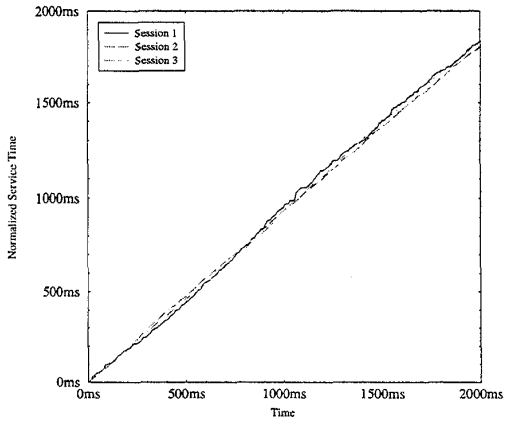

Figure 6: Normalized service time received by the three sessions: MFIQ

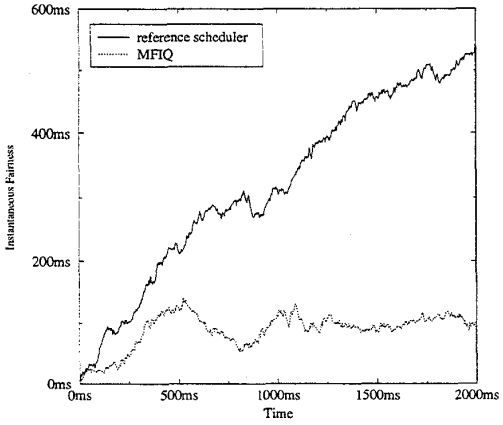

Figure 8: Instantaneous fairness: MFIQ versus reference scheduler

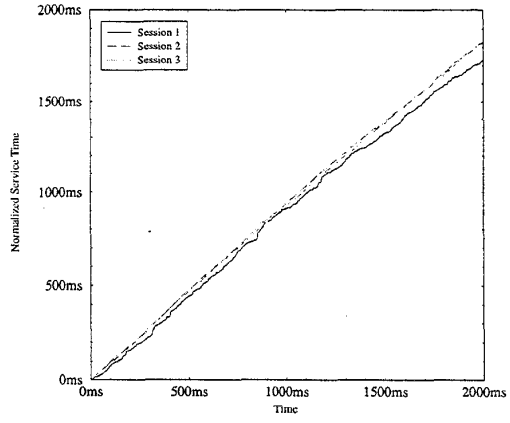

Figure 7: Normalized service time received by the three sessions: reference scheduler

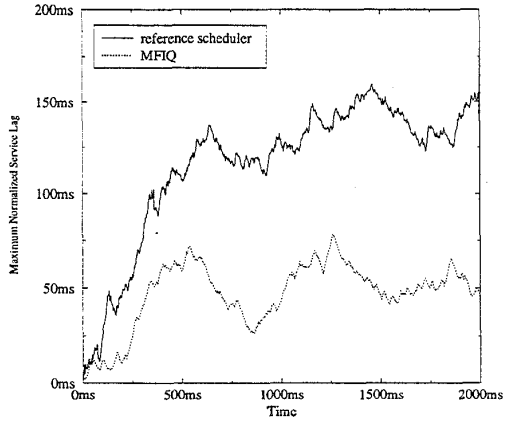

Figure 9: Maximum normalized service lag: MFIQ versus reference scheduler

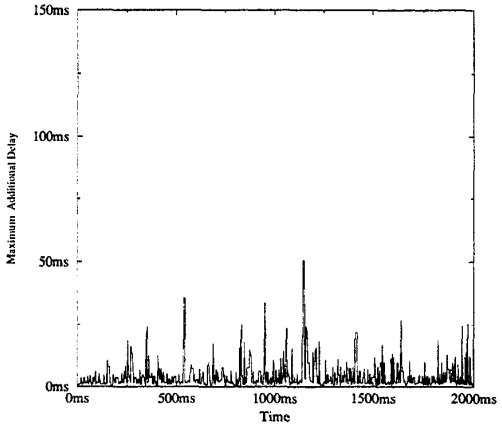

Figure 10: Maximum additional delay: MFIQ

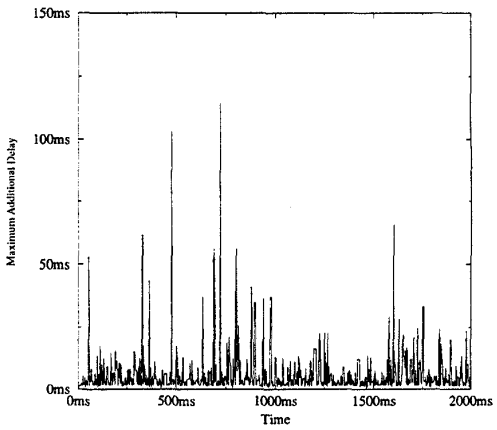

Figure 11: Maximum additional delay: reference scheduler

load	0.8		0.9		0.95	
schedulers	RS[1]	MFIQ	RS	MFIQ	RS	MFIQ
average delay of session 1(ms): rate=1Mbps	0.98	1.17	2.07	2.54	5.24	4.24
average delay of session 2(ms): rate=5Mbps	0.50	0.41	0.88	0.82	1.85	1.49
average delay of session 3(ms): rate=10Mbps	0.28	0.28	0.43	0.61	1.35	1.01
average queue length (cell)	19.97	19.33	39.56	39.26	71.89	75.77
average instantaneous fairness(ms)	327.45	51.87	314.75	88.45	378.96	88.83
average maximum additional delay(ms)	3.47	3.20	5.78	4.22	11.09	5.19
average maximum normalized lag (ms)	66.66	15.87	110.73	47.13	116.35	41.65

Table 1 Statistics of the simulation results

[1]reference scheduler.