Vol. 44 No. 8 IPSJ Journal Aug. 2003

*Regular Paper*

# An Adaptive Scheduler to Improve QoS in Input-Buffered Switches

Chun-Xiang Chen† and Masaharu Komatsu††

One difficult problem in the current Internet is how to guarantee the Quality of Service (QoS) for each connection. This paper proposes an adaptive scheduler for input-buffered switches to improve QoS guarantees for various traffic classes. The algorithm introduced in the scheduler is based on the required bandwidth of each connection. Our scheduler uniformly deals with various traffic classes without any strict priority discipline. Thus, it can effectively use the residual resources of the switch for any traffic pattern without facing resource starvation. From simulation results, it is clarified that the proposed scheduler has better performance, in terms of throughput, packet delay and deviation, than some well-known schedulers.

## 1. Introduction

The current Internet has had great success due to the simplicity and the connectivity of the Internet Protocol (IP). The philosophy of IP is to keep the network as simple as possible and leave the complicated processing to end hosts. IP only supports the best-effort service. However, we face a difficult problem when we transmit various types of data including real-time streaming such as movies and audio via the current Internet. The problem is how to guarantee the Quality-of-Service (QoS) for each connection. To provide the QoS guarantees through the best-effort network, two methods can be considered. One is to increase the transfer speed of physical circuits and the exchange speed of switches. However, this way is not drastic due to the connection-less nature of the IP, and the packet flow of a connection would be broken by other connections if there were any. The other is by designing switches which provide QoS support.

So far, most of the switches with QoS support employ output buffer structure because the departure time of a packet stored in the output buffer can be easily controlled by a simple algorithm to provide QoS guarantee[1),2)]. However, in an extreme case, the fabric of an $N \times N$ switch must run $N$ times faster than the line speed (called speedup in some literature)[1),3),4)]. This is very difficult and sometimes impossible when the line speed is very high (up to Tbit/s range) and when the number of input-(output-) ports of the switch is very large.

On the other hand, the input-buffered (IB) switch only needs to operate as fast as the line speed[5)~7)]. Therefore, the input buffer is very suitable for switches with a large number of ports and with extremely high line speed. However, the IB switch requires Virtual Output Queueing (VOQ) and a scheduler to avoid contentions among the packets which share the same input port and/or the packets which depart for the same output port. The scheduler selects the packets which would be transmitted in the next time slot. Thus, the algorithm used in the scheduler is especially important since it is directly related to QoS. Considering the advantages and disadvantages of the switches stated above, we employ the IB switch in this paper.

To provide QoS guarantee, one of the traditional methods is to give strict priorities to different connections (high priority is given to real-time connection)[8)~10)] or reserve a strict bandwidth for each connection (such as Resource Reservation Protocol (RSVP)). However, these schemes do not provide flexibility and fairness of service. The other methods are to design a smart scheduler to find a suitable matching between the input and the output ports. There are several scheduling programs designed to achieve high performance, such as first-in/first-out (FIFO) with VOQ support, longest queue first (LQF)[5),11)], credit based scheme[7)] and oldest cell first (OCF)[6)] etc. The common matter in these schedulers is to take an appropriate parameter as the priority weight when the scheduler selects the next packets. For example, FIFO (LQF) takes the arrival time (queue

† Information Processing Center, Hiroshima Prefectural University
†† Department of Information and Computer Engineering, Okayama University of Science

length) of the packet as the priority weight. The credit-based bandwidth reservation scheme 7) takes the bandwidth assigned by an admission controller (AC) as the credit. The credit increases in proportion to the assigned bandwidth capacity in time slots and is consumed by the incoming packets. However, when the credit is exhausted, packets waiting in the queue are unable to be transmitted even if there is a surplus of bandwidth.

On the communications via Internet, QoS can be quantitatively described with some indexes, such as transmission delay, transmission error, packet loss, deviation of delay, jitter of delay etc. If these indexes are kept within the expectations, the QoS can be considered as being satisfied. In this paper, we use the packet delay and its deviation to evaluate the quality of service.

In this paper, we focus on the IB switch in which VOQs are employed to overcome head-of-line (HOL) blocking 12). The aim of this paper is to design an effective scheduler to select packets to be transmitted in the next time slot. To achieve this, we propose a scheduling algorithm (called MEF: Most Expected packet First) with the aim of improving QoS. The main feature of MEF is that it fairly and flexibly deals with all traffic based on the assigned bandwidth. However, differing from RSVP, MEF only refers to the bandwidth of the connection and does not make a strict bandwidth allocation. The contribution of this paper is as follows. First, the MEF algorithm has greater adaptability to deal with multi-traffic classes without using strict priority. Thus, MEF can easily accept new connections. Second, MEF has very high flexibility to make effective use of the network resources when total traffic is not at full capacity even if the traffic is unbalanced. Third, MEF does not have the starvation phenomenon 13) even for unbalanced traffic or overload traffic because MEF deals with packets based on the bandwidth of each connection in whole time slots.

The rest of this paper is organized as follows. Section 2 gives a simple model of the IB switch and the definition of symbols used in this paper. The proposed scheduling algorithm is given in Section 3. Section 4 gives the simulation method including the traffic patterns dealt with in Section 5. The simulation results compared with the other schedulers are shown in Section 5, and finally the concluding remarks are stated in Section 6.

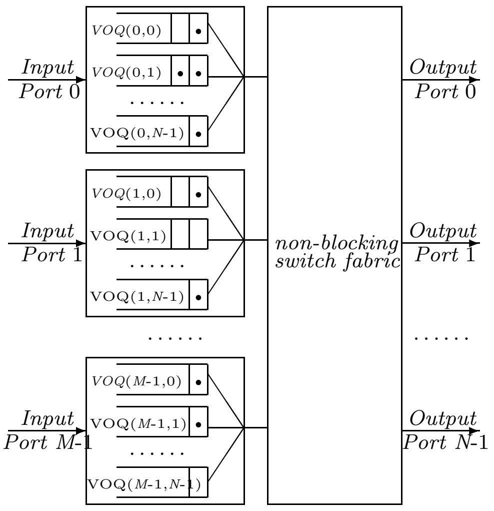

**Fig. 1** $M \times N$ ATM switch with VOQ.

## 2. Switching Model

A simple $M \times N$ IB switch with VOQs is shown in **Fig. 1**. We assume that the length of packet is fixed☆, and the time axis is divided into fixed-length units (called time slots) in which at most one packet arrives at each input-port and/or is transmitted to its destined output-port. The switch includes a non-blocking fabric which is able to concurrently send $min(N, M)$ packets to the corresponding output ports. The packet destined to output port $j$ arrives at input port $i$, is temporarily stored in the corresponding FIFO VOQ$(i, j)$ and waits to be transmitted, where $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$.

The operation of FIFO VOQ is as follows. If a nonempty VOQ is selected by the scheduler, the packet at the head of the VOQ can be transmitted, and would be removed immediately after the end of its transmission. Obviously, at most one nonempty VOQ in an input port would be selected to transmit its head packet.

Without loss of generality, we assume that the transmission capacity (bandwidth) of the switch is initialized as 1. The switch has no speedup. We give some terms and assumptions as follows:

☆ It is likely that the packet size would vary within different switches (protocols). However, the packet scheduler is considered to be implemented at the data link layer, and the segmentation and assembly are performed at the upper layer. Hence, it is suitable to assume that the packet size is uniform here.

$VC_{ij}$ : Virtual connection in which a packet arrives at input port $i$ and is destined to output port $j$. Actually, it is likely that several connections have the same input-output pair; in this case, each connection needs its own QoS guarantee. For simplicity, here, we assume that each connection has a distinct input-output pair.

$b_{ij}$ : Bandwidth of $VC_{ij}$. Thus $T_{ij} = 1/b_{ij}$ is the average interarrival time of packets with the same $VC_{ij}$.

$L(i,j)$ : Capacity of $VOQ(i,j)$.

The traffic load of each $VC_{ij}$ is $b_{ij}$. Thus the condition that the switch is stable, is as follows:

$$\begin{cases} \sum_{j=0}^{N-1} b_{ij} < 1, & for \quad \forall i \\ \sum_{i=0}^{M-1} b_{ij} < 1, & for \quad \forall j. \end{cases} \tag{1}$$

Of course, if the above expression is not satisfied, the switch is unstable or overloaded.

## 3. Scheduling Algorithm

In this section, we describe the proposed scheduling algorithm (called MEF). Similar to the other weighted-fair schedulers, we carefully choose the priority weight to improve QoS guarantee. In a connection-oriented network such as ATM (Asynchronous Transfer Mode) network, the rate of packet generation would be declared and allocated by AC according to the remaining resource of the network. Surely, the bandwidth allowed (assigned) by AC is not always the same as the packet rate requested by the connection. Thus the assigned bandwidth is able to be considered as the average arrival rate from the connection. Let $e_{ij}(n)$ be the expected arrival time of the $n$-th packet from connection $VC_{ij}$ (**Fig. 2**). Obviously, the expected arrival time of the next packet, $e_{ij}(n+1)$, satisfies

$$e_{ij}(n+1) = e_{ij}(n) + T_{ij},\ T_{ij} = 1/b_{ij}. \tag{2}$$

Unfortunately, the real arrival time is not always the same as the expected time. It would be influenced by other connections. Let $a_{ij}(n)$ be the real arrival time of the $n$-th packet from $VC_{ij}$. Furthermore, let $w_{ij}(n)$ be the interval from the current time to the expected arrival time of the packet. We have

$$w_{ij}(n) = e_{ij}(n) - t_{now}, \tag{3}$$

where $t_{now}$ is the current time. $w_{ij}$ means the urgency of the transmission of the $n$-th packet. Therefore, $w_{ij}(n)$ can be considered as a very fair priority weight to the $n$-th packet in its transmission. The packet with smaller $w_{ij}(n)$ is assigned higher priority. Now the MEF algorithm can be given as follows:

(1) Calculate $w_{ij}$ of each HOL packet in VOQs if it exists.
(2) Request: Each unmatched[☆] input port, in which there is at least one packet in its VOQ, sends the requests (including the priority weight $w_{ij}$) to the corresponding output ports.
(3) Grant: If an unmatched output port $j$ receives any requests, it selects input port $k$, which satisfies $w_{kj} = \underset{i}{min}(w_{ij})$, and grants the request to input port $k$.
(4) Accept: If input port $k$ receives any grants, it accepts the output port $h$ among these outputs from which grants have been received, where $h$ satisfies $w_{kh} = \underset{j}{min}(w_{kj})$, and informs output port $h$ of this acceptance. Thus an input-output pair $(k,h)$ is established.
(5) Repeat above steps 2, 3 and 4 until all of the possible pairs are established among input-output ports.

An example of the first iteration of MEF is shown in **Fig. 3** for a $4\times4$ switch. After the first iteration, pairs $(0,2)$, $(1,1)$ and $(2,3)$ are established. Then the algorithm enters the second iteration. Input port 3 sends its request to output ports 0, 1, 2 and 3 (step 2), and input port 3 receives only a grant from output port 0 (step

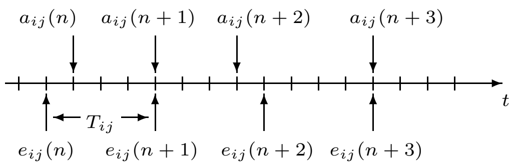

**Fig. 2** Expected arrival times and real arrival times.

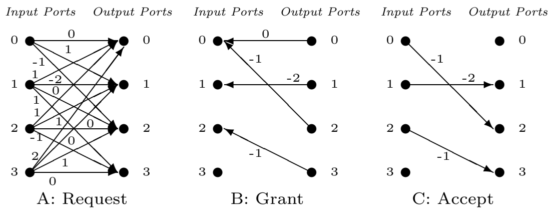

**Fig. 3** An example of the first iteration for a $4\times4$ switch.

☆ The goal of the scheduler is to find the input-output pairs. If an input port is paired with an output port by the scheduler, we say that this input/output port is matched; inversely, this input/output is unmatched.

3). Thus, the pair $(3,0)$ is established (step 4). After the second iteration, all of the possible pairs have been established. The MEF algorithm finds all the possible pairs, on average, in $O(\log H)$ iterations[13], where $H = min(M,N)$.

## 4. Simulation Method

To evaluate the performance of our scheduler with simulation, we consider an $M \times N$ switch, where $M$ is the number of input ports and $N$ the number of output ports. We also assume three kinds of arrival traffic patterns as follows:

(1) Constant interval traffic — The interarrival time of packets is precisely equal to a constant. It means that there is exactly one arrival at the end of each interval time. Let $T_{ij}$ be the constant interarrival of the packets from connection $VC_{ij}$. Then, the average load is given by $\alpha_{ij} = 1/T_{ij}$, where $i \in [0, M-1]$, $j \in [0, N-1]$.

(2) Geometric traffic — The interarrival time of the packets is distributed according to a geometric distribution. Let $f_{ij}(n)$ be the probability that the interarrival of the packets from $VC_{ij}$ is $n$ time slots. $f_{ij}(n)$ is given by

$$f_{ij}(n) = g_{ij}(1-g_{ij})^{n-1},\ n=1,2,3,\ldots \quad (4)$$

where $g_{ij}$ is the arrival rate of the packet from $VC_{ij}$ in a time slot. Thus, the average interval time is $1/g_{ij}$ and the average load ($\beta_{ij}$) is $\beta_{ij} = g_{ij}$.

(3) Burst traffic — The packet arrives according to a two-state Markov chain. One state is called *On* in which one packet arrives in each slot time. The other is called *Off* in which there are no arrivals (**Fig. 4**).

If we assume that the transition probability of the burst connection $VC_{ij}$ from state *On* to *On* is $p_{ij}$ and the transition probability from state *Off* to *Off* is $q_{ij}$, then the average length of state *On* ($B_{ij}$) and the average length of state *Off* ($S_{ij}$) are shown as

$$B_{ij} = \frac{1}{1-p_{ij}}, \qquad S_{ij} = \frac{1}{1-q_{ij}}. \quad (5)$$

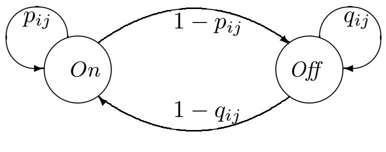

**Fig. 4** State transition of the burst traffic $VC_{ij}$.

Therefore, let $\gamma_{ij}$ be the average load of the burst connection $VC_{ij}$. $\gamma_{ij}$ is given as

$$\gamma_{ij} = \frac{B_{ij}}{B_{ij}+S_{ij}}, \quad (6)$$

and let $\rho_i$ be the traffic load of input port $i$, which is obtained by

$$\rho_i = \sum_{j=0}^{N-1} (\alpha_{ij} + \beta_{ij} + \gamma_{ij}). \quad (7)$$

Now, let $\alpha$, $\beta$ and $\gamma$ be the total arrival rate of the three traffic classes respectively. They are given as

$$\alpha = \sum \alpha_{ij},\ \textit{for all constant interval traffic},$$
$$\beta = \sum \beta_{ij},\ \textit{for all geometric traffic},$$
$$\gamma = \sum \gamma_{ij},\ \textit{for all burst traffic}. \quad (8)$$

Also let $\alpha_0$, $\beta_0$ and $\gamma_0$ be the ratios of the three traffic classes respectively.

$$\alpha_0 = \frac{\alpha}{\alpha+\beta+\gamma},$$
$$\beta_0 = \frac{\beta}{\alpha+\beta+\gamma},$$
$$\gamma_0 = \frac{\gamma}{\alpha+\beta+\gamma}. \quad (9)$$

Now, considering the MEF scheduling algorithm described in Section 3, the main point in the algorithm is to consider the traffic load of a connection in a long time period (i.e., the average load). Thus the bandwidth of each connection is distributed as its average load by the AC.

## 5. Simulation Results

In this section, we investigate the characteristics of the presented scheduler (MEF) using simulation. Also, the results obtained under MEF, are compared with well-known algorithms FIFO, LQF[11] and Credit card[7]. The FIFO algorithm takes the arrival time, the LQF algorithm takes the queue length in each VOQ and the Credit card algorithm takes the credit, as the priority weight, respectively. For the traffic classes, we consider the three types of communications described in the previous section. The burst traffic, such as *compressed* movies and audio communications, can be considered as real-time traffic; the constant interval traffic, such as the FTP, *uncompressed* video and audio communications, can be considered as real-time constant bit rate traffic; and the geometric traffic can be considered as general data transmission (best-effort commu-

**Table 1** Throughput comparison : $32 \times 32$ switch. ($\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$, $B_{ij} = 5$)

| $\rho_i$ | MEF | Credit | LQF | FIFO |
|---|---|---|---|---|
| 0.60 | 0.9943 | 0.9917 | 0.9943 | 0.9943 |
| 0.75 | 0.9947 | 0.9923 | 0.9947 | 0.9947 |
| 0.90 | 0.9944 | 0.9916 | 0.9945 | 0.9946 |

nications). Thus, it is suitable to assume that the burst traffic and constant interval traffic require the QoS guarantee and the geometric traffic does not require the QoS guarantee. Furthermore, we assume that each VOQ has infinite capacity.

In the simulation, the type and the traffic load of each $VC$ are set as follows. The simulator randomly generates a $VC_{ij}$ $(\forall i, \forall j)$ and its traffic load $(b_{ij})$, where $b_{ij}$ must satisfy expression (1), or $b_{ij}$ is set to be zero (this connection is refused because the bandwidth requested exceeds the remaining resources. So this process can be considered as a simple emulator of the AC). Here, we let the upper bound of the bandwidth of each $VC$ be 0.1[☆]. The simulation runs until it converges (more than 10,000,000 time slots).

**Table 1** gives the comparison of throughput in different schedulers for a $32 \times 32$ switch. The throughput is defined as the ratio of the number of successfully transmitted packets to the total number of packets arrived in a long time interval. Table 1 shows that all of these schedulers achieve about 100% ($\geq$ 99%) throughput by introducing VOQ. We also investigated the throughput in various $M$, $N$ and other parameters. As a result, it was shown that the throughput is about 100% in all cases. Here we omit those comparison tables.

Generally, different schedulers provide different QoS for different traffic classes. It is likely that providing better QoS for a traffic class will sacrifice the QoS of another traffic class when the switch runs at high traffic load[14]. Therefore, it is very important to provide the required QoS for the connections which really need QoS guarantees. In the following, we compare the packet delay, queue length and standard deviation of the packet delay in different schedulers.

The packet delay $(d)$ is defined as the interval time slots from the arrival time of a packet to the time slot in which it is successfully transmitted to the output port. **Figures 5**, **6** and **7** give the average delays of the constant interval, geometric and burst traffic classes, respectively, where $B$ is the average burst length of the burst traffic class. From these figures, it is clear that (1) the average delays of constant interval traffic and the burst traffic, under MEF, are relatively smaller and increase slowly with the increase of the traffic load $(\rho_i)$ even if the traffic load is high (up to 0.95); whereas the average delay of geometric traffic increases sharply when $\rho_i > 0.7$. This is true because, as the traffic load is very high, MEF adaptively serves the burst traffic and the constant traf-

☆ If the port speed of the switch is faster than 1 Gbps, 0.1 (corresponding to 100 Mbps) is sufficient to satisfy the practical applications.

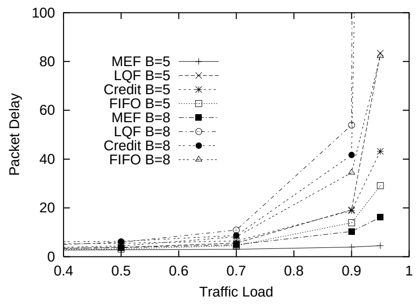

**Fig. 5** Average packet delay of constant interarrival-time traffic
($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

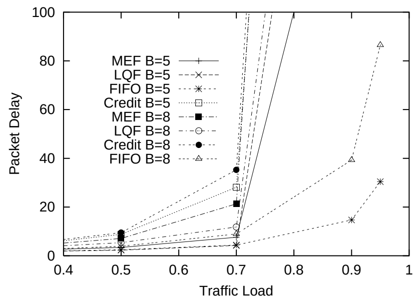

**Fig. 6** Average packet delay of geometric traffic
($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

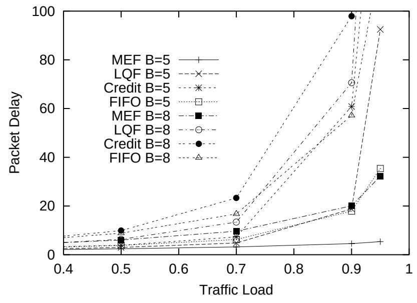

**Fig. 7** Average packet delay of burst traffic
($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

fic with high priority. (2) The average delay under the Credit card scheme is the worst, especially for the geometric traffic and the burst traffic because the packet of a connection is unable to get the service if the credit is exhausted even though surplus resources exist. The result agrees with that in Ref. 7). (3) The average delays for all the traffic types, under FIFO, increase in the same way when the traffic load increases. (4) The average delay, under LQF, has some resemblance to that of FIFO. (5) The burst length strongly influences the packet delay, the bigger the burst length, the longer the packet delay.

To evaluate the performance of QoS provided by each scheduler, only the packet delay is insufficient. The jitter of the packet delay must be investigated. Here the jitter of the packet delay is evaluated by the standard deviation ($\sigma$). $\sigma$ is defined as

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N_P}(d_i - \text{average packet delay})^2}{N_P}},$$

where, $N_P$ is the total number of the packets which belong to that traffic class and $d_i$ is the delay of each packet.

**Figures 8–10** give the standard deviation of each traffic class. Figures 8 and 10 show that the standard deviations of the constant traffic or the burst traffic, under MEF, increase gradually as the traffic load increases. Comparing with other schedulers, MEF gives the smallest standard deviation. However, the standard deviation of the geometric traffic (**Fig. 9**) increases sharply, especially at $\rho_i > 0.7$. Thus, it is evident that MEF improves the QoS of the constant interval traffic and the burst traffic at the cost of the geometric traffic's QoS when traffic load is relatively high. The standard deviation of each class in the FIFO scheme has a similar trend. In other words, the FIFO scheme serves all classes without difference even if some classes need the QoS guarantees. The credit card scheme, as before, gives the worst performance in terms of $\sigma$. Also, the influence of the burst length of the burst traffic class is illustrated in the same figures. It is clear that the longer the burst length, the higher the standard deviation.

Other results with different parameters are shown in **Figs. 11**, **12** and **13** (where $B = 5$). The performance changes with the parameters. However, it is clear that the performance (in terms of throughput and deviation) of the MEF

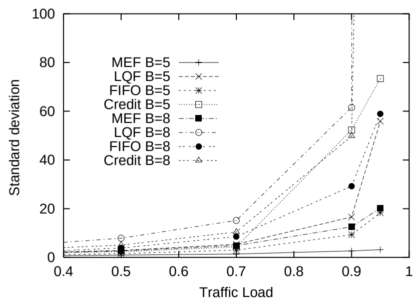

**Fig. 8** Standard deviation of constant interval traffic ($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

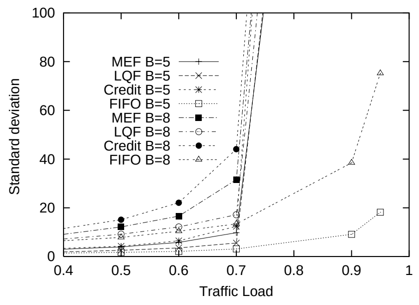

**Fig. 9** Standard deviation of geometric traffic ($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

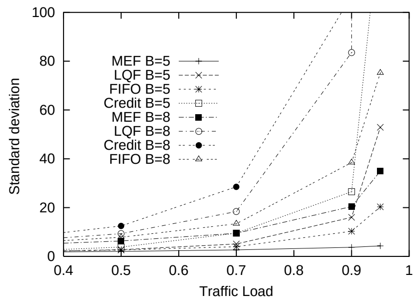

**Fig. 10** Standard deviation of burst traffic ($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

scheduler is better than the other schedulers. We also investigated the average packet delay, and the standard deviation in various $B$, $M$, $N$, $\alpha_0$, $\beta_0$ and $\gamma_0$ for each traffic class. The simulation results in all cases show that MEF gives a better performance in terms of packet delay and $\sigma$ than the other schedulers for the constant interarrival time traffic and the burst traffic. We omit these figures because of space limitations.

In our simulation, we assumed that each VOQ has infinite capacity. We also investigated the queue length of each VOQ. The maximal queue length is shown in **Fig. 14**. We can see from Fig. 14 that the necessary buffer size under

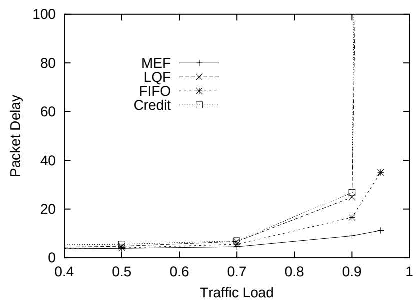

**Fig. 11** Average packet delay of constant interarrival-time traffic
($48 \times 48$ switch, $\alpha_0 = 0.507$, $\beta_0 = 0.312$, $\gamma_0 = 0.181$).

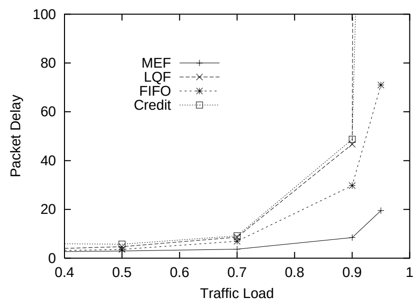

**Fig. 12** Average packet delay of constant interarrival-time traffic
($48 \times 48$ switch, $\alpha_0 = 0.181$, $\beta_0 = 0.312$, $\gamma_0 = 0.507$).

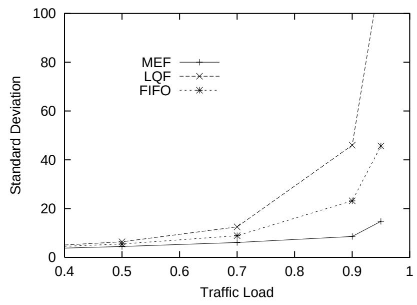

**Fig. 13** Standard deviation of burst traffic
($48 \times 48$ switch, $\alpha_0 = 0.277$, $\beta_0 = 0.322$, $\gamma_0 = 0.401$).

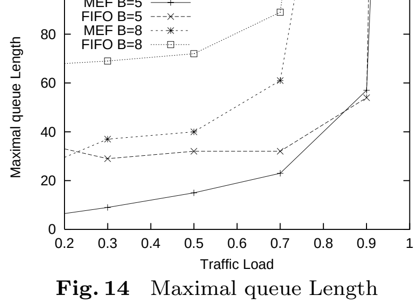

**Fig. 14** Maximal queue Length
($32 \times 32$ switch, $\alpha_0 = 0.375$, $\beta_0 = 0.422$, $\gamma_0 = 0.203$).

the MEF scheduler is relatively shorter.

## 6. Concluding Remarks

It is a difficult problem to provide QoS guarantees in the current Internet. However, once the Internet has become an integrated infrastructure of communications, especially as a commercial communication infrastructure, it becomes essential that the Internet can guarantee the QoS for each connection.

In this paper, trying to improve the QoS, we presented a scheduling algorithm, called MEF, based on the bandwidth of each connection. MEF can not be implemented directly in IP networks due to the connection-less nature of the Internet protocol. However, MEF could be implemented in the core switch of networks in which traffic engineering is becoming more prevalent to provide connection-oriented packet switching for certain input-output connections (e.g., ATM networks, MPLS (Multi-Protocol Label Switching) networks)15). In such networks, the bandwidth of each connection is declared at the establishment of the connection. MEF skillfully uses the bandwidth and adaptively provides packet scheduling for the switch.

To show the advantage of MEF, we compared its performance with some well-known schedulers. From the simulation results, it was shown that LQF and FIFO schedulers provide service to each traffic class without any distinction. On the other hand, MEF outperforms the FIFO or LQF schedulers with respect to QoS measured in terms of average packet delay and standard deviation. Furthermore, in contrast to RSVP, MEF does not require the actual bandwidth allocation; it only refers to the bandwidth of the connection. Thus the MEF scheme does not use strict priority in different traffic classes. This makes the MEF scheme simpler and more flexible to be implemented in edge routers. Also, MEF has no starvation phenomenon13) which occurs in some other schemes (such as LQF) because the priority of the packet waiting in VOQ adaptively becomes higher as the time slot passes.

**Acknowledgments** The authors would like to thank the anonymous reviewers for their valuable comments.

## References

1) Liebeherr, J. and Wrege, D.E.: Priority Queue Schedulers with Approximate Sorting in Output-Buffered Switches, *IEEE Journal*

*on Selected Areas in Commun.*, Vol.17, No.6, pp.1127–1144 (1999).
2) Schoenen, R.: An Architecture Supporting Quality of Service in Virtual-Output-Queued Switches, *IEICE Trans. Commun.*, Vol.E83-B, No.2, pp.171–181 (2000).
3) Chuang, S.-T., Goel, A., McKeown, N. and Prabhakar, B.: Matching Output Queueing with a Combined Input/Output-Queued Switch, *IEEE Journal on Selected Areas in Commun.*, Vol.17, No.6, pp.1030–1039 (1999).
4) Krishna, P., Patel, N.S., Charny, A. and Simcoe, R.J.: On the Speedup Required for Work-Conserving Crossbar Switches, *IEEE Journal on SAC*, Vol.17, No.6, pp.1057–1066 (1999).
5) Mekkittikul, A. and McKeown, N.: A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches, *IEEE INFOCOM'98*, Vol.2, pp.792–799 (Apr. 1998).
6) Nabeshima, M. and Yamanaka, N.: The *i*-QOCF: Scheduling Algorithm for Input-Queued ATM Switches, *IEICE Trans. Commun.*, Vol.E83-B, No.2 pp.182–189 (2000).
7) Kam, A.C. and Siu, K.-Y.: Linear-Complexity Algorithms for QoS Support in Input-Queued Switches with No Speedup, *IEEE Journal on Selected Areas in Commun.*, Vol.17, No.6, pp.1042–1056 (1999).
8) Chen, W.-T., Lee, R.-R. and Lin, H.-J.: A Novel Cell Scheduler with QoS Guarantee for Services in ATM Networks, *IEICE Trans. Commun.*, Vol.E82-B, No.2, pp.447–454 (1999).
9) Oki, E., Yamanaka, N. and Nabeshima, M.: Performance of Scalable-Distributed-Arbitration ATM Switch Supporting Multiple QoS Classes, *IEICE Trans. Commun.*, Vol.E83-B, No.2, pp.204–213 (2000).
10) Marsan, M.A., Bianco, A., Leonardi, E. and Milia, L.: RPA: A Flexible Scheduling Algorithm for Input Buffered Switches, *IEEE Trans. Commun.*, Vol.47, No.12, pp.1921–1933 (1999).
11) McKeown, N., Mekkittikul, A., Anantharam, V. and Walrand, J.: Achieving 100% Throughput in an Input-Queued Switch, *IEEE Trans. Commun.*, Vol.47, No.8, pp.1260–1267 (1999).
12) Karol, M., Hluchyj, M. and Morgan, S.: Input versus Output Queueing on a space-division switch, *IEEE Trans. Commun.*, Vol.35, pp.1347–1356 (1987).
13) McKeown, N.: The *i*SLIP Scheduling Algorithm for Input-Queued Switches, *IEEE Trans. on Networking*, Vol.7, No.2 (1999).
14) Wang, F. and Mohapatra, P.: An Efficient Bandwidth Management Scheme for Real-Time Internet Application, *International Symposium on Intelligent Multimedia, Video and Speech Processing* (2001).
15) Hsu, N.-B., Lin, Y.-D., Li, M.-H. and Lee, T.-H.: Service-Sensitive Routing in Diff-Serv/MPLS Networks, *IEICE Trans. Commun.*, Vol.E84-B, No.10, pp.2871–2879 (2001).

(Received August 12, 2002)
(Accepted May 6, 2003)

**Chun-Xiang Chen** received the B.E. degree from Sichuan University, Sichuan, China in 1986, the M.E. and Ph.D. degrees from Kanazawa University and Osaka University, Japan, in 1991, 1994, respectively. From 1994 to 1996, he was an assistant professor in the Department of Computer Engineering at Hiroshima City University, Japan. Since May 1997, he has been an Associate Professor at Hiroshima Prefectural University, Hiroshima, Japan. His current research interests include performance evaluation of communication networks, high speed switching and QoS on the Internet. He is a member of the IEICE, the IPS of Japan and the IEEE Communications society.

**Masaharu Komatsu** received B.E., M.E., and Dr. E. degrees in Communication Engineering from Osaka University, Osaka, Japan, in 1972, 1975, and 1978, respectively. From 1978 to 1987, he was an Assistant Professor at Hiroshima University, Hiroshima, Japan. From 1988 to 1991, he was an Associate Professor at Kanazawa University, Kanazawa, Japan. From 1991 to 2001, he was an Associate Professor at Osaka University, Osaka, Japan. Since April 2001, he has been a Professor in the Department of Information and Computer Engineering at Okayama University of Science, Okayama, Japan. His current interests include performance evaluation of communication systems, ATM switching and error control protocols. He is a member of the IPS of Japan and the IEEE.