

PAPER

A Pipelined Maximal-Sized Matching Scheme for High-Speed Input-Buffered Switches *

Eiji OKI[†], Roberto ROJAS-CESSA^{††}, *Members*, and H. Jonathan CHAO^{††}, *Nonmember*

SUMMARY This paper proposes an innovative Pipeline-based Maximal-sized Matching scheduling approach, called PMM, for input-buffered switches. It dramatically relaxes the limitation of a single time slot for completing a maximal matching into any number of time slots. In the PMM approach, arbitration is operated in a pipelined manner, where K subschedulers are used. Each subscheduler is allowed to take more than one time slot for its matching. Every time slot, one of the subschedulers provides the matching result. We adopt an extended version of Dual Round-Robin Matching (DRRM), called iterative DRRM (*i*DRRM), as a maximal matching algorithm in a subscheduler. PMM maximizes the efficiency of the adopted arbitration scheme by allowing sufficient time for the number of iterations. We show that PMM preserves 100% throughput under uniform traffic and fairness for best-effort traffic of the non-pipelined adopted algorithm, while ensuring that cells from the same virtual output queue (VOQ) are transmitted in sequence. In addition, we confirm that the delay performance of PMM is not significantly degraded by increasing the pipeline degree, or the number of subschedulers, when the number of outstanding requests for each subscheduler from a VOQ is limited to 1.

key words: Scheduling, pipeline, input-buffered switch, maximal-sized matching

1. Introduction

The explosion of Internet traffic has led to a greater need for high-speed switches and routers that have over 1-Tbit/s throughput [1]. Most high-speed packet switching systems adopt a fixed-size cell in the switch fabric. Variable-length packets are segmented into several fixed-sized cells when they arrive, are switched through the switch fabric, and are reassembled into packets before they depart.

There are various types of buffering strategies in switch architectures: input buffering, output buffering, or crosspoint buffering [2]–[6]. Input buffering is a cost-effective approach for high-speed switches. This is because input-buffered switches do not require internal speedup or allocation of buffers at each crosspoint. It relaxes memory-bandwidth and memory-size

constraints.

In input-buffered switches, it is well known that head-of-line (HOL) blocking limits the maximum throughput to 58.6% in a input-buffered switch with the First-In-First-Out (FIFO) structure [7]. A Virtual-Output-Queue (VOQ) structure is used to overcome the HOL-blocking problem [10]. Consider an $N \times N$ input-buffered switch with VOQs at the inputs and a crossbar switch fabric, as shown in Figure 1. A fixed-size cell is sent from any input to any output, provided that no more than one cell is sent from the same input and no more than one cell is received by the same output. Each input i has N VOQs, each of which is denoted as $VOQ(i, j)$, where cells that are destined for output j are stored. The HOL cell in each VOQ can be selected for transmission across the switch in each time slot. Therefore, every time slot, a scheduler has to determine one matching set.

For an input-buffered switch with VOQs, maximum-sized matching algorithms have been proposed to achieve 100% throughput [8], [9]. Although the maximum-sized matching algorithms provide a maximum match, they suffer from high-computing time complexity. Therefore, it is difficult to implement such algorithms for high-speed switching systems.

Maximal-sized matching algorithms have been proposed as an alternative to maximum matching ones, such as *i*SLIP [10] and Dual Round-Robin Matching (DRRM) [11], [12]. Both algorithms reduce their com-

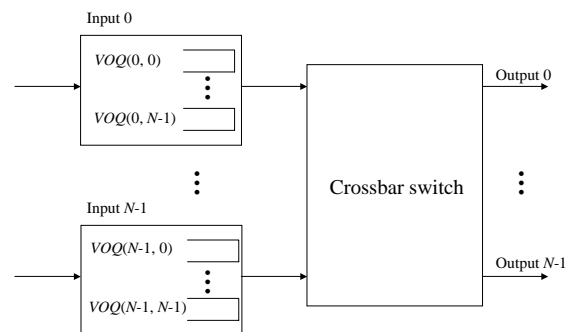


Fig. 1 Input-buffered switch with VOQs.

Manuscript received xx, 2001.

Manuscript revised xx, 2002.

[†]The author is with NTT Network Innovation Laboratories, NTT Corporation, Musashino-shi, 180-8585 Japan. This work was done while he was a Visiting Scholar at Polytechnic University, Brooklyn, NY.

^{††}The authors are with Polytechnic University, Department of Computer and Electrical Engineering, 6 Metrotech Center, Brooklyn, NY 11201 USA

*This research is supported by NSF Grant 9906673.

puting complexity compared with maximum matching ones, and provide 100% throughput under uniform traffic and complete fairness for best-effort traffic. However, they still have the strict constraint that the maximal matching has to be completed within one time slot. The constraint is a bottleneck when the switch size increases or a port speed becomes high, because the arbitration time becomes longer than one time slot or the time slot shrinks, respectively. Consider a 64-byte fixed-length cell at a port speed of 40Gbit/s (OC-768). The computation time given for maximal-sized matching is only 12.8ns.

To relax the strict scheduling timing constraint, a pipeline-based scheduling algorithm called Round-Robin Greedy Scheduling (RRGS) was proposed by Smiljanić et al. [13]. The RRGS switch has N round-robin arbiters, each of which is associated with an input. Each input performs one round-robin arbitration to select one VOQ in one time slot. N round-robin operations that select their own candidates to be transmitted at time slot T are performed during the previous N time slots $\{T-N, T-N+1, \dots, T-1\}$. Each round-robin operation is assigned into different time slots in a simple pre-determined cyclic manner so that RRGS can avoid output contention. (See Appendix A.)

However, RRGS is not able to provide max-min fair share for a best-effort service. Let $\lambda(i, j)$ and $\mu(i, j)$ be input offered load to $VOQ(i, j)$ and acceptable transmission rate from $VOQ(i, j)$, respectively. Consider a 3×3 switch, and $\lambda(0, 0) = \lambda(1, 0) = 1.0$ and other input offered loads $\lambda(i, j) = 0$, as shown in Figure 2. According to the RRGS algorithm described in [13], the acceptable transmission rate is obtained as $\mu(0, 0) = 2/3$ and $\mu(1, 0) = 1/3$. This is due to the simple cyclic allocation mechanism. The RRGS operation in this example is also described in Appendix A. Thus, when traffic is not balanced, some inputs can unfairly send more cells than others. Smiljanić also proposed weighted-RRGS (WRRGS), which guarantees pre-reserved bandwidth [14]. In WRRGS, however, the fairness problem is not yet solved for best-effort traffic. In addition, every N time-slot cycle, an idle time slot is produced when N is an even number. This means that RRGS does not efficiently utilize the switching capacity.

In addition, RRGS causes a fixed latency of N time slots at worst case. The average fixed latency is $N/2$ time slots. When the switch size is large, the latency will be the dominant delay. Thus, RRGS is not scalable in terms of the switch size.

It is a challenge to find a maximal matching scheduling scheme to meet the following requirements.

- The scheduling time should be relaxed into more than one time slot.
- High throughput should be provided.
- Fairness should be maintained for best-effort traf-

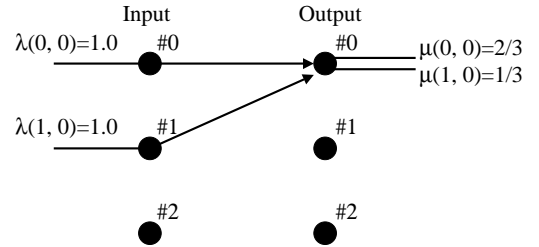


Fig. 2 Example of unfairness for RRGS under unbalanced traffic

fic.

- Cell sequence should be kept to avoid a cell resequence function at an output port.

To our knowledge, no study has been presented to satisfy these requirements.

This paper presents a solution to these requirements. We introduce an innovative Pipeline-based approach for Maximal-sized Matching scheduling in input-buffered switches, called PMM. Within the scheduler, more than one subscheduler operate in a pipelined manner. Each subscheduler is allowed to take more than one time slot. Every time slot, a subscheduler provides a matching result. The subschedulers can adopt a maximal matching algorithm such as *i*SLIP or DRRM, while preserving the properties of the non-pipelined versions. In this paper, we adopt an extended version of DRRM, called iterative DRRM (*i*DRRM), as a maximal matching algorithm in each subscheduler by considering the hardware implementation. *i*DRRM is easier to be implemented than *i*SLIP when round-robin arbiters are implemented in a distributed manner. As a result, PMM provides 100% throughput under uniform traffic, and maintains fairness for best-effort traffic. To use the advantage of the features of the adopted maximal matching algorithm, PMM allows one subscheduler to grant a younger request earlier than another subscheduler to grant an older request. However, PMM ensures that cells from the same VOQ are transmitted in sequence as only the HOL cell is transmitted when a grant is received. Therefore, PMM does not need a resequence function at an output port. We investigated the impact of the allowable number of outstanding requests distributed to subschedulers on the PMM delay performance. Furthermore, we show that the delay performance is not significantly degraded by increasing the number of subschedulers when the number of outstand-

ing requests for each subscheduler is limited to 1.

The remainder of this paper is organized as follows. Section 2 describes the PMM scheme. Section 3 describes the performance of the PMM scheme. Section 4 summarizes the key points.

2. Pipeline-Based Maximal-Sized Matching (PMM)

The PMM scheme is able to relax the computation time for maximal-sized matching into more than one time slot. The PMM scheduler consists of N^2 request counters and K subschedulers, as shown in Figure 3. Each subscheduler has N^2 request subcounters. Detailed notations in Figure 3 are described below. Each subscheduler operates the maximal-sized matching in a pipelined manner, and takes K time slots to complete the matching, as shown in Figure 4. We assume that PMM adopt an extended version of DRRM in each subscheduler.

DRRM was originally intended to use input and output round-robin arbitration with one iteration. The brief description of the DRRM algorithm is described in Appendix B. In the extended version of DRRM, the number of iterations are equal to or more than one. We call it iterative DRRM (i DRRM). In i DRRM, input and output round-robin arbitration is iterated to find matching input-output pairs among unmatched pairs. The input and output round-robin pointers are updated only when a grant is issued at the first iteration.[†] i DRRM mainly improves the delay performance of DRRM by considering more iterations.

From the hardware implementation point of view, i DRRM is easier to implement than i SLIP when input and output round-robin arbiters are distributed at the input/output ports [12], [15]. We describe the decentralized implementation comparison of DRRM and i SLIP and clarify why i DRRM is easier than i SLIP in Appendix C.^{††} When a scheduling algorithm is implemented in a decentralized manner, the amount of communication signals between printed circuit boards (PCBs) is sensitive due the limitation of the number of PCB pins and the transmission delay at a signal speed. In i SLIP, a request from $VOQ(i, j)$ at input port i is first sent to an output arbiter at output port j . On the other hand, i DRRM, a request from $VOQ(i, j)$ at input port i is first sent to an input arbiter at input port i . Thus, i SLIP first performs the round-robin arbitration for output ports and then does for input ports, while i DRRM first performs the round-robin arbitration for input ports and then does for output ports. As a result, i SLIP needs multiple requests and multiple

[†]This operation of i DRRM ensures max-min fair share as that of i SLIP.

^{††}Although Appendix C focuses on the non-pipelined schemes with one iteration for simplicity, the discussion is easily applied to the pipelined schemes with multiple iterations.

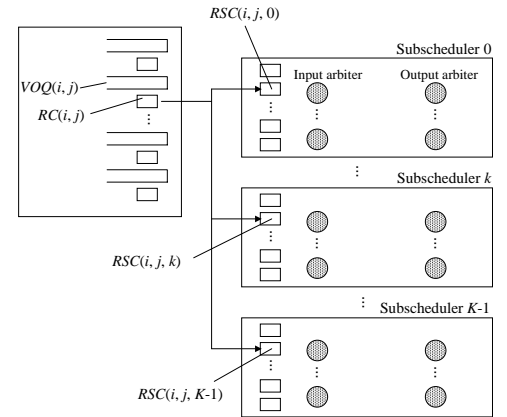


Fig. 3 Structure of PMM scheduler

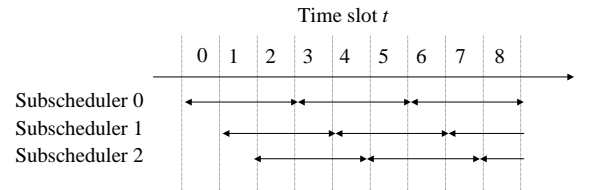


Fig. 4 Timing diagram of PMM with $K = 3$

grants between an input PCB and the crossbar PCB, while i DRRM needs only one request and one grant between them. In addition, i SLIP has an additional signal compared with i DRRM. Therefore, i DRRM has less communication signals between PCBs than i SLIP.

We note that i DRRM is logically equivalent to i SLIP, although the implementation of i DRRM is simpler than that of i SLIP [12]. The logical equivalence between i DRRM and i SLIP can be easily derived in the same ways as a logical equivalence between Parallel Iterative Matching (PIM) and Logical-Equivalent PIM (LE-PIM) was derived by Nong et al. in [16]. Let Θ be an input-traffic matrix. Each element $\theta(i, j)$ expresses input traffic to $VOQ(i, j)$. The performance of i DRRM with Θ is equivalent to that of i SLIP with Θ^T , where Θ^T is a transposed matrix of Θ .

Several notations used here are defined in the following. A request counter $RC(i, j)$ is associated with $VOQ(i, j)$, as shown in Figure 3. The value of $RC(i, j)$ is denoted as $C(i, j)$, where $0 \leq C(i, j) \leq L_{max}$. L_{max} is the maximum VOQ occupancy. $C(i, j)$ expresses the number of accumulated requests associated with $VOQ(i, j)$ that have not been sent to any subscheduler.

Each request subcounter $RSC(i, j, k)$ is associated with $VOQ(i, j)$ and subscheduler k , where $0 \leq k \leq K - 1$. The value of $RSC(i, j, k)$ is denoted as $SC(i, j, k)$, where $0 \leq SC(i, j, k) \leq SC_{max}$. SC_{max} has to be set carefully to provide the best performance. We found that when $SC_{max} = 1$, the delay performance is the best. The impact of SC_{max} will be discussed in detail in Section 3. $SC(i, j, k) > 0$ means that input i has at least one request for output j in subscheduler k . $SC(i, j, k) = 0$ means that input i has no request for output j in subscheduler k . At initial time, $C(i, j)$ and $SC(i, j, k)$ are set to zero.

PMM operates as follows.

- Phase 1: When a new cell enters $VOQ(i, j)$, the counter value of $RC(i, j)$ is increased: $C(i, j) = C(i, j) + 1$.
- Phase 2: At the beginning of every time slot t , if $C(i, j) > 0$ and $SC(i, j, k) < SC_{max}$, where $k = t \bmod K$, $C(i, j) = C(i, j) - 1$ and $SC(i, j, k) = SC(i, j, k) + 1$. Otherwise, $C(i, j)$ and $SC(i, j, k)$ are not changed.
- Phase 3: At $Kl + k \leq t < K(l+1) + k$, where l is an integer, subscheduler k operates the maximal-sized matching according to the adopted algorithm.
- Phase 4: By the end of every time slot t , subscheduler k , where $k = (t - (K - 1)) \bmod K$, completes the matching. When input-output pair (i, j) is matched, $SC(i, j, k) = SC(i, j, k) - 1$. The HOL cell in $VOQ(i, j)$ is sent to output j at the next time slot. This ensures that cells from the same VOQ are transmitted in sequence, even if $L(i, j) - C(i, j) > 1$, where $L(i, j)$ is the occupancy of $VOQ(i, j)$. Note that $L(i, j) - C(i, j) = \sum_{k=0}^{K-1} SC(i, j, k) \leq K \times SC_{max}$. In PMM, a request from $RC(i, j)$ belongs to only $VOQ(i, j)$, but not the HOL cell of $VOQ(i, j)$. Therefore, any request that is granted, independent of its age, is used by the HOL cell, thus cells are always transmitted in sequence. A VOQ can accumulate up to $K \times SC_{max}$ issued requests to the PMM scheduler. When input-output pair (i, j) is not matched, $SC(i, j, k)$ is not changed.

Whenever a condition associated with any phase is satisfied, the phase is executed by the main scheduler.

To apply the i DRRM algorithm as a matching algorithm in subscheduler k in PMM, we use $SC(i, j, k)$ instead of VOQ requests as described in the DRRM scheme [11], [12]. Each subscheduler has its own round-robin pointers. The position of pointers in subscheduler k is modified by the results only from subscheduler k . The operation of i DRRM in subscheduler k is the same as that of the non-pipelined i DRRM scheme. Therefore, it may happen that one subscheduler grants to a younger request earlier than another subscheduler

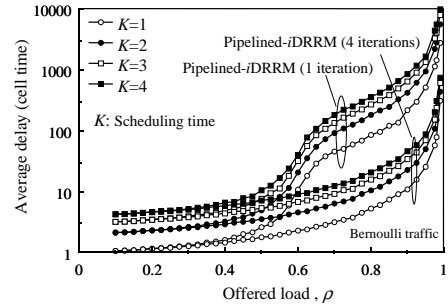


Fig. 5 Delay dependency on scheduling time K for Bernoulli traffic with $SC_{max} = L_{max}$ (switch size $N = 32$)

does to an older request, but PMM avoids the out-of-sequence problem. This is because a request is not related to a cell but to a VOQ in PMM.

3. Performance

This section describes delay performance and throughput of PMM under uniform traffic, and the fairness for best-effort traffic. In addition, the effect of the PMM scheduling relaxation is described.

3.1 Delay

First, we discuss the dependency of SC_{max} . Let us consider two extreme cases, which are $SC_{max} = L_{max}$ and $SC_{max} = 1$. Here, we assume that L_{max} is large enough to avoid cell loss at a VOQ to simplify the discussion.

Figures 5 and 6 show the average delay with $SC_{max} = L_{max}$ and $SC_{max} = 1$, respectively. Simulation results are obtained with a 95% confidence interval, not greater than 5% for the average cell delay. In this evaluation, we include absolute delay caused by the scheduling time in the delay performance.

These two figures show that delay performance degrades with $SC_{max} = L_{max}$, while it does not significantly degrade with $SC_{max} = 1$. Figure 7 shows the delay difference of $K > 1$ from that of $K = 1$. For example, with $\rho=0.95$, where ρ is the offered load, and $K = 4$, the delay difference of $SC_{max}=1$ is only 126 cell times, while that of $SC_{max} = L_{max}$ is 1155 cell times. When the input line speed is 10 Gbit/s, 126 cell times are equal to $6.45 \mu\text{s}$ when assuming a 64-byte cell. Thus, K does not affect the delay performance with $SC_{max} = 1$ for a practical use.

We explain why the delay performance of $SC_{max} = L_{max}$ is significantly degraded although that of

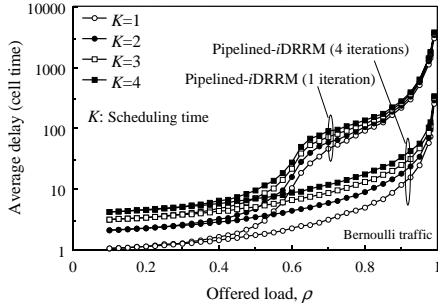


Fig. 6 Delay dependency on scheduling time K for Bernoulli traffic with $SC_{max} = 1$ (switch size $N = 32$)

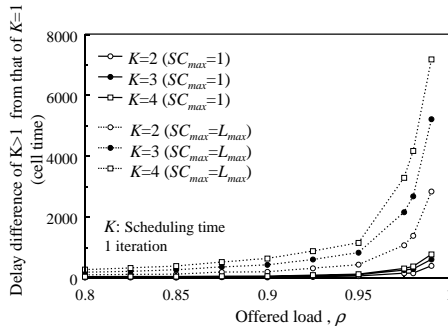


Fig. 7 Delay difference of $K > 1$ from that of $K = 1$ (switch size $N = 32$)

$SC_{max} = 1$ is not. When the value of SC_{max} is large, some subschedulers have more requests, while others have less requests. The large value of SC_{max} causes unbalance of the number of requests among subschedulers. The unbalanced request distribution makes delay performance worse.

Figures 8 and 9 show examples of counter status for $C(i, j)$ and $SC(i, j, k)$ with $SC_{max} = L_{max}$ and $SC_{max} = 1$, respectively. We consider $N = 3$ and $K = 2$. In both cases, $C(0, 0) = 1$, $C(0, 1) = 1$, and $C(0, 2) = 3$ are set and other counter values of $C(i, j)$ and $SC(i, j, k)$ are zero at the initial condition. We assume that no additional new cell enters a VOQ and all round-robin pointers point at zero. In these figures, $t = T$ (r) means a request phase, which is phase 2 as described in Section 2. $t = T$ (g) means a grant phase,

which is phase 4. The counter status follows the PMM algorithm as described in Section 2.

In Figure 8, at $t = 0$ (r), $C(0, 0)$, $C(0, 1)$, and $C(0, 2)$ are decremented and $SC(0, 0, 0)$, $SC(0, 1, 0)$, and $SC(0, 2, 0)$ are incremented. At $t = 1$ (r), $C(0, 2)$ is decremented to 1 and $SC(0, 2, 1)$ is incremented. At $t = 1$ (g), $SC(0, 0, 0)$ is granted by subscheduler 0, because the input round-robin pointer points at zero. The pointer is updated to 1. At $t = 2$ (r), $C(0, 2)$ is decremented and $SC(0, 2, 0)$ is incremented. At this time, all the values of $C(i, j)$ are zero. At $t = 2$ (g), $SC(0, 2, 1)$ is granted by subscheduler 1. $SC(0, 1, 0)$, $SC(0, 2, 0)$, and $SC(0, 2, 0)$ are granted by subscheduler 0 at $t = 3$ (g), $t = 5$ (g), and $t = 7$ (g), respectively. The main problem of $SC_{max} = L_{max}$ is that there is no request in subscheduler 1 from $t = 3$ (r), although subscheduler 0 has some requests until $t = 7$ (g).

On the other hand, in Figure 9, at $t = 2$ (r), $SC(0, 2, 0)$ is not fed by $C(0, 2)$. This is because $SC(0, 2, 0)$ is equal to $SC_{max} (= 1)$. $C(0, 2)$ is decremented and $SC(0, 2, 1)$ is incremented at $t = 3$ (r). At this time, the values of all $C(i, j)$ are zero. $SC(0, 0, 0)$, $SC(0, 2, 1)$, $SC(0, 1, 0)$, $SC(0, 2, 1)$, and $SC(0, 2, 0)$ are granted at $t = 1$ (g), $t = 2$ (g), $t = 3$ (g), $t = 4$ (g), and $t = 5$ (g), respectively. Thus, all requests of $SC_{max} = 1$ are granted two cell time slots earlier than that of $SC_{max} = L_{max}$, although the values of all $C(i, j)$ of $SC_{max} = 1$ are zero one time slot later than those of $SC_{max} = L_{max}$.

From these observations, when we set SC_{max} to 1, the requests from $RC(i, j)$ are distributed more effectively and the delay degradation is suppressed. When K increases, the requests from $RC(i, j)$ are dispersed more to each $RSC(i, j, k)$ associated with each subscheduler k . As a result, the desynchronization effect becomes less efficient as K increases. The delay degradation with $SC_{max} = 1$ is suppressed better than when $SC_{max} = L_{max}$ and, in this case, the delay is slightly degraded with K .

Since we know that when SC_{max} is set to 1, the delay performance is the best, we discuss the PMM performance with $SC_{max} = 1$ in the following.

In Figure 6, the delay with one iteration increases slightly at $0.65 \leq \rho < 0.9$. At the light offered load, $\rho < 0.5$, the contention among input requests is not significant. At $0.5 \leq \rho < 0.65$, the contention becomes significant. When ρ reaches 0.65, the pointer desynchronization is more easily obtained compared with the light offered load [10], [12]. However, when ρ exceeds 0.9, the contention among input requests significantly affects the delay performance.

The delay performance is improved with more iterations. Since PMM relaxes the scheduling timing constraint, a large number of iterations is not a bottleneck even when the switch size increases or a port speed becomes fast, compared with the non-pipelined algorithm, as will be described in Section 3.4. Note that

	Initail	$t=0$ (r)	$t=0$ (g)	$t=1$ (r)	$t=1$ (g)	$t=2$ (r)	$t=2$ (g)	$t=3$ (r)	$t=3$ (g)	$t=4$ (r)	$t=4$ (g)	$t=5$ (r)	$t=5$ (g)	$t=6$ (r)	$t=6$ (g)	$t=7$ (r)	$t=7$ (g)
$C(0,0)$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$C(0,1)$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$C(0,2)$	3	2	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
$SC(0,0,0)$	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
$SC(0,1,0)$	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$SC(0,2,0)$	0	1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
$SC(0,0,1)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$SC(0,1,1)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$SC(0,2,1)$	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0


 A request is granted.

Fig. 8 Example of counter status with $SC_{max} = L_{max}$

	Initail	$t=0$ (r)	$t=0$ (g)	$t=1$ (r)	$t=1$ (g)	$t=2$ (r)	$t=2$ (g)	$t=3$ (r)	$t=3$ (g)	$t=4$ (r)	$t=4$ (g)	$t=5$ (r)	$t=5$ (g)
$C(0,0)$	1	0	0	0	0	0	0	0	0	0	0	0	0
$C(0,1)$	1	0	0	0	0	0	0	0	0	0	0	0	0
$C(0,2)$	3	2	2	1	1	1	1	0	0	0	0	0	0
$SC(0,0,0)$	0	1	1	1	1	0	0	0	0	0	0	0	0
$SC(0,1,0)$	0	1	1	1	1	1	1	1	1	0	0	0	0
$SC(0,2,0)$	0	1	1	1	1	1	1	1	1	1	1	1	0
$SC(0,0,1)$	0	0	0	0	0	0	0	0	0	0	0	0	0
$SC(0,1,1)$	0	0	0	0	0	0	0	0	0	0	0	0	0
$SC(0,2,1)$	0	0	0	1	1	1	0	1	1	1	0	0	0


 A request is granted.

Fig. 9 Example of counter status with $SC_{max} = 1$

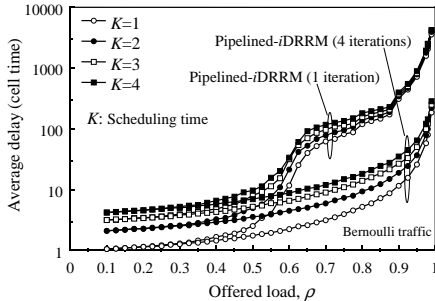


Fig. 10 Delay dependency on scheduling time K for Bernoulli traffic (switch size $N = 64$)

we showed the delay performance up to four iterations because there is no measurable improvement with more iterations.

The above observations for $N = 32$ are applied to different switch sizes. Figure 10 shows the same delay tendency as that in Figure 6 for a switch with $N = 64$.

We also consider an ON-OFF source model in which an arrival process to an input port alternates between ON (active) and OFF (idle) periods. A traffic source, during the ON period, continues sending cells in every time slot, but stops sending cells in the OFF period. Both the durations on the ON and OFF periods are assumed to be geometrically distributed. The average burst length used in this paper corresponds to the average ON period. In this evaluation, the average burst length is set to 10.

Figure 11 shows that K does not affect the delay performance for a practical use even when a bursty arrival process is considered. The dependency on K under bursty traffic is lighter than under Bernoulli traffic. This is because the arrival process of bursty traffic is less independent than that of Bernoulli traffic.

3.2 Throughput

As we saw the delay performance of PMM in Section 3.1, as PMM adopts the i DRRM algorithm, PMM provides 100% throughput under uniform traffic.

The reason is as follows. Consider the offered load as 1.0. If some inputs cannot send cells, outstanding requests are maintained in each subscheduler. In other words, $SC(i, j, k) = 1$. As a result, $C(i, j)$ is not decremented in phase 2 and increased in phase 1. Since $C(i, j)$ reaches a large enough value to be always satisfied with $C(i, j) > 0$, $SF(i, j, k) = 1$ is kept in phase 2.[†]

[†]Although $SC(i, j, k)$ becomes 0 in phase 4 when a re-

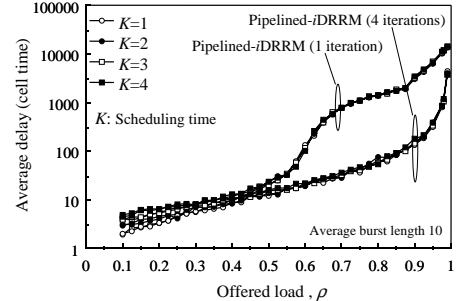


Fig. 11 Delay dependency on scheduling time K for bursty traffic (switch size $N = 32$)

In this situation with $SC(i, j, k) = 1$ at any t in phase 3, subscheduler k provides a complete matching result every K time slots due to the desynchronization effect of input and output round-robin arbiters, as described in [12], [17].^{††}

Thus, PMM preserves the throughput advantage of i DRRM.

3.3 Fairness

Since we adopt a round-robin-based algorithm in subscheduler k , subscheduler k can maintain a fair scheduling. Therefore, PMM provides max-min fair share for best-effort traffic.

3.4 Scheduling Timing Relaxation

Figure 12 shows the effect of the PMM scheduling timing relaxation. We assume that a cell size, L_{cell} , is 64×8 bits. Let the allowable arbitration time per iteration, a port speed, and the number of iterations be T_{arb} , C , and I , respectively. T_{arb} is given by,

$$T_{arb} = \frac{KL_{cell}}{CI}. \quad (1)$$

T_{arb} decreases with I and C , but increases with K . In the non-pipelined i DRRM scheme, K is 1 as a special case of PMM in Eq. (1). In the non-pipelined i DRRM, when $C=40\text{Gbit/s}$ and $I = 4$, $T_{arb}=3.2\text{ns}$. Under this timing constraint, it is difficult to implement a round-robin arbiter that supports a large N in hardware by

quest is granted, $SC(i, j, k)$ is always changed to 1 in phase 2.

^{††}We note that the pointer desynchronization is achieved within the same subscheduler. There is no relationship between pointers in different subschedulers.

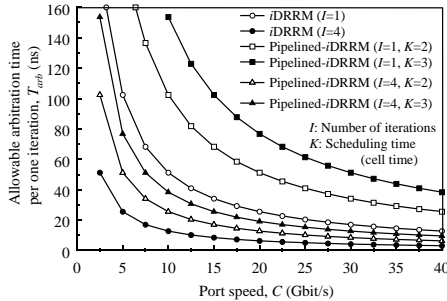


Fig. 12 Effect of PMM scheduling timing relaxation

using available CMOS technologies, in which, for example, a typical gate-delay time is about 100ps [18]. In this case, PMM can be used to expand T_{arb} by increasing K . When $C=40\text{Gbit/s}$, $I=4$ and $K=3$, $T_{arb}=9.6\text{ns}$. Therefore, PMM achieves the desired number of iterations even when N increases or C becomes fast.

4. Conclusions

This paper has proposed a Pipeline-based Maximal-sized Matching scheduling approach, called PMM, for input-buffered switches. It dramatically relaxes the scheduling timing constraint. Each subscheduler is allowed to take more than one time slot to complete a maximal-size matching. Every time slot, one subscheduler provides the matching results. The subschedulers adopt an extended version of DRRM, called iterative DRRM (*i*DRRM). *i*DRRM offers flexibility for implementation in a centralized or distributed manner. PMM maximizes the efficiency of the adopted arbitration scheme by allowing sufficient time for a number of iterations. We showed that PMM provides 100% throughput under uniform traffic and fairness for best-effort service as the non-pipelined algorithm does, while ensuring that cells from the same VOQ are transmitted in sequence. In addition, we investigated the impact of SC_{max} on the PMM delay performance. The delay performance is not significantly degraded by increasing the number of subschedulers when SC_{max} is set to 1.

The hardware in PMM is proportional to K , but PMM provides the flexibility to implement the schedulers in a distributed manner so that the increasing of hardware has a slight impact in the overall hardware amount.

Thus, PMM is a solution to a maximal-sized matching scheduler for input-buffered switches when the switch size increases or a port speed becomes high,

such as OC-768.

References

- [1] N. Yamanaka, E. Oki, S. Yasukawa, R. Kawano, and K. Okazaki, "OPTIMA: Scalable, Multi-Stage, 640-Gbit/s ATM Switching System Based on Advanced Electronic and Optical WDM Technologies," *IEICE Trans. Commun.*, vol. E83-B, no. 7, pp. 1488-1496, 2000.
- [2] G. Nong and M. Hamdi, "On the Provision of Quality-of-Service Guarantees for Input Queued Switches," *IEEE Commun. Mag.*, vol. 38, no. 12, pp.62-69, Dec. 2000.
- [3] E. Oki, N. Yamanaka, Y. Ohtomo, K. Okazaki, and R. Kawano, "A 10-Gb/s (1.25 Gb/s \times 8) 4×2 0.25- μm CMOS/SIMOX ATM Switch Based on Scalable Distributed Arbitration," *IEEE J. Solid-State Circuits*, vol. 34, no. 12, pp.1921-1934, Dec. 1999.
- [4] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined Input-One-Cell-Crosspoint Buffered Switch," *Proc. 2001 IEEE Workshop on High Performance Switching and Routing (HPSR)*, pp. 324-329, May 2001.
- [5] J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Switches," *IEICE Trans. Commun.* vol. E81-B, no. 2, pp. 120-137, Feb. 1998.
- [6] H. J. Chao, C. H. Lam, and E. Oki, *Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers*, Wiley, John & Sons, Inc., New York, 2001.
- [7] 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, 1987.
- [8] J. E. Hopcroft and R. M. Karp, "An Algorithm for Maximum Matching in Bipartite Graphs," *Soc. Ind. Appl. Math. J. Computation*, vol. 2, pp. 225-231, 1973.
- [9] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switches," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, Aug. 1999.
- [10] N. Mckeown, "The *i*SLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-200, Apr. 1999.
- [11] H. J. Chao and J.-S. Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM switch," *Proc. IEEE ATM Workshop'97*, Fairfax, VA, May 1998.
- [12] H. J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin," *IEEE Commun. Mag.*, vol. 38, no. 12 pp. 78-84, Dec. 2000.
- [13] A. Smiljanić, R. Fan, and G. Ramamurthy, "RRGS-Round-Robin Greedy Scheduling for Electronic/Optical Terabit Switches," *Proc. IEEE Globecom'99*, pp. 1244-1250, 1999.
- [14] A. Smiljanić, "Flexible Bandwidth Allocation in Terabit Packet Switches," *Proc. IEEE Workshop on High Performance Switching and Routing 2000*, pp. 233-239, 2000.
- [15] K. Genda, Y. Doi, K. Endo, T. Kawamura, and S. Sasaki, "A 160-Gb/s ATM Switching System using an Internal Speed-up Crossbar Switch," *Proc. IEEE GLOBECOM'94*, pp.123-133, 1994.
- [16] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of Nonblocking ATM Switches with Multiple Input Queues," *IEEE/ACM Trans. Networking*, vol. 7, no. 1, pp.60-74, Feb. 1999.
- [17] Y. Li, S. Panwar, and H. J. Chao, "On the performance of a dual round-robin switch," *Proc. IEEE INFOCOM 2001*, pp. 1688 -1697, 2001.
- [18] Texas Instruments, "GS40 0.15- μm CMOS, Standard Cell/Gate Array," <http://www.ti.com/>, version 0.2, May

T_0	T_1	T_2	T_3	T_4	T_5	T_6
		I_1 for T_5	I_2 for T_5	I_0 for T_5	I_1 for T_8	I_2 for T_8
	I_2 for T_4	I_0 for T_4	I_1 for T_4	I_2 for T_7	I_0 for T_7	I_1 for T_7
I_0 for T_3	I_1 for T_3	I_2 for T_3	I_0 for T_6	I_1 for T_6	I_2 for T_6	I_0 for T_9

Fig. A.1 Timing diagram of RRGs switch

2000.

Appendix A: Unfairness of RRGs

Before we explain why RRGs suffers from the unfairness problem, we describe briefly the RRGs algorithm. Figure A.1 shows an example of the RRGs algorithm. Let us consider a 3×3 switch. We denote the round-robin arbiter associated with input i as I_i . At time slot T_0 , I_0 chooses a candidate to be transmitted at time slot T_3 . At time slot T_1 , I_1 and I_2 choose their own candidates to be transmitted at time slot T_3 and T_4 , respectively. At time slot T_2 , I_2 , I_0 , and I_1 choose their own candidates to be transmitted at time slot T_3 , T_4 , and T_5 , respectively. In this way, at each time slot, each input arbiter chooses one candidate to be transmitted at different time slot from other arbiters. If we focus on a certain transmission time slot, for example, T_3 , the candidates are selected at T_0 , T_1 , and T_2 .

Next, we explain the unfairness problem of RRGs. We consider input traffic as shown in Figure 2. The results obtained by RRGs are shown in Figure A.2. At time slot T_0 , I_0 chooses $VOQ(0,0)$ as a candidate to be transmitted at time slot T_3 . Output 0 for T_3 is reserved. At time slot T_1 , I_1 and I_2 try to choose their own candidates to be transmitted at time slot T_3 and T_4 , respectively. However, I_1 cannot select any candidate for T_3 because $VOQ(1,0)$ is the only non-empty VOQ at input 1 and output 0 for T_3 is already reserved. I_2 cannot select any candidate for T_4 because all the VOQs at input 2 are empty. As a result, output 0 for T_4 is not reserved at T_1 . At time slot T_2 , I_0 and I_1 choose $VOQ(0,0)$ and $VOQ(1,0)$ as candidates to be transmitted at time slot T_4 and T_5 , respectively. As can be seen in Figure A.2, every three-time-slot cycle, $VOQ(0,0)$ is selected twice while $VOQ(1,0)$ is once. Thus, when traffic is not balanced, as in the example shown in Figure 2, input 0 can unfairly send twice more

T_0	T_1	T_2	T_3	T_4	T_5	T_6
		$VOQ(1,0)$ for T_5	<i>Not selected</i>	<i>Not selected</i>	$VOQ(1,0)$ for T_8	<i>Not selected</i>
	<i>Not selected</i>	$VOQ(0,0)$ for T_4	<i>Not selected</i>	<i>Not selected</i>	$VOQ(0,0)$ for T_7	<i>Not selected</i>
$VOQ(0,0)$ for T_3	<i>Not selected</i>	<i>Not selected</i>	$VOQ(0,0)$ for T_6	<i>Not selected</i>	<i>Not selected</i>	$VOQ(0,0)$ for T_9

Fig. A.2 Candidates selected by RRGs

cells than input 1.

Appendix B: DRRM algorithm

We briefly describe the DRRM algorithm. The more details are described in [11], [12]. Although the scheduling algorithm with one iteration is presented here, it is easy to apply it for i DRRM in the same way as i SLIP [9].

DRRM arbitration has four steps in cycle:

- Each input arbiter performs request selection.
- The input arbiters send their requests to the output arbiters.
- Each output arbiter performs grant arbitration.
- The output arbiters send grant signal to input arbiters.

Figure A.3 shows an example of the DRRM arbitration algorithm. In a request phase, each input chooses a VOQ and sends a request to an output arbiter. Assume input 0 has cells destined for outputs 0 and 1. Since its round-robin pointer is pointing 0, input arbiter 0 sends a request to output 0. Let us consider output 2 in the grant phase. Since its round-robin pointer is pointing to input 2, the output arbiter grants input 2. The input and output pointers are updated only when a grant is issued.

Similar to i SLIP, the DRRM scheme has the desynchronization effect. Consider the fully loaded situation in which every VOQ always has cells. Figure A.4 shows the head-of-line cells chosen from each input port in different time slots. In time slot 1, each input port chooses a cell destined for output A. Among those three cells, only one (the first in this example) is granted and the other two have to wait at HOL. The round-robin pointer of the first input advances to point to output B

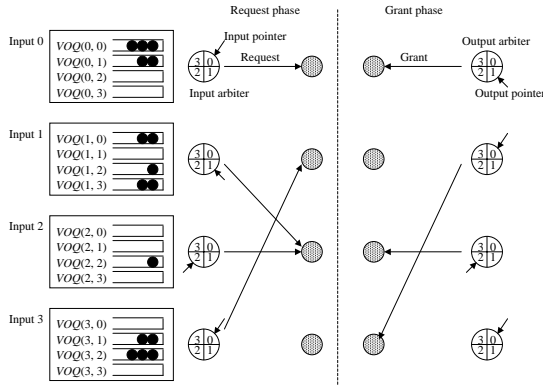
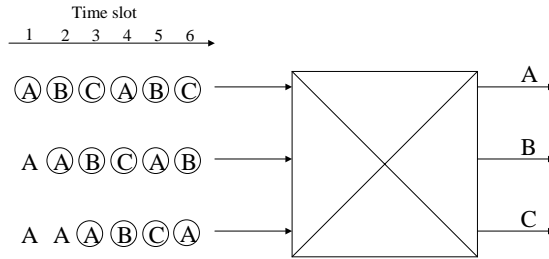


Fig. A·3 An example of the dual round-robin scheduling algorithm



The circle indicates that the cell is granted in the time slot.

Fig. A·4 The desynchronization effect of DRRM under the fully loaded situation

in time slot 2, and a cell destined for B is chosen and then granted because of no contenders. The other two inputs have their HOL cells unchanged, both destined for output A. Only one of them (the one from the second input) is granted; the other has to wait until the third time slot. At that time, the round-robin pointers among the three inputs have been desynchronized and point to C, B, and A, respectively. As a result, all three cells chosen are granted.

Appendix C: Decentralized implementation of DRRM and *i*SLIP

The *i*SLIP scheduler is implemented in a centralized manner [10]. However, when the switch size increases, the scheduler cannot be implemented in a single chip

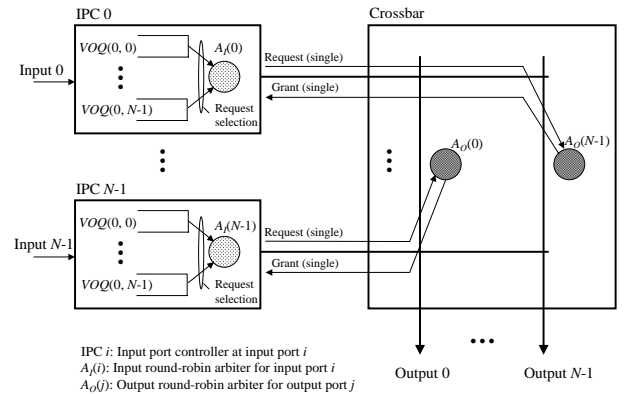


Fig. A·5 Decentralized implementation of DRRM

due the limitations of the gate and the pin number. Even if a multiple-chip implementation in a centralized manner is employed, the bottleneck of the number of pins in a chip still exists. The DRRM scheduler has also the same problem as *i*SLIP when it is implemented in a centralized manner. Therefore, a decentralized implementation should be adopted for a large-size switch.

We describe why DRRM is easier to implement than *i*SLIP when input and output round-robin arbiters are distributed at the input/output ports. Figures A·5 and A·6 show schematic examples of DRRM and *i*SLIP when they are implemented in a decentralized manner, respectively. We consider one iteration in these figures for simplicity.

In Figure A·5, input arbiter $A_I(i)$ for input port i is located in input port controller (IPC) i and output arbiter $A_O(j)$ for output port j is located in a crossbar part [12]. An example of the distributed round-robin arbiters was presented in [6], [15]. IPC i is usually implemented in a single PCB. In DRRM, $A_I(i)$ first selects one request out of several ones. IPC i sends the selected request to the crossbar. $A_O(j)$ selects one grant and the crossbar sends the grant to IPC i . These communications has to be done between different PCBs.

In Figure A·6, $A_I(i)$ and $A_O(j)$ of *i*SLIP are implemented in the same way as DRRM. In *i*SLIP, IPC i first sends multiple requests at most N to the crossbar. $A_O(j)$ selects one grant and the crossbar sends multiple grants up to N to IPC i . $A_I(i)$ accepts one grant and sends the accept to the crossbar.

Thus, *i*SLIP needs multiple requests and multiple grants between IPC i and the crossbar, while DRRM needs only one request and one grant between them. In addition, *i*SLIP has an additional accept signal compared with DRRM. Since DRRM has less communication signals between PCBs than *i*SLIP, DRRM is easier to implement in a decentralized manner than *i*SLIP.

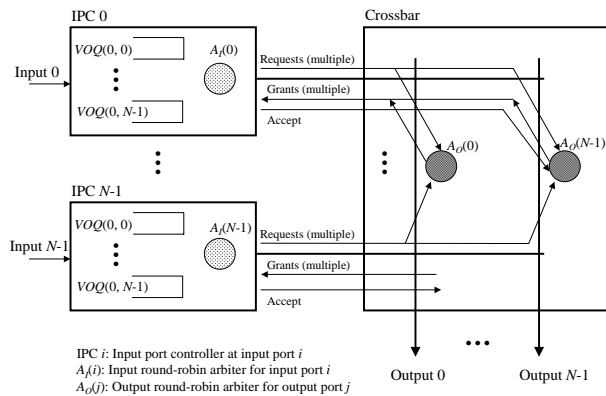


Fig. A-6 Decentralized implementation of iSLIP

Eiji Oki received B.E. and M.E. degrees in Instrumentation Engineering and a Ph.D. degree in Electrical Engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. In 1993, he joined Nippon Telegraph and Telephone Corporation's (NTT's) Communication Switching Laboratories, Tokyo, Japan. He has been researching multimedia-communication network architectures based on ATM techniques,

traffic-control methods, and high-speed switching systems in NTT Network Service Systems Laboratories. He was a Visiting Scholar at Polytechnic University, Brooklyn, New York, from 2000 to 2001. He is now engaged in researching and developing high-speed optical IP backbone networks as a Research Engineer with NTT Network Innovation Laboratories. Dr. Oki received the Switching System Research Award and the Excellent Paper Award from the IEICE in 1998 and 1999, respectively. He co-authored a book, "Broadband Packet Switching Technologies," published by John Wiley, New York, in 2001. He is a member of the IEICE and the IEEE.

Roberto Rojas-Cessa received his B.S. in Electronic Instrumentation from University of Veracruz, Mexico, in 1991. He received a microelectronics certificate from The Sappopo Electronics Center, Japan, in 1994. He joined UNITEC, Mexico City, from 1994 to 1996. He holds a M.S. degree in Electrical Engineering from CINVESTAV-IPN, Mexico City, in 1995 and a Ph. D. degree from Polytechnic University, Brooklyn, NY, in 2001. He

has been involved in VLSI design for biomedical applications and in research of packet scheduling schemes for high speed packet switches. He is currently a Postdoctoral fellow in Electrical Engineering at Polytechnic University. His research interests include high-speed and high performance switching, fault tolerance, and implementable scheduling algorithms for packet switches. He is a member of the IEEE and the IEICE. He was a CONACYT fellow.

H. Jonathan Chao is Professor of Electrical Engineering at Polytechnic University, New York, where he joined in January 1992. He has been doing research in the areas of terabit packet switches/routers and quality of service (QoS) control in IP/ATM/MPLS networks. He holds 19 patents with 5 more in pending and has published over 100 journal and conference papers in the above areas. He has also served as a consultant

for various companies, such as NEC, Lucent, and Telcordia, in the areas of ATM switch, packet scheduling, and MPLS traffic engineering. He has been giving short courses to industry in the subjects of the IP/ATM/SONET network over a decade. From 2000-2001, he was Co-Founder and CTO of Core Networks, NJ, where he led a team to implement a multi-terabit packet switch system with carrier-class reliability. From 1985 to 1992, he was a Member of Technical Staff at Telcordia, where he was involved in transport and switching system architecture designs and ASIC implementations, such as the first SONET-like Framing chip, ATM Layer chip, Sequencer chip (the first chip handling packet scheduling), and ATM switch chip. He received Telcordia Excellence Award in 1987. From 1977 to 1981, he was a senior engineer at Telecommunication Labs of Taiwan doing circuit designs for a digital telephone switching system. He is a Fellow of IEEE for his contributions to the architecture and application of VLSI circuits in high-speed packet networks. He is one of the recipients of the Best Paper Award of 2001 IEEE Transactions on Circuits and Systems for Video Technology. He co-authored two networking books, "Broadband Packet Switching Technologies," and "Quality of Service Control in High-Speed Networks," published by John Wiley in 2001. He served as a Guest Editor for IEEE Journal on Selected Areas in Communications (JSAC) with special topic on "Advances in ATM Switching Systems for B-ISDN" published in June 1997, and "Next Generation IP Switches and Routers" published in June 1999. He also served as an Editor for IEEE/ACM Transactions on Networking from 1997-2000. Dr. Chao holds B.S. and M.S. degrees in electrical engineering from National Chiao Tung University in Taiwan, and a Ph.D. in electrical engineering from Ohio State University.