# Chapter 1 Introduction

## 1.1 Problem Statement and Motivation

Packet switching is the core technology of the Internet. Internet link speeds are increasing to beyond tens of gigabits per second to satisfy a continuing growth in traffic. This increase in link speed and the need for quality of service (QoS) for diverse applications such as voice, audio, and video drive research into new switch architectures. Figure 1.1 abstracts the architecture of a generic switch. In general, a switch consists of four main components: 1) input ports, 2) output ports, 3) a switching fabric, and 4) an efficient scheduling algorithm and/or a service discipline for QoS requirements. The switching fabric interconnects input ports and output ports. The crossbar is a common switching fabric for high-speed switches used in the Internet because of its non-blocking capacity, simplicity, and market availability. The input port is the entry of input link. It receives incoming packets from the input link, classifies them according to their destination port. A scheduling policy decides the order in which packets at input ports are transferred across the switching fabric to the output ports. The goal of the scheduling policy is to maximize the throughput of the switch and provide QoS guarantees. At the output ports, the packets are received from switching fabric and buffered (if required) while they await to be transmitted to the outgoing link based on the service discipline.

Because two or more packets may compete simultaneously for the same output port, buffering within a switch is used for queuing of packets. Switch buffering can be in the input ports for input queued (IQ), in the output ports for output queued (OQ), and/or within the switching fabric. Combinations such as combined input and output queued (CIOQ) switches are possible.

For providing QoS guarantees in an integrated service network, the most important clause in QoS service contract made by clients and the network is the specifications of performance requirements and it includes: 1)the end-to-end delay bound, 2)the end-to-end delay jitter, 3)throughput, 4)and the loss probability. The end-to-end delay bound is essential for many applications that have stringent real-time requirements. The delay jitter is defined to be the maximum differences between delays experienced by any two packets. Having a bounded delay-jitter service from the network makes it possible for the destination to calculate the amount of buffer space needed to eliminate the jitter. The smaller the jitter bound, the less amount of buffer space is needed. Throughput guarantee is provided with the amount specified by the traffic characterization which negotiated by clients and the network. Packet loss can occur due to buffer overflown or delay bound violation. A statistic service [1], [2],[3] allows a non-zero loss probability while a deterministic service guarantees zero loss. A large number of service disciplines have been proposed to provide the QoS requirements, for examples: Delay Earliest-Due_Date (Delay-EDD) [4], [5], Virtual Clock [6], Fair Queuing (FQ) [7], and its weighted version (WFQ) also called Packetized Generalized Processor Sharing (PGPS) [8], Self-Clocked Fair Queuing (SCFQ) [9], Worst-case Fair Weighted Fair Queuing ($WF^2Q$) [10], Jitter Earliest-Due-Date (Jitter-EDD) [11], Stop-and-Go [12], Hierarchical Round Robin (HRR) [13], Rate-Controlled Static Priority (RCSP) [14], and a survey in [15]. Most of these algorithms were designed to be used at the output ports of an OQ switch and can be classified as either work-conserving or non-work-conserving. The service discipline is classified as a work-conserving scheme if an output never idles so long as there is a packet destined for it in the system. In an OQ switch, all packets coming to an input are immediately forwarded to the destination outputs as they arrive. Thus, it is ensures that the switch can achieve 100% throughput and it allows the departure of packets to be delivered to meet latency constraints by the service discipline and thus QoS is guaranteed. But the main problem of OQ switches is that the switching fabric of an $N \times N$ switch must run $N$ times as fast as its line rate in the worst case if packet loss is to be prevented, where $N$ denotes the number of input/output ports. The worst case occurs when $N$ input ports all simultaneously transfer a packet destined to the same output port. Since the recent developments in networking technology produced a dramatic growth of the line rates and thus made the speed requirements of switching fabric and memory accessing difficult to meet. As a result, OQ switches have serious scaling problem. A parallel packet switch (PPS) architecture, which is a variation of the switched connection inverse multiplex (SCIMUX) [16] architecture, is constructed with multiple identical lower speed OQ switches operating independently and in parallel was proposed to make a large-capacity switch with extremely high line-rates feasible [17]-[19]. It is possible that the lower speed OQ switch used in PPS can operate at a fraction of the line rate. However, it requires a centralized scheduling algorithm with unreasonable processing and communication complexity. Besides, the lower speed OQ switch is still a factor to set a limit on system capacity.

To alleviate memory bandwidth and the speedup requirement of the switching fabric, input queuing is widely considered in building a large capacity switch. In an IQ switch, packets arriving at input lines are all queued at input ports. Then the packets are extracted based on the scheduling algorithm from input queues to across the switching fabric and then be forwarded to the destined output ports. The switching fabric and the memory of an IQ switch needs only to run as fast as the line rate. However, IQ switches with a single first in first out (FIFO) buffer per input port have long been studied to have a limiting throughput of 58.6 percent for Bernoulli packet arrivals with uniformly randomly selected output port [20]. Moreover, it has been shown that the maximum throughput of the switch decreases monotonically with increasing burst size under bursty traffic arrivals [21]. This limited throughput is due to head-of-line (HOL) blocking in the input queues. In the single FIFO queue maintained at each input, a packet destined to an output that is free may be held up in line behind a packet that is waiting for an output that is busy. Many techniques have been suggested for reducing HOL blocking, for example by considering the first $K$ packets in the FIFO queue, where $K > 1$ [22]-[24]. Although these schemes can improve throughput, they are sensitive to traffic arrival patterns and may perform no better than regular FIFO queuing when the traffic is bursty. Another way of eliminating the HOL blocking is by changing the queuing structure at the input. Instead of maintaining a single FIFO queue at the input, a separate queue per each output can be maintained at each input. This scheme is called virtual output queuing (VOQ) and was first introduced in [25]. Each arriving packet is classified and then queued in the appropriate VOQ according to its determined destination output port. HOL blocking is eliminated because packets only queue behind packets that are destined to the same output; no packet can be held up by a packet ahead of it that is destined to a different output.

IQ switches that use VOQ's have been employed in a number of studies [26]-[30]. The key component for achieving high performance of these IQ switches is a scheduling algorithm that configures the switching fabric and decides which inputs will be connected to which outputs. The throughput can be improved to approach 100% if packets are delivered from input ports to output ports based on maximum matching algorithm for a bipartite graph problem [31]. There exist many algorithms for solving these problems, the most efficient currently known converges in computation complexity of O($N^{5/2}$) for uniform traffic [32] and O($N^3 logN$) for non-uniform traffic [33]. In addition to the high computation complexity, the maximum matching can lead to instability and unfairness when the traffic is admissible (particularly for non-uniform traffic patterns) and lead to starvation when the traffic is inadmissible [31]. The traffic is admissible if the aggregate arrival rate to an input or output port is less than 1. These disadvantages prohibit maximum matching from being used in a high-speed switch. To reduce the complexity, a large number of iterative maximal matching algorithms (e.g., PIM [27], WPIM[34], LPF [35], *i*SLIP [36], and FIRM [37], to name a few) have been proposed to achieve 100% throughput for IQ switches that use VOQ's when the traffic is addmissible. These algorithms need to be performed iteratively about $\log_2 N$ times to achieving high throughput and thus it is time consuming. In addition, none of these algorithms can match the performance of an OQ switch.

Another approach to improve the performance of an IQ switch closer to that of an OQ switch is to increase the speedup of the switching fabric. A switch is said to have a speedup *S*, if the switching fabric runs *S* times faster than each of the input or output line rates. Hence, an OQ switch has a speedup of *N*, while an IQ switch has a speedup of one. Because of speedup, an output port may receive packets faster than it can transmit. As a result, buffering at output ports is necessary and the switch becomes a combined input and output queued (CIOQ) switch while $1<S<N$.

The architecture of CIOQ switches has received considerable attention in the literature [38]-[41]. The goal is to find an efficient matching algorithm such that a CIOQ switch with a moderate speedup can exactly emulate an OQ switch. A CIOQ switch is said to exactly emulate an OQ switch if, under identical input traffics, the departure time of every packet from both switches is identical. Both analytical and simulation studies of a CIOQ switch which maintains a single FIFO queue at each input have been conducted for various values of the speedup [42]-[45]. A common conclusion of these studies is that with $S$ = 4 or 5 one can achieve about 99% throughput when arrivals are independent and identically distributed at each input, and the distribution of packet destinations is uniform across the outputs. In [46], the authors show that speedup of 4 is sufficient to ensure 100% asymptotic throughput with any maximal matching algorithm. Furthermore, an algorithm named the least output occupancy first algorithm (LOOFA) [47] was proposed for a CIOQ switch with a speedup factor of 2 to achieve 100% throughput. These algorithms basically perform maximal matching and thus are possible to realize in a high-speed switch. However, achieving 100% throughput is not sufficient for QoS guarantee. In [48], [49], it is proved that a CIOQ switch with a speedup factor of 2 can exactly emulate an OQ switch which employs a wide variety of service disciplines (e.g. WFQ and strict priority) if stable matching is adopted. Unfortunately, the complexity of stable matching is $O(N^2)$, and thus is impractical for a large system. In [50], the authors proposed a parallel maximal matching algorithm (of complexity $O(N)$), called the least cushion first/most urgent first (LCF/MUF) algorithm, and proved exact emulation of an OQ switch which can be achieved with a speedup factor of 2. There is no constraint on the service discipline used by the emulated OQ switch. The LCF/MUF algorithm simplifies the matching procedures and provides a variety of QoS guarantees. However, to implement the LCF/MUF algorithm, a switch has to know the cushion of all packets and their relative departure order in the emulated OQ switch. Besides, the complexity of cushion calculation and packet re-ordering required at the output ports make the algorithm very difficult to be realized in a high-speed network. In the first part of this thesis, we propose an approximate LCF/MUF algorithm and evaluate its performance via computer simulations for the weighted round robin (WRR) service discipline. The proposal simplifies cushion calculation and does not perform packet re-ordering at output ports. The tradeoff is that it loses the property of exact emulation. However, we found that the performance of a CIOQ switch using the proposed single-iteration approximate LCF/MUF algorithm is close to that of an OQ switch under uniform, non-uniform, and correlated input traffic models with fixed-length packets. Besides, the packet departure order is maintained for the WRR service discipline. Consequently, the re-ordering mechanism is not needed at output ports.

Furthermore, to exactly emulate OQ switches, the CIOQ switch is assume that there is always enough buffer space to store the packets when and where needed. Thus, packet drop due to insufficient buffer space never occurs, and all packet arriving to the switch eventually cross the switch. However, contrary to this setting it is observed empirically in the Internet that packets are routinely dropped in switches. Therefore, the performance of a CIOQ switch with limited input and output buffers is also presented.

As mentioned above, the selection of packets to be transmitted in each time slot from the input ports to output ports is accomplished using a scheduling algorithm, which is the key component in achieving high performance using a switch. Most of previous scheduling algorithms found in open literature were designed for fixed-length packets (or cells) scheduling. These proposed algorithms configure a set of non-conflict connections of the switching fabric between input ports and output ports in each time slot and re-configure all connections in the next slot. Hence, in current Internet environment, segmenting packets of variable-lengths into cells and re-assembling the cells into original packets are needed before and after packets transmissions across the switching fabric because cells belonging to various packets may interleave with one another. As a result, such a process can be time consuming and may be not be easily implemented at very high speed. Therefore, a scheduling algorithm designed for fixed-length packets has to be modified to make it useful for switching Internet packets. There are some algorithms [51]-[54] based on variable-length packet scheduling have been proposed for IQ switches. A variable-length packet will be scheduled as a whole and transmitted continuously in the switch rather than schedule them on a cell-by-cell basis. These algorithms were developed for IQ switches in achieving high throughput. But they are difficult to provide QoS guarantees. In the second part of this thesis, we present a variation of the approximate LCF/MUF algorithm for variable-length packet traffics and evaluate its performance via computer simulations again. We name it as packet-based LCF/MUF (PB-LCF/MUF) algorithm. Numerical results show that the performance of a CIOQ switch with a speedup factor of 5 which adopts the single iteration PB-LCF/MUF algorithm is close to that of an OQ switch under the WRR service discipline. The single iteration version not only can handle variable-length packets quite well but also avoid out-of-order packet delivery from input ports to output ports.

In order to provide QoS guarantee, a switch may have to maintain a large number of queues. However, when considering the cost of hardware implementation, the number of queues is a decisive factor. It is well known that IQ switches suffer from HOL blocking which limits the maximum throughput to 0.586 under uniform traffic and OQ switches have serious scalability problem. One possible approach to improve the performance of an IQ switch is to use virtual output queuing (VOQ) combined with a parallel matching algorithm such as *i*SLIP [36]. As a result, based on the VOQ structure, the number of queues maintained at each input port is $N \times K$ for an $N \times N$ switch with $K$ traffic classes which means high complexity when $N$ and $K$ are large. To reduce the complexity and make a system feasible, it is important to reduce the number of queues maintained by the switch. In the last part of this thesis, we propose a novel output initiated parallel matching (OIPM) algorithm for large-scale input buffered switches. In our proposed architecture, packets are buffered at input ports while queues are maintained at output ports. The number of queues maintained at each output port is equal to the number of traffic classes. It means that only $K$ queues maintained at each output port is needed under the traffics with $K$ priority classes which is much smaller than $N \times K$ virtual output queues maintained in an IQ switch. In addition to reducing the number of queues, our proposal guarantees starvation free, in-order packet delivery, and achieves weighted fairness among traffic classes.

## 1.2 Outline of Thesis

We will, in the next three chapters, present several scheduling algorithms that we have devised. It is the objective of each algorithm to match the set of inputs of a switch to the set of outputs more efficiently, fairly and feasibly in implementation. The rest of this thesis is organized as follows. In chapter 2, we propose the approximate LCF/MUF algorithm for a CIOQ switch and evaluate its performance under uniform, non-uniform, and correlated input traffic models with fixed-length packet. The performance of a CIOQ switch with finite input and output buffers is also presented. In chapter 3, a packet-based LCF/MUF matching algorithm was proposed for dealing with variable-length packets in the current Internet environment. Extensive simulations are performed for evaluating the performance of a CIOQ switch adopts the proposed PB-LCF/MUF algorithm. In chapter 4, a novel output initiated parallel matching algorithm was proposed for large-scale input buffered switches. The switch needs less number of queues and guarantees starvation free, in-order packet delivery, and achieves weighted fairness among traffic classes. Finally, we draw conclusions in chapter 5.

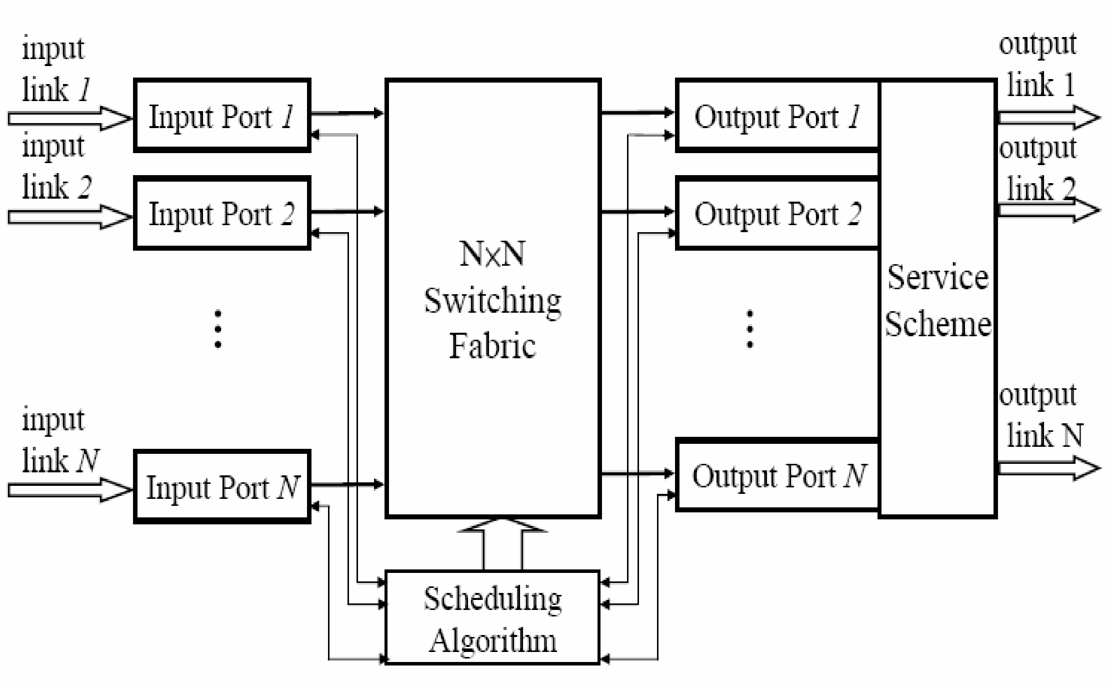


Fig. 1.1 The architecture of a generic switch

# Chapter 2

# The Approximate LCF/MUF Maximal Matching Algorithm

## 2.1 Introduction

It has recently been shown that a combined input and output queued (CIOQ) switch with a moderate speedup factor can exactly emulate an output queued (OQ) switch [47]-[50]. A CIOQ switch is said to exactly emulate an OQ switch if, under identical input traffics, the departure time of every packet from both switches is identical. The key component for exact emulation is a matching algorithm for bipartite graphs. In [46], the authors show that speedup of 4 is sufficient to ensure 100% asymptotic throughput with any maximal matching algorithm. Furthermore, an algorithm named the least output occupancy first algorithm (LOOFA) [47] was proposed for a CIOQ switch with a speedup factor of 2 to achieve 100% throughput. These algorithms basically perform maximal matching and thus are possible to realize in a high-speed switch. However, achieving 100% throughput is not sufficient for QoS guarantee. In [48], [49], it is proved that a CIOQ switch with a speedup factor of 2 can exactly emulate an OQ switch which employs a wide variety of service disciplines (e.g. WFQ and strict priority) if stable matching is adopted. Unfortunately, the complexity of stable matching is $O(N^2)$, and thus is impractical for a large system. In [50], the authors proposed a parallel maximal matching algorithm (of complexity $O(N)$), called the least cushion first / most urgent first (LCF/MUF) algorithm, and proved that a CIOQ switch with a speedup factor of 2 can exactly emulate an OQ switch with any arbitrary service discipline. The LCF/MUF algorithm simplifies the matching procedures and provides a variety of QoS guarantees. However, to implement the LCF/MUF algorithm, a switch has to know the cushion of all packets and their relative departure order in the emulated OQ switch. Besides, the complexities of cushion calculation and packet re-ordering required at the output ports make the algorithm very difficult to be realized in a high-speed switch. In this chapter, we propose an approximate LCF/MUF algorithm and evaluate its performance for the weighted round robin (WRR) service discipline adopted at output ports. For ease of implementation, the proposed algorithm calculates approximate cushions and does not perform re-ordering at the output ports. The tradeoff is that it loses the property of exact emulation. We found, via computer simulations, that the performance of a CIOQ switch with the proposed single-iteration matching algorithm is close to that of an OQ switch under uniform, non-uniform, and correlated input traffic models for offered load up to 0.9. Besides, the packet departure order can be maintained under the single-iteration algorithm.

Furthermore, to exactly emulate an OQ switch, the buffer size at each input and output port is assumed to be of infinite size. This assumption is obviously unrealistic in practice. Therefore, the performance of a CIOQ switch with finite input and output buffers is also presented in this chapter.

In the operation of the CIOQ switch, we assume that the switch handles fixed-length packets (or cells). Time is divided into slots such that the duration of a slot equals the time between cell arrivals at input ports. For simplicity, we assume that the link capacities of all input ports and output ports are equal. A CIOQ switch with a speedup factor of *S* is able to make *S* cell transfers from each input to outputs during each time slot. In other words, a slot is further divided into *S* equal sub-slots for the switching fabric and scheduling is performed at the beginning of each sub-slot.

## 2.2 The Least Cushion First / Most Urgent First (LCF/MUF) Algorithm

In this section, we briefly review the LCF/MUF algorithm proposed in [50]. Before describing the algorithm, we need the following definitions. Let $x_{i,j}$ denote a cell buffered at input port *i* destined to output port *j*.

**Definition 1：** The cushion of cell $x_{i,j}$ buffered at input port *i*, denoted by $C(x_{i,j})$, equals the number of cells currently residing in output port *j* which will depart the emulated *OQ* switch earlier than cell $x_{i,j}$.

**Definition 2：** The cushion between input port *i* and output port *j*, denoted by $C(i,j)$, is the minimum of $C(x_{i,j})$ of all cells buffered at input port *i* destined to same output port *j*. If there is no cell destined to output port *j*, then $C(i,j)$ is set to ∞.

**Definition 3：** The scheduling matrix of a $N \times N$ switch is a $N \times N$ square matrix whose $(i,j)^{th}$ entry equals $C(i,j)$.

The matching procedures made by the LCF/MUF algorithm are described below:

**Step 1.** Select the $(i,j)^{th}$ entry of the scheduling matrix which satisfies $C(i,j) = min_{k,l}\{C(k,l)\}$ (Least Cushion First). If the selected entry is ∞, then stop. If there are more than one entry with the least cushion residing in different columns, then choose the most urgent cell $x_{i,j}$ among those input ports which correspond to the selected entries (Most Urgent First). (The most urgent cell $x_{i,j}$ means the cell which will depart output port *j* in the emulated OQ switch earlier than any other cell currently residing in some input port. It is clear that urgency of a cell depends on the service discipline adopted at output ports.)

**Step 2.** Eliminate the $i^{th}$ row and the $j^{th}$ column (i.e., match output port *j* to input port *i*) of the scheduling matrix. If the reduced matrix becomes null, then stop. Otherwise, use the reduced matrix and go to *Step 1*.

It was proved in [50] that a CIOQ switch with a speedup factor of 2 that adopts the LCF/MUF matching algorithm can exactly emulate an OQ switch with any arbitrary service discipline. For convenience, the emulated OQ switch is referred to as the shadow OQ switch. As was pointed out before, it is very difficult to obtain the cushions of cells at all input ports for a complicated service discipline such as WFQ[7], PGPS[8] or WF$^2$Q[10]. As such, for high-speed switches, it is better to select a service scheduling algorithm which can provide QoS guarantee and is simple for cushion calculation. The WRR scheme satisfies the requirements and thus is studied in this chapter. We assume that output port *j* ($1 \leq j \leq N$) maintains *K* queues, denoted by $OQ_{j,k}$ for $1 \leq k \leq K$, with integral weights $w_1$, $w_2$, …, and $w_K$. The WRR service algorithm adopted at output ports for the shadow OQ switch is described below.

```
Step 1.  Let $k = 1$.
Step 2.  If $w_k > 0$, then
             $w_k = w_k - 1$;
             The HOL cell of output queue $OQ_{j,k}$ is served if queue $OQ_{j,k}$
             is non-empty.
             If $w_k = 0$ for all $k$, $1 \leq k \leq K$, then reset all the weights and go to
             Step 1.
Step 3.  $k = k + 1$ modulo $K$.   Go to Step 2.
```

Fig. 2.1 shows an example of the WRR service discipline with $K = 3$, $w_1 = 4$, $w_2 = 3$, and $w_3 = 1$. Both Figs. 2.1(a) and 2.1(b) show one cycle of the WRR scheme. In Fig. 2.1(a), every queue is non-empty when visited. In Fig. 2.1(b) queue 2 is empty in its second and third visits and thus the cycle is shorter than a regular one.

Note that the complexity of cushion calculation strongly depends on the service discipline. Several service disciplines have been proposed for QoS requirements. In [8], the authors demonstrated that, by employing generalized processor sharing (GPS) server at switches and leaky bucket admission control at the source, throughput and delay bound can be guaranteed to a connection. GPS has received much attention but it is impractical as it assumes that the server can serve all connections with non-empty queues simultaneously and that the traffic is infinitely divisible. In a more realistic packet system, only one connection can receive service at a time and an entire packet must be served before another packet is picked up for service. Consequently, there are many disciplines try to approximate GPS. The most well-known one is the weighted fair queuing (WFQ) discipline, also known as packet GPS (PGPS). Furthermore, the worst-case fair weighted fair queuing ($WF^2Q$) is proposed and shown that it provides almost identical service to GPS differing by no more than one maximum size packet when the packet departs the output port. Since both WFQ and $WF^2Q$ are defined with respect to the corresponding GPS system, a hypothetical fluid-flow reference GPS server is needed for specifying the order of services in the WFQ and WF$^2$Q scheme. However, it leads to considerable computational complexity, especially at high transmission speeds, due to the need for simulating events in the hypothetical GPS system. One simpler packet approximation algorithm of GPS is self-clocked fair queuing (SCFQ) discipline. The SCFQ scheme is based on the adoption of an internally generated virtual time as the index of work process instead of the hypothetical queuing system. The virtual time is simply extracted from the packet situated at the head of the queue. While simpler than WFQ and WF$^2$Q, SCFQ provides a much large delay bounded than that by WFQ and WF$^2$Q. Another related discipline is the virtual clock discipline [6]. Each arrival packet is allocated a virtual transmission time, which is the time at which the packet would have been transmitted in the switch. Packets are transmitted in the increasing order of virtual transmission times in the switch. The virtual transmission time of a packet is assigned based on the arrival pattern and the reservation rate of the connection which the packet belongs. Virtual clock discipline can provide the identical delay bound to a connection whose source is constrained by a leaky bucket. But the virtual transmission time of each packet in the switch is difficult to maintain and thus it is impractical in implementation.

Although these advanced service disciplines can provide diverse QoS applications, but these disciplines are difficult to implement. Besides, it is very difficult to obtain the cushions of packets at all input ports when a CIOQ switch uses out proposed matching algorithm and one of these complicated service disciplines.

## 2.3 The Approximate LCF/MUF Algorithm

The conceptual model of the considered CIOQ switch is illustrated in Fig. 2.2. Every input port maintains per output port, per priority queues. In other words, input port $i$ ($1 \le i \le N$) maintains a separate set of FIFO queues for output port $j$ ($1 \le j \le N$). As a consequence, there are $N$ sets of queues at each input port and each set consists of $K$ priority FIFO queues. The set of queues maintained at input port $i$ for output port $j$ are named as $VOQ_{i,j,k}$ for $1 \le k \le K$. A new arrival cell at input port is placed at the tail of the appropriate queue depending on its destination and priority. At output port *j,* there are *K* queues, denoted by $OQ_{j,k}$, which are served based on the WRR service scheme.

### 2.3.1 Single-iteration Matching

The proposed approximate LCF/MUF matching algorithm in the CIOQ switch is similar to the PIM algorithm [27] and has three phases in each iteration. In phase 1, input ports send requests to output ports. These requests contain the arrival times of the HOL cells. In phase 2, every output port computes the cushions for all cells contained in the requests, selects the least cushion cell, and then sends a grant message back to the corresponding input port. Finally, in phase 3, every input port which receives grant messages picks one based on the cushion and sends an accept message to the selected output port. To simplify calculation of cushions in phase 2, cells belonging to the same queue at an output port are not re-ordered, i.e., the queue is the same as a FIFO queue and cells are served in the order they arrive at the output port. Notice that the calculated cushion is just an approximation because, for exact cushion calculation, one has to know the departure times of the cells for the shadow OQ switch. Details of the matching algorithm are described below.

**Phase 1.** Input ports send requests to output ports.
In Phase 1, every input port *i* sends a request to each output port *j*. The request contains the arrival times of the HOL cells of all the *K* queues $VOQ_{i,j,k}$. The arrival time is set to infinity if a queue is empty.

**Phase 2.** Output ports send grants to input ports.
In Phase 2, every output port calculates the approximate cushions for the cells of each received request which was sent in Phase 1. The cushion of a cell is the number of time slots it has to wait before its service at the output port, assuming that it is delivered to the output port in the current time slot and there is no other arrival in the future. After the cushions of all cells are calculated, the output port selects the request with the least cushion and sends a grant message to the input port which sent

the request. If there are ties, then the cell with the earliest arrival time wins the contention. The grant message contains the calculated cushion of the request and the identification of the selected queue.

**Phase 3.** Input ports send accepts to output ports.
If an input port receives exactly one grant message from some output port, then it sends an accept message to the output port in Phase 3. In case an input port receives multiple grants, then it selects the grant with the least cushion. If there are ties, then the cell with the earliest arrival time is selected.

Fig. 2.3 shows an example of cushion calculation when the WRR service scheme is adopted at output ports. In this example, there are 3 queues at an output port with weights 4, 3, and 1. Note that the cushion of a cell depends on the state (i.e., the position in a service cycle) of the output port. In Fig. 2.3(a), the cushion of a request belonging to the second priority queue, denoted as $C(w_2)$, is equal to 4 as the current service token is at the first position of the service cycle. In Fig. 2.3(b), $C(w_2)$ = 2 as the current service token is at the fifth position.

Based on the outcome of the above matching procedure, cells are delivered from input ports to output ports. Remember that the switching fabric is speeded up by a factor of 2 and thus every input port can send at most 2 cells to output ports in a slot.

For ease of implementation, our scheme does not allow cell re-ordering at output ports. In fact, for the single-iteration scheme, it is not necessary to re-order cells at output ports. The reason is explained as follows. Consider two cells *x* and *y* of the same priority at different input ports. Assume that both *x* and *y* are HOL cells and destined to the same output port. If cell *x* arrived earlier than cell *y*, then the output port will select cell *x* in phase 2. Therefore, cell out-of-order will not happen at any output port.

### 2.3.2 Multiple-iteration Matching

To improve throughput performance of the switching fabric, the above matching procedure can be repeated for multiple iterations. After each iteration, an input port stops to send requests once it sends accept message to certain output port. Similarly, an output port stops to calculate cushions and send grant message once it receives an accept message. The subsequent iterations will try to match the rest input and output ports that remain unmatched in previous iterations. Obviously, the throughput increases as the number of iteration increases. However, multiple iterations may result in out-of-order delivery of cells and hurt the goal to emulate an OQ switch if cell re-ordering is not allowed at output ports. Our simulation results show that multiple-iteration matching does not yield a better performance than single-iteration matching.

## 2.4 Input Traffic Models

To quantitatively evaluate the performance of our considered CIOQ switch, we describe practical input traffic models that will be sent simultaneously to both the shadow OQ switch and the CIOQ switch in each simulation experiment. The following introduces three input traffic models — uniform, non-uniform and correlated models.

### 2.4.1 Uniform and Non-uniform Input Traffic

The uniform input traffic model is used to characterize the interactive behavior of data arrivals in computer networks [20]; there is no time correlation between arrivals. Cells arrival on the $N$ input links are governed by i.i.d. Bernoulli processes.

For non-uniform traffic, we use the same model as in [55]. We define non-uniform traffic by using an unbalance weight probability $T_w$. Let us consider input port $i$ and output port $j$. Given $\rho_i$, the offered input load for input port $i$, the traffic load from input port $i$ to output port $j$, denoted by $\rho_{i,j}$, is given by

$$\rho_{i,j} = \begin{cases} \rho_i (T_w + \dfrac{1-T_w}{N}) & \text{if } i = j \\ \rho_i \dfrac{1-T_w}{N} & \text{otherwise} \end{cases} \qquad (1)$$

When $T_w$=0, the offered traffic is uniform. On the other hand, when $T_w$ =1, the traffic is completely unbalanced in the sense that all the traffic of input port $i$ is destined for output port $i$.

## 2.4.2 Correlated Input Traffic

The correlated traffic model we studied is characterized by a 2-states Markov process alternating between active and idle states with probabilities $p$ and $q$, respectively [21][56] (see Fig. 2.4). This model is often used to describe IP packets fragmented into ATM cells. Based on this model, the traffic source will generate a cell every slot when it is in the active state. Note that there is at least one cell in a burst. Define random variable $B$ as the number of time slots that the active period (burst) lasts. The probability which the active period lasts for a duration of $m$ time slots (consists of $m$ cells in a burst) is

$$\Pr[B = m] = p(1-p)^{m-1}, \qquad m \geq 1 \qquad (2)$$

The mean burst length is given by

$$E[B] = \sum_{m=1}^{\infty} m \cdot \Pr[B = m] = 1/p \qquad (3)$$

Similarly, define random variable $I$ as the number of time slots the idle period lasts. The probability that an idle period lasts for $n$ time slots is

$$\Pr[I = n] = q(1-q)^{n}, \qquad n \geq 0 \qquad (4)$$

and the mean idle period duration is given by

$$E[I] = \sum_{n=1}^{\infty} n \cdot \Pr[I = n] = (1-q)/q \tag{5}$$

Given $p$ and $q$, the offered traffic load $\rho$ can be found by

$$\rho = \frac{E[B]}{E[B] + E[I]} = \frac{q}{p + q - pq} \tag{6}$$

We assume there is no correlation between different bursts and the destination of each burst is uniformly distributed among the output ports.

## 2.5 Numerical Results

To evaluate the performance of the CIOQ switch using the proposed matching algorithm, we measure the latency of cells which are simultaneously fed to both the shadow OQ switch and the CIOQ switch under uniform, non-uniform, and correlated traffic models. We define $d(x) = | d_{OQ}(x) - d_{CIOQ}(x) |$ as the deviation index of cell $x$. Here $d_{OQ}(x)$ and $d_{CIOQ}(x)$ denote, respectively, the departure times of cell $x$ for the shadow OQ switch and the CIOQ switch. Given a fixed $d$, we measure the percentage of cells that has a deviation index smaller than or equal to $d$. This percentage is denoted by $P_d$. For example, $P_{d=0}$ equals 100% means that the CIOQ switch exactly emulates the shadow OQ switch.

Our simulation experiments are divided into five parts. In the first three parts, we measure the values of $P_d$ under uniform, non-uniform, and correlated traffic models, respectively. We found that the performance of a CIOQ switch using the proposed matching algorithm is close to that of an OQ switch. In the fourth part, we study the effect of multiple-iteration matching. Results show that single-iteration matching is good enough and thus cell re-ordering is not needed. In the last part, we change the speedup factor and show that a CIOQ switch with a speedup factor of 2 yields much better performance than one without speedup. Besides, a speedup factor larger than 2 is not necessary. Our simulation results also show the performance of the CIOQ switch, with a speedup of 2, using the FIRM algorithm for comparison. It will be seen that the LCF/MUF algorithm has better performance under these three traffic models. We performed simulations for the CIOQ switch using the *i*SLIP algorithm. The performance of *i*SLIP is worse than that of FIRM and is not shown in this thesis.

Simulations are performed for $N$ = 8, 16, and 32 for 3 priority levels with weights $w_1 = 4$, $w_2 = 3$, and $w_3 = 1$, under uniform, non-uniform, and correlated traffic models. For correlated traffic, the mean burst length $\ell$ = 8, 16, 32, and 64 are considered. In each experiment, the statistics of about one million cells over all queues at input ports are collected

### 2.5.1 Performance under Uniform Traffic

For uniform input traffic, simulation results are shown in Fig. 2.5 and Fig. 2.6. For switch size $N = 16$ and the switch adopts the proposed LCF/MUF algorithm, Fig. 2.5 shows that the values of $P_{d=0}$ are larger than 0.866 for all the three priority traffics under an offered load up to 0.9 and the values of $P_{d\leq 2}$ are close to 100%. The curves also show that the performances of all the three priority traffics are similar. In Fig. 2.6, the curves show the performances of the CIOQ switch for various switch sizes. It can be seen that the performance degrades as the number of ports increases. But it degrades slowly when $N$ is larger than 16. Based on these results, we conclude that the performance of a CIOQ switch with the proposed matching algorithm is close to that of an OQ switch under uniform traffic.

Fig. 2.5 and Fig. 2.6 also show that the performances of the FIRM algorithm are worse than those of the LCF/MUF algorithm. The difference is significant under heavy load conditions. To avoid drawing too many lines to make these figures less comprehensible, the curves of $P_{d\leq 2}$ for the FIRM algorithm are not shown. And the curves for the 2nd and 3rd priority traffic cells are not shown in Fig. 2.6. Our results show that $P_{d\leq 2}$ for the FIRM algorithm is smaller than that for the LCF/MUF algorithm. And the performances of the 2nd and 3rd priority traffic cells are similar to those of the 1st priority traffic cells in Fig. 2.6. Notice that cushion calculation helps the LCF/MUF algorithm to achieve a better performance than the FIRM algorithm.

Moreover, the FIRM algorithm may result in out-of-sequence delivery that also hurts its performance if re-ordering is not allowed at the output ports.

### 2.5.2 Performance under Non-uniform Traffic

For non-uniform input traffic, simulation results are shown in Fig. 2.7 and Fig. 2.8 for switch size $N$ = 16 which adopts the proposed LCF/MUF algorithm and the FIRM algorithm respectively. When $T_w$ = 0, the offered traffic is uniform and thus the values of $P_{d=0}$ are the same as those shown in Fig. 2.5. The value of $P_{d=0}$ increases and is equal to 100% as $T_w$ increases to one. The reason is that, when $T_w$ = 1, output port $j$ is always matched to input port $j$ for all $j$.

### 2.5.3 Performance under Correlated Traffic

For the correlated traffic model, results depend on the mean burst length $\ell$. In Fig. 2.9, we show the results for $\ell$ = 16 cells and $N$ = 16. When the switch adopts the proposed algorithm, the values of $P_{d=0}$ are larger than 0.875 for all the three priority traffic under an offered load up to 0.9 and the values of $P_{d\leq 2}$ are close to 100%. The performances of all the three priority traffics are similar under both uniform and correlated traffic models. In Fig. 2.10, the curves show the performances of the CIOQ switch for various switch sizes. It can be seen that the performance degrades as the number of ports increases. Again, it degrades slowly when $N$ is larger than 16. In Fig. 2.11 we performed similar simulations for others values of $\ell$. It can be seen that the performance degrades as $\ell$ increases. But the degradation becomes slow once $\ell$ is larger than 16. Again, to avoid drawing too many lines to make these figures less comprehensible, the curves of $P_{d\leq 2}$ for the FIRM algorithm are not shown. And the curves for the 2nd and 3rd priority traffic cells are not shown in Fig. 2.10 and Fig. 2.11. Based on these results, we conclude that the performance of a CIOQ switch with the proposed matching algorithm is close to that of an OQ switch under correlated traffic.

### 2.5.4 Effects of Multiple-iteration Matching

In this sub-section, we study the effects of multiple-iteration matching. In our experiments, the algorithm is performed with three and five iterations. The results are shown in Fig. 2.12. As mentioned before, multiple-iteration matching does not improve the performance in terms of emulation of a shadow OQ switch. Therefore, we conclude that single-iteration matching is enough for the proposed approximate LCF/MUF algorithm.

### 2.5.5 Effects of Speedup Factors

In this sub-section, we study the effects of speedup factors. In Fig. 2.13, we show the results for $\ell$ = 16 cells and $N$ = 16 for speedup factor $S$ = 1, 2, and 3. It can be seen that, for $S$ = 1 which means no speedup, system performance degrades seriously as traffic load increases. The reason is that the approximate LCF/MUF is not a maximum matching algorithm. The performance can be improved if the matching is performed for multiple iterations. However, the improvement is not obvious. As illustrated in the figure, the results for $S$ = 3 are almost the same as those for $S$ = 2. We also performed simulations for $S$ = 4 and 5 and the results are similar. So, a speedup factor of 2 is important to achieve satisfactory performance and a speedup factor greater than 2 is not necessary.

## 2.6 CIOQ switches with finite input and output buffers

In [50], the authors prove that a CIOQ switch with a speedup factor of 2 can exactly emulate an OQ switch adopts any arbitrary service discipline. The key component is an efficient matching algorithm named as the LCF/MUF algorithm. Besides, the CIOQ switch is assume that there is always enough buffer space to store the packets when and where needed. Thus, packet drop due to insufficient buffer space never occurs, and all packets arriving to the switch eventually cross the switch. However, contrary to this setting it is observed empirically in the Internet that packets are routinely dropped in switches. In this section, we investigate via computer simulations the performance of a CIOQ switch with limited input and output buffers. To implement the LCF/MUF algorithm, a switch has to know the cushion of all cells and the relative departure order of cells destined to the same output port. The complexity of cushion calculation strongly depends on service discipline. For ease of simulation and feasibility in implementation, we select the strict priority scheme as the service discipline of the shadow OQ switch. We assume that every output port $j$ maintains $K$ FIFO queues named as $Q_{j,k}$ with priority $1$ to priority $K$ respectively (priority $1$ has the highest priority). At each output port $j$, the strict priority service discipline is described below.

```
for $k=1$ to $K$ {
    if queue $Q_{j,k}$ is non-empty    then
        the head-of-line cell of $Q_{j,k}$ is served and break for-loop.
}
```

In our study, the input traffic is characterized by uniform or correlated model as mentioned before. With finite input and output buffers, the property of exact emulation is lost. However, simulation results show that, under uniform traffic, a CIOQ switch behaves almost like an OQ switch if the buffer size at every input port and every output port are 3 and 9 cells respectively. For correlated traffic, to achieve similar results, both input and output buffer sizes have to be increased to about 7 and 11 times of the mean burst size, respectively.

Our simulation experiments are divided into three parts. In the first two parts, we measure the latency of cells which are simultaneously fed to both the shadow OQ switch and the CIOQ switch with infinite input buffer and finite output buffer under uniform and correlated traffic models, respectively. We use $P_d$ mentioned before as the criteria of the performance. For example, $P_{d=0}$ equals 100% means that the CIOQ switch exactly emulates the shadow OQ switch. In the third part, we study the effect of finite input buffer. We measure the cell-loss probability (denoted by $P_{loss}$) of a CIOQ switch with finite input buffer and finite output buffer.

Simulations are performed for $N$ = 4, 8, 16, and 32, with 4 priority levels under both uniform and correlated traffic models. For correlated traffic, the mean burst length $\ell$ = 4, 8, 16, 32, and 64 are considered at the same time. The statistics of about one million cells are collected (over all queues in the switch) and thus loss probabilities smaller than $10^{-6}$ are not measurable.

### 2.6.1 Results of Uniform Input Traffic

For uniform input traffic load, simulation results are shown in Fig. 2.14 to Fig. 2.16. Fig. 2.14 shows that the CIOQ switch almost behaves like an OQ switch under moderate loads if sufficiently many output buffers are installed. For $N$ =16 with infinite input buffer ($B^i = \infty$) and finite output buffer $B^o$ = 9 cells, the value of $P_{d=0}$ is larger than 90% for the 1st priority traffic cells under an offered load up to 0.8. Fig. 2.15 shows the values of $P_{d=0}$ and $P_{d<=2}$ for all the four priority cells. We observe that the performance behaves similarly for all the four priority cells and the values of $P_{d<=2}$ are close to 100% even under an offered load up to 0.8. As is shown in Fig. 2.16, the performance degrades as the number of ports increases. But it degrades slowly when $N$ is larger than 8. Based on these results, we conclude that a CIOQ switch performs like an OQ switch if output buffer size is at least 9 cells.

Note that, the maximum queue length at all input ports is also estimated in these simulations. The input buffer size of 60 cells is sufficient under the uniform traffic model.

### 2.6.2 Results of Correlated Input Traffic

For the correlated traffic model, results depend on the mean burst size $\ell$. In Fig. 2.17, we show the results for $\ell$ = 16 cells for $N$ = 16 with $B^o$ = 5 $\ell$, 7 $\ell$, 9 $\ell$, 11 $\ell$ and 13 $\ell$ cells. The value of $P_{d=0}$ is larger than 90% for the 1st priority traffic under an offered load up to 0.8 when $B^o$ = 11 $\ell$ cells are installed. From Fig. 2.18, one can see that the performances are roughly the same for various switch sizes. In Fig. 2.19 we performed similar simulations for other values of $\ell$. Our observation is that the performance degrades as a function of the value of $\ell$. But the degradation is became slow once $\ell$ is larger than 16. Based on these results, we suggest to allocate $B^o = 11\ell$ cells at each output port for correlated traffic.

Note that the maximum queue length at all input ports is smaller than 237 cells (about $15\ell$ cells) which is also estimated in simulations.

### 2.6.3 Effect of Finite Input Buffer

In this sub-section, we study the effect of finite input buffer. The output buffer size is chosen to be 9 cells for uniform traffic and $11\ell$ cells for correlated traffic. The switch size and the mean burst length investigated in our simulations are $N = 16$ and $\ell = 16$. We measure cell loss probability ($P_{loss}$) due to finite input buffer. Fig. 2.20 shows the curves of $P_{loss}$ versus $\rho$ for input buffer size $B^i = 2$, 3, 5, 7 and 9 cells under uniform traffic. Fig. 2.21 shows the curve of $P_{loss}$ versus $\rho$ for input buffer size $B^i = 3\ell$, $5\ell$, $7\ell$, $9\ell$ and $11\ell$ cells under correlated traffic. It can be seen that $P_{loss}$ of the CIOQ switch with $B^i = 3$ and $B^o = 9$ is smaller than $10^{-5}$ under the uniform traffic model for an offered load up to 0.8. For correlated input traffic, one can achieve similar performance with $B^i = 7\ell$ and $B^o = 11\ell$.

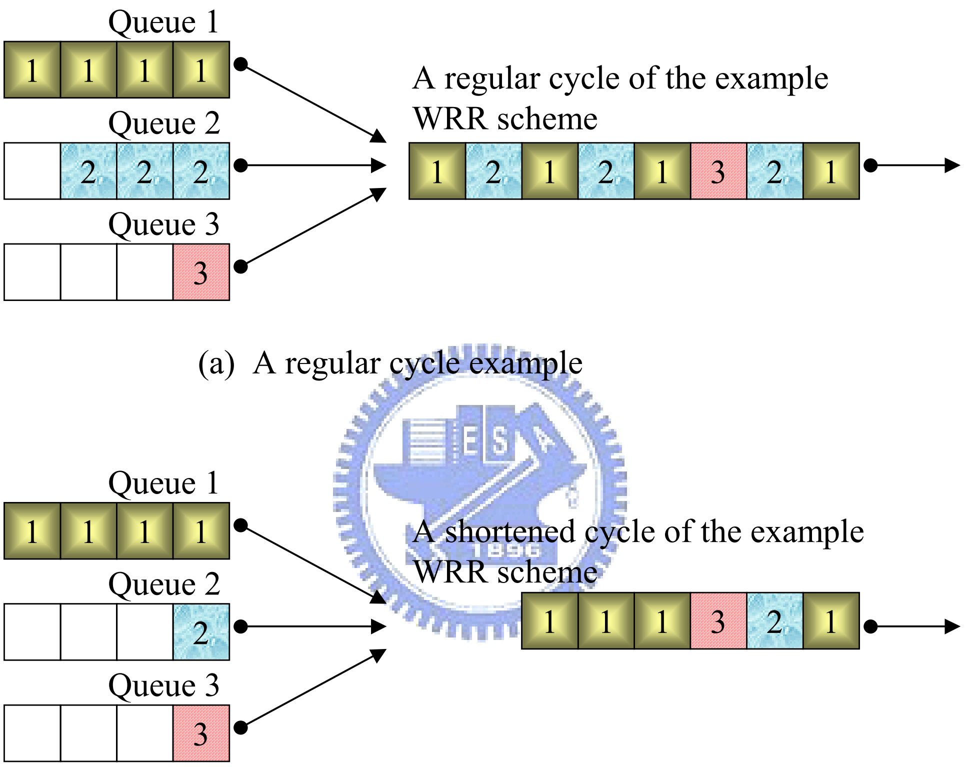
Note that, based on our simulation results, the CIOQ switch needs to maintain a buffer of 60 cells at each input port to achieve zero cell loss probability under uniform traffic (or $15\ell$ cells under correlated traffic) for an offered load up to 0.9.

## 2.7 Summary

We have proposed in this chapter an approximate LCF/MUF matching algorithm. Our proposal makes the LCF/MUF algorithm much more feasible for implementation because it needs only single-iteration matching and calculation of cushions of cells is largely simplified. Numerical results show that the performance of the proposed approximate matching is very close to that of the LCF/MUF algorithm under uniform, non-uniform, and correlated traffic models for offered load up to 0.9. Our simulation results show that the performance of the proposed approximate LCF/MUF algorithm is significantly better than that of a simple matching algorithm such as FIRM or *i*SLIP. The price paid for the LCF/MUF algorithm is higher hardware cost because it requires cushion calculation.

We have also investigated the performance of a CIOQ switch with finite input and output buffers. In this study, we choose strict priority service discipline because it allows cushions to be easily calculated. Based on simulation results, we found that a CIOQ switch with sufficient input and output buffers can mimic an OQ switch quite well for both uniform and correlated traffics.

Since memory bandwidth sets a limit on switch capacity, CIOQ switch is likely to be a more suitable choice than OQ switch in building a large-capacity switch. Our proposal presented in this chapter reduces the implementation complexity of a CIOQ switch and thus hopefully is useful for building large-capacity switches with QoS guarantee.



(a) A regular cycle example

(b) A shortened cycle example

Fig. 2.1 Examples of the WRR service algorithm with $K = 3$, $w_1 = 4$, $w_2 = 3$, and $w_3 = 1$.

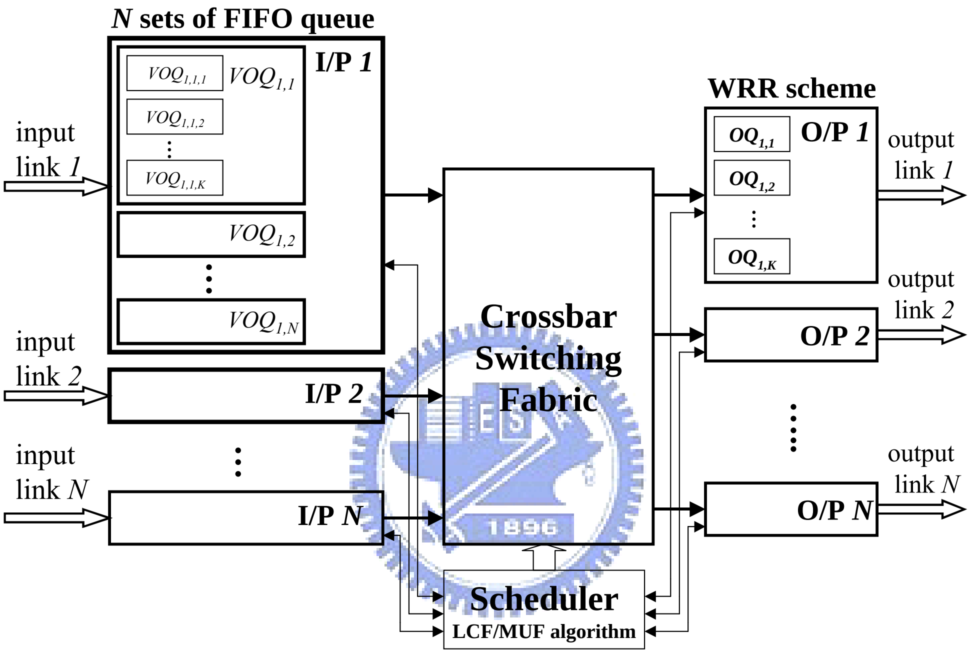
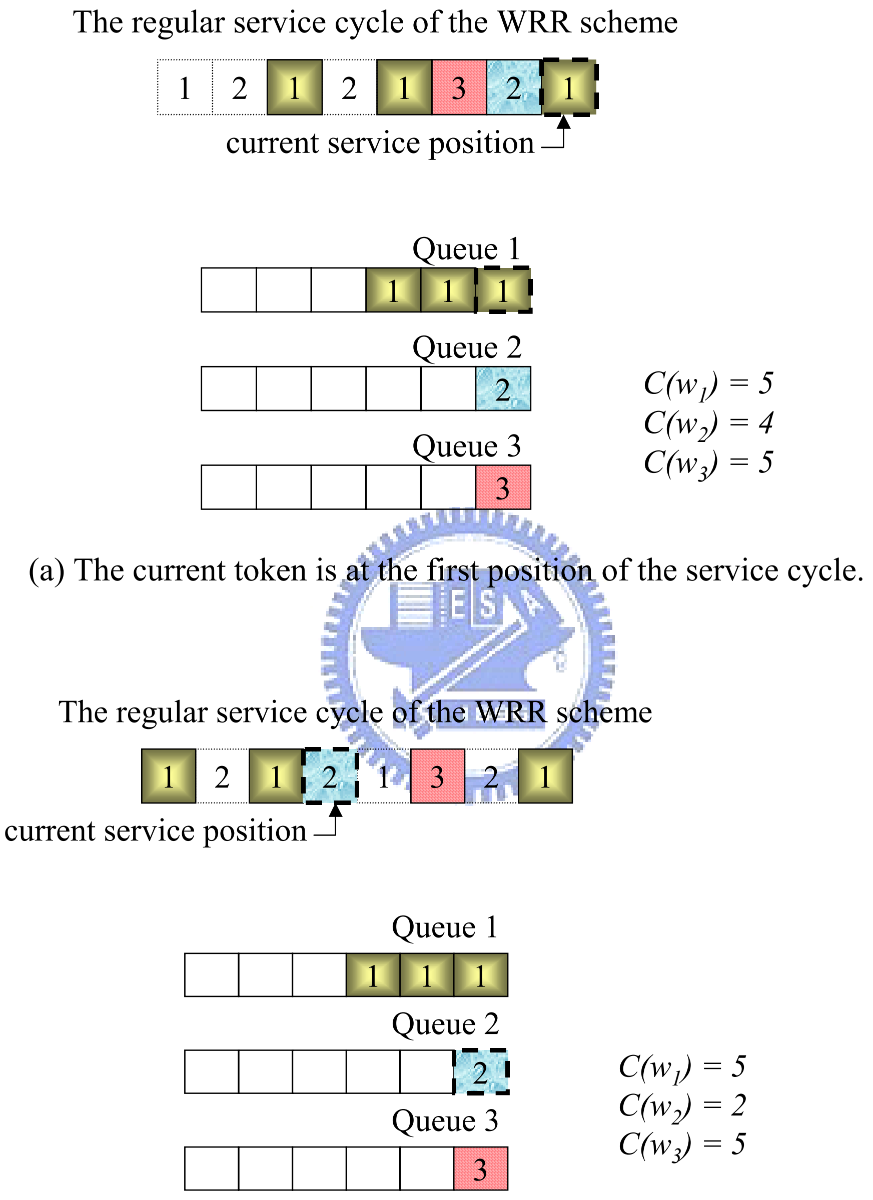


Fig. 2.2 The CIOQ switch using the proposed approximate LCF/MUF matching algorithm and serving cells at output ports with the WRR scheme.



(a) The current token is at the first position of the service cycle.

(b) The current token is at the fifth position of the service cycle.

Fig. 2.3 Examples of cushion calculation for the WRR scheme.

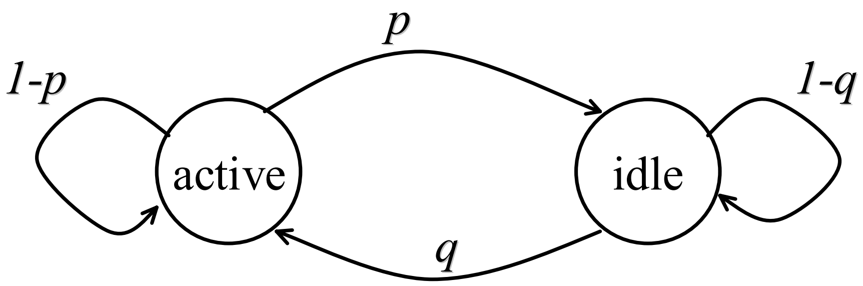


Fig. 2.4 The 2-states Markov chain process.

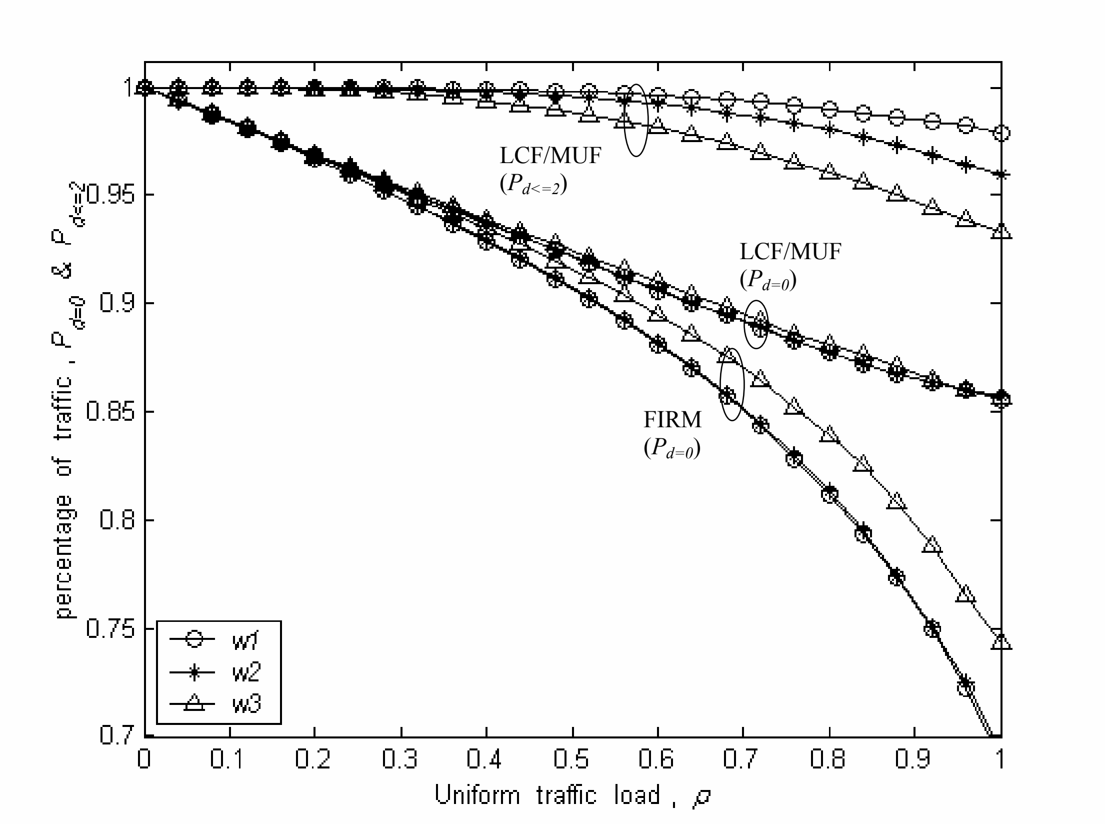


Fig. 2.5 Performances of a 16×16 CIOQ switch under uniform traffic. The curves show the values of $P_{d=0}$ and $P_{d<=2}$ for all the three priority traffic ($w_1 = 4$, $w_2 = 3$, and $w_3 = 1$).

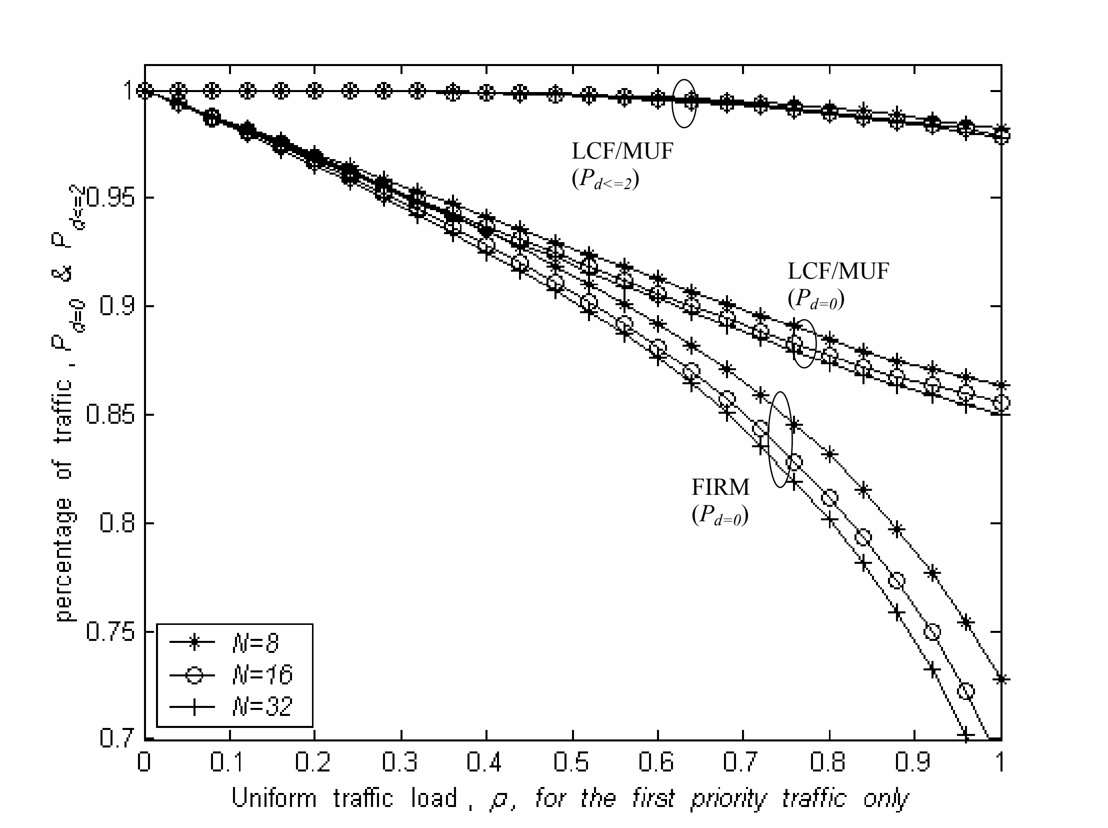


Fig. 2.6 Performances of the first priority traffic as a function of offered load for various switch sizes.

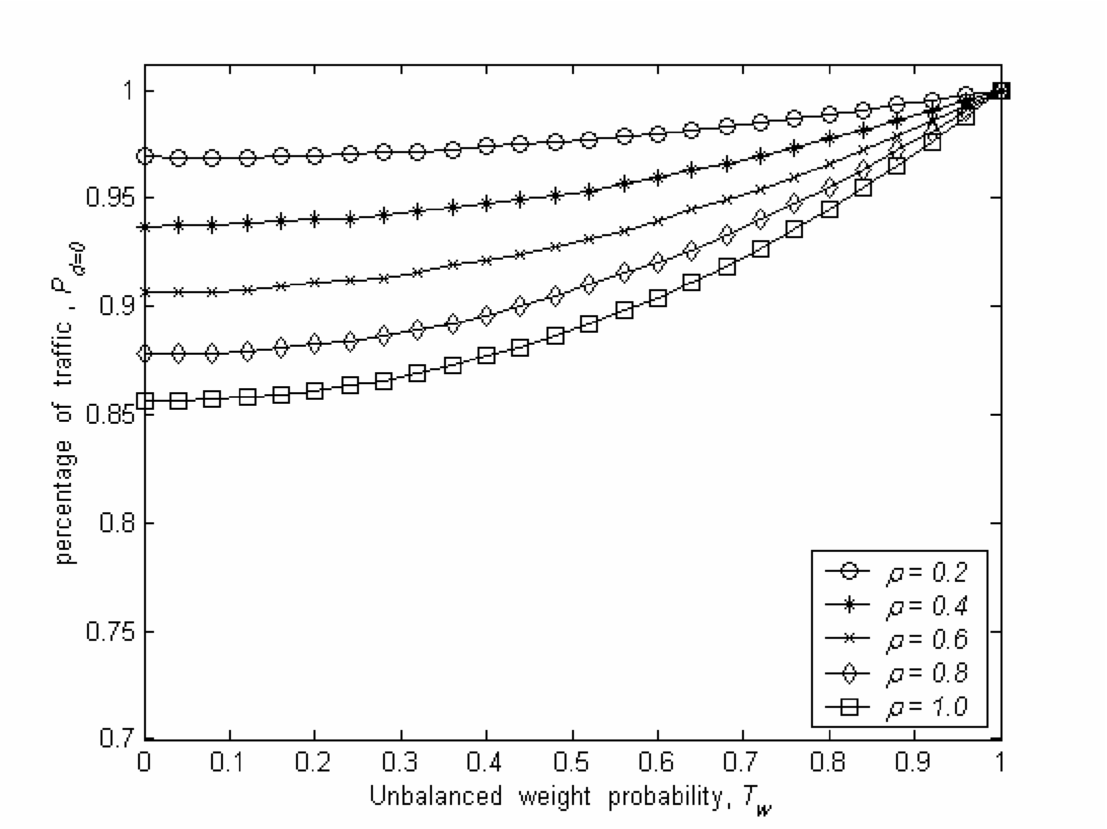


Fig. 2.7 Performances of a 16×16 CIOQ switch adopted the proposed LCF/MUF algorithm. The curves show the values of $P_{d=0}$ of the first priority traffic for different $T_w$ values non-uniform traffic for various traffic loads.

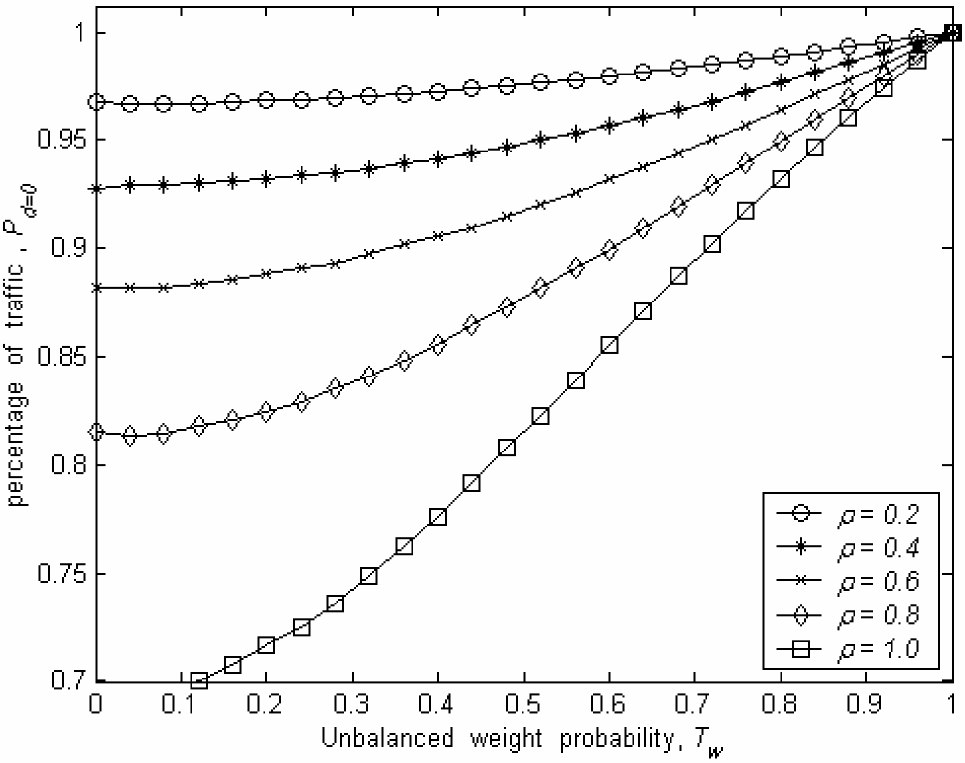


Fig. 2.8 Performances of a 16×16 CIOQ switch adopted the FIRM algorithm. The curves show the values of $P_{d=0}$ of the first priority traffic for different $T_w$ values non-uniform traffic for various traffic loads.

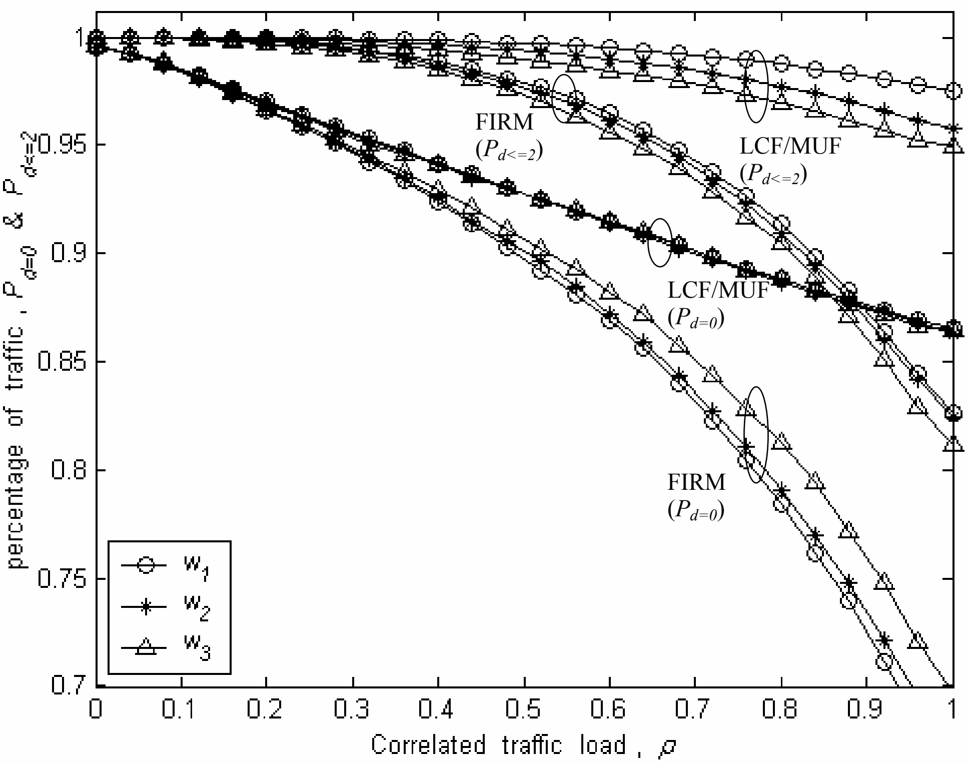


Fig. 2.9 Performances of a 16×16 CIOQ switch with $\ell$ = 16. The curves are shown for all the three priority traffic ($w_1 = 4$, $w_2 = 3$, and $w_3 = 1$).

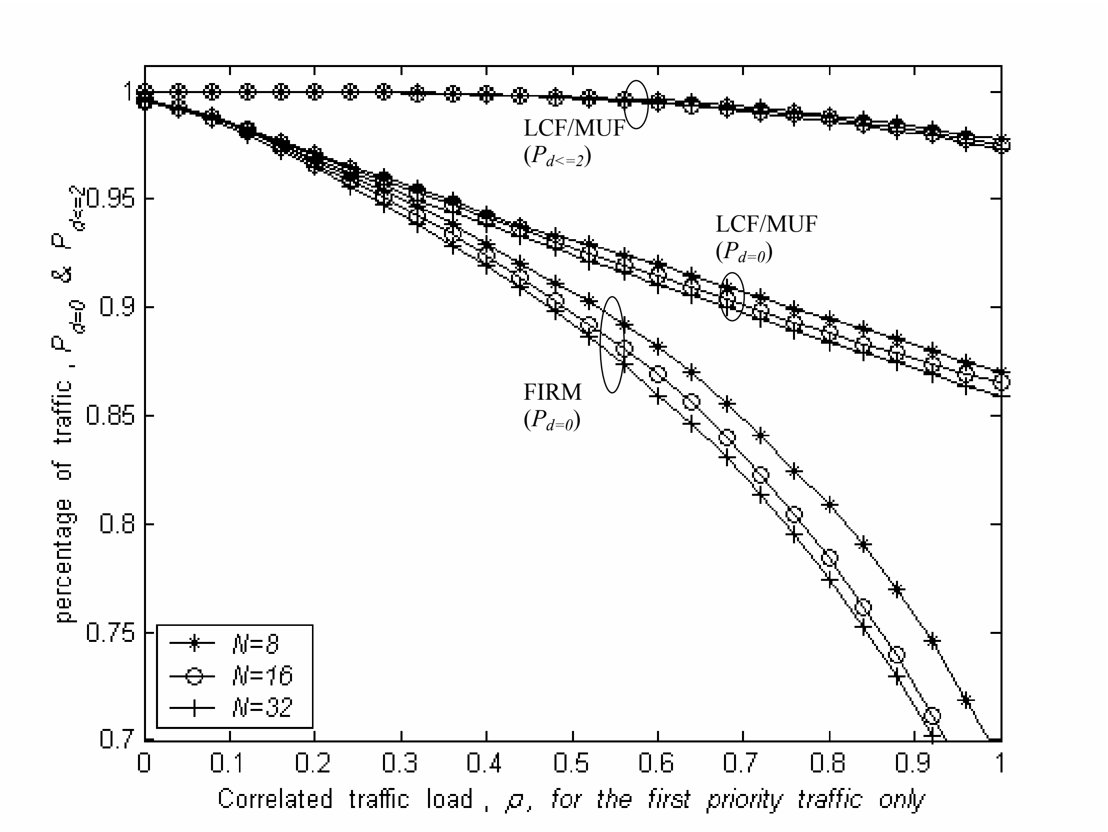


Fig. 2.10 Performances of the first priority traffic for various switch sizes with $\ell$ = 16 cells.

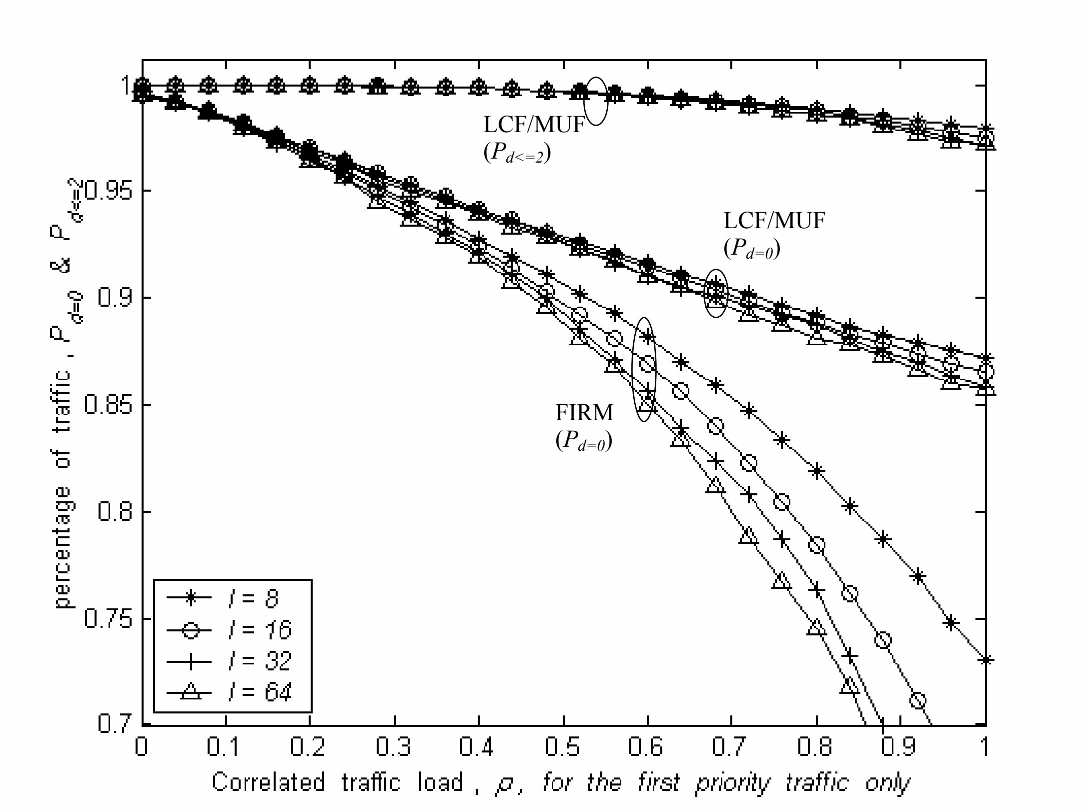


Fig. 2.11 Performance of a 16×16 CIOQ switch under correlated traffic. The curves show the performances of the first priority traffic as a function of offered load for various mean burst length $\ell$ = 8, 16, 32, and 64 cells.

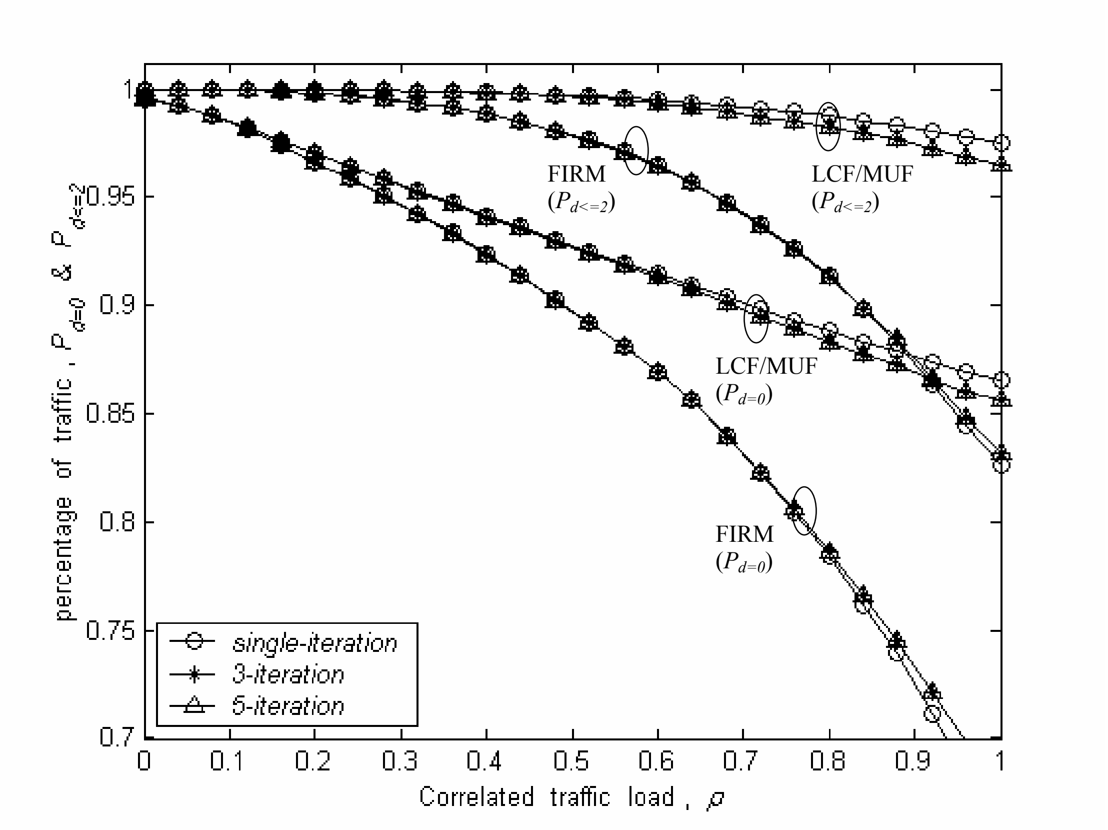


Fig. 2.12 Performance of a 16×16 CIOQ switch under correlated traffic with mean burst length ℓ = 16. The curves show the performances of the first priority traffic under single-iteration, 3-iteration, and 5-iteration matching.

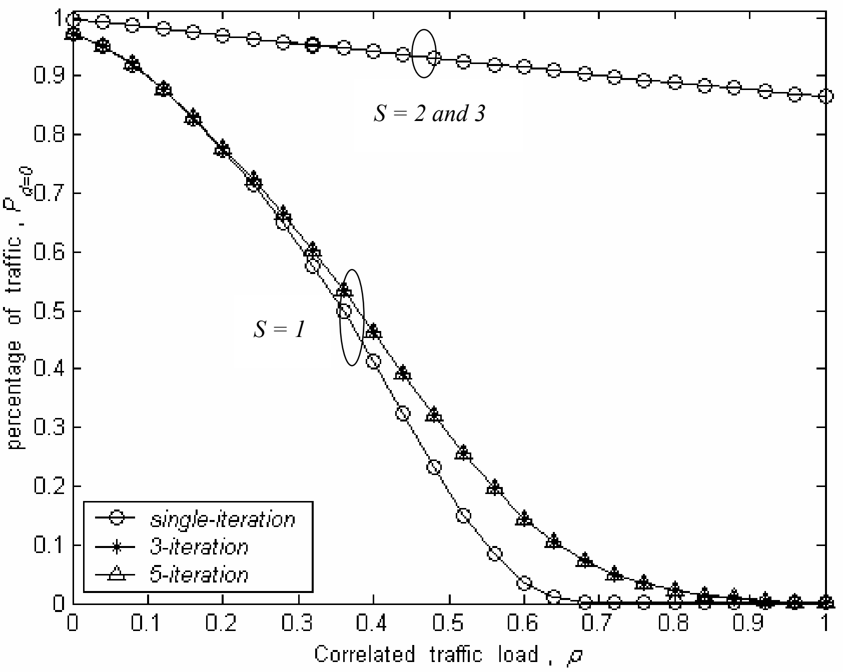


Fig. 2.13 Performance of a 16×16 CIOQ switch under correlated traffic with mean burst length $\ell$ = 16. The curves show the performances of the first priority traffic under single-iteration, 3-iteration, and 5-iteration matching for various speedup factors ($S$ = 1, 2, and 3).

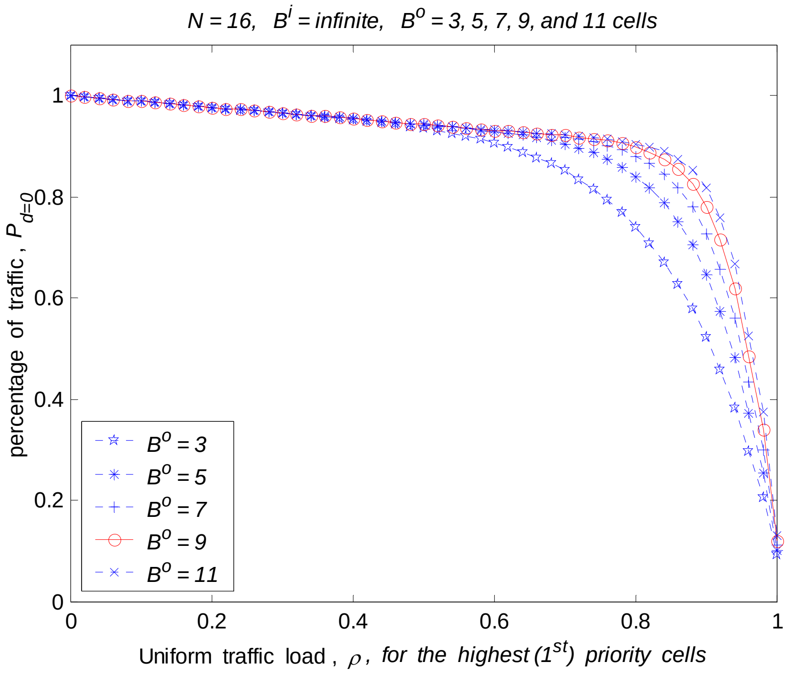


Fig. 2.14 Performance of the 16 × 16 CIOQ switch with $B^i = \infty$ and $B^o = 3, 5, 7$, 9, and 11 cells. The curve is shown for the highest priority cells only.

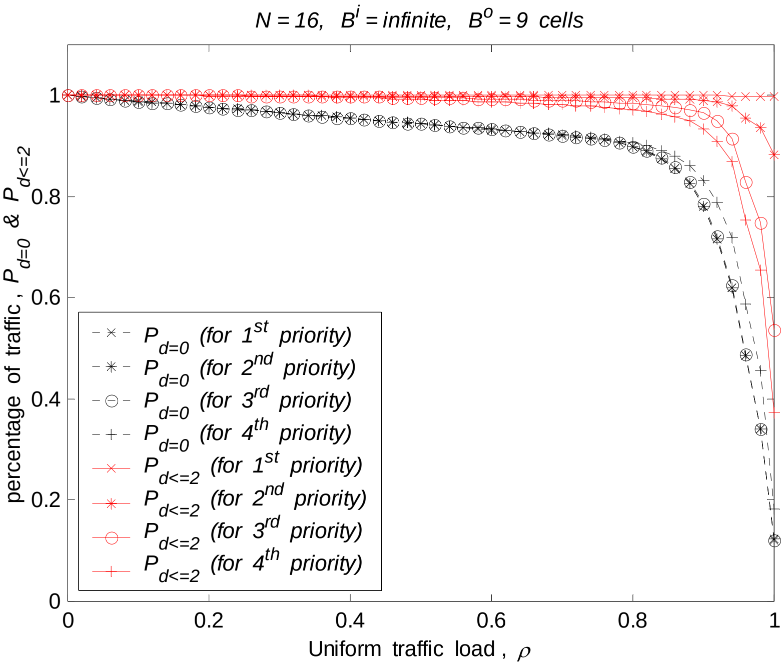


Fig. 2.15 Performance of the 16×16 CIOQ switch with $B^i = \infty$ and $B^o = 9$ cells. The curves show the performance of values of the $P_{d=0}$ and $P_{d<=2}$ for all the four priority cells.

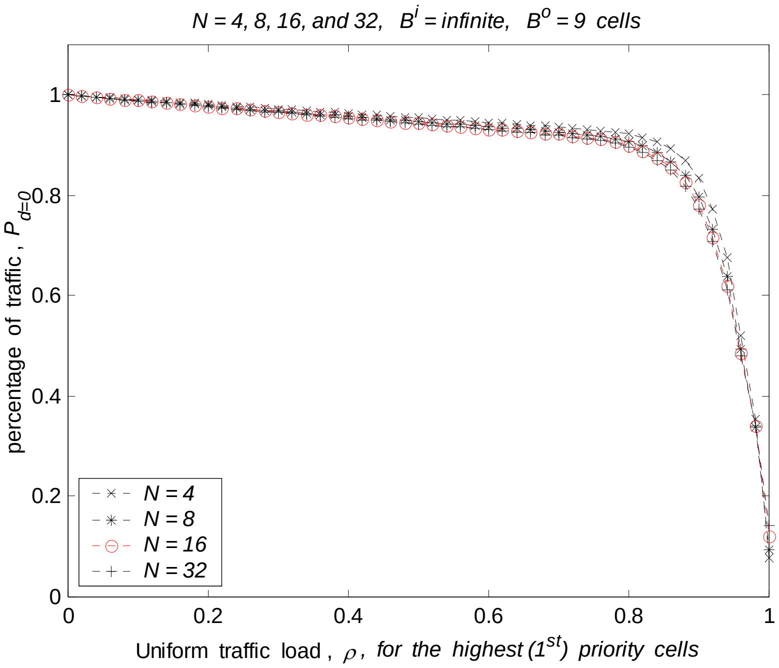


Fig. 2.16 Performance of the CIOQ switch with $B^i = \infty$ and $B^o = 9$ cells. The curve shows the performance of the highest priority cells as a function of the switch size $N = 4$, 8, 16, and 32.

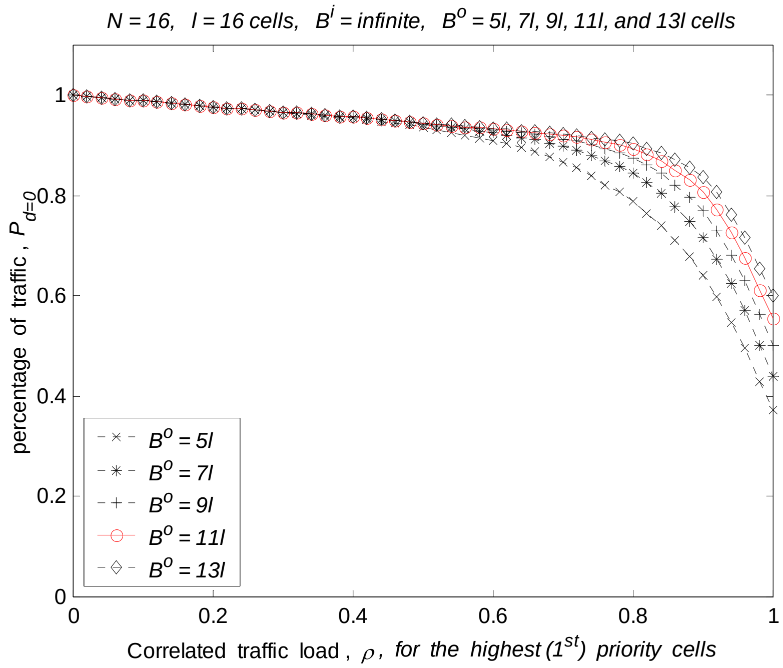


Fig. 2.17 Performance of the 16×16 CIOQ switch with $B^i = \infty$ and $B^o = 5\ell$, $7\ell$, $9\ell$, $11\ell$ and $13\ell$ cells ($\ell$=16). The curve is shown for the highest priority cells only.

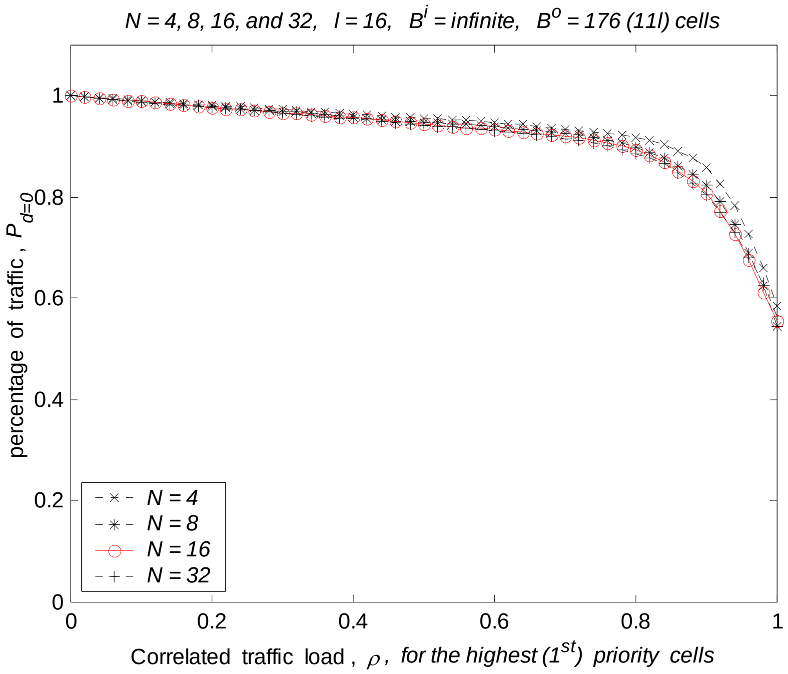


Fig. 2.18 Performance of the CIOQ switch with $B^i = \infty$ and $B^o$ = 11 ℓ cells ( ℓ =16). The curve shows the performance of the highest priority cells as a function of the switch size $N$ = 4, 8, 16, and 32.

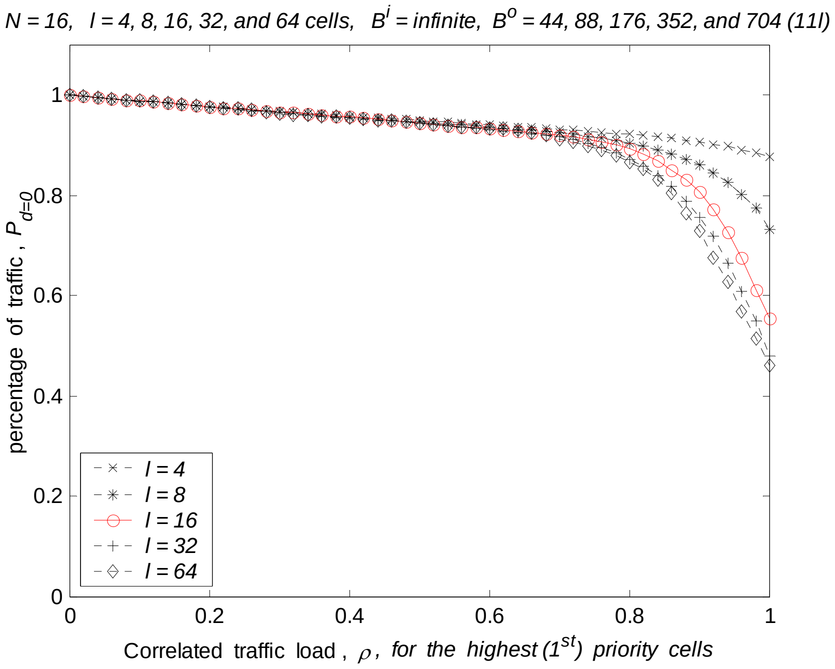


Fig. 2.19 Performance of the 16×16 CIOQ switch with $B^i = \infty$ and $B^o = 11\ \ell$ cells. The curve shows the performance of the highest priority cells as a function of the mean burst length $\ell$ = 4, 8, 16, 32 and 64 cells.

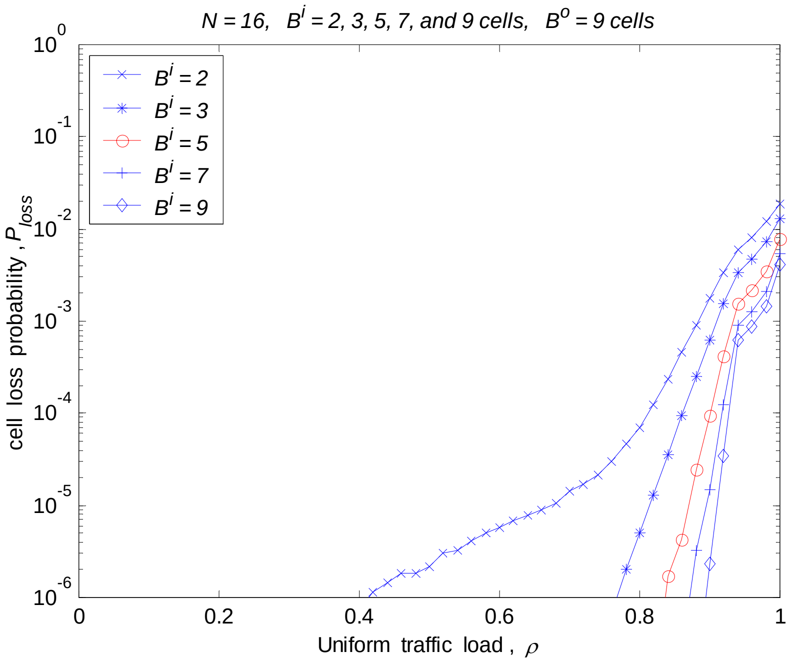


Fig. 2.20 $P_{loss}$ of a 16×16 CIOQ switch with $B^i$ = 2, 3, 5, 7, and 9 cells, and $B^o$ = 9 cells under uniform input traffic.

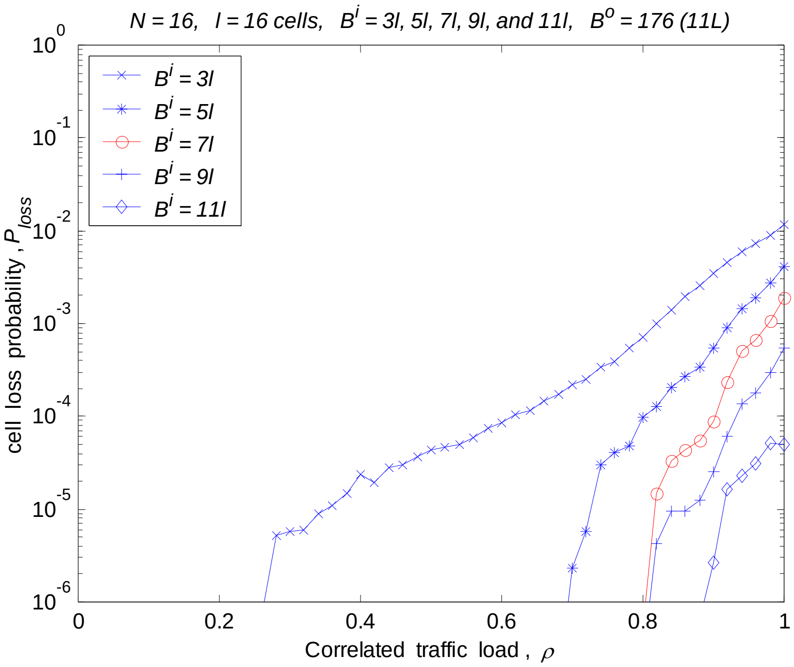


Fig. 2.21 $P_{loss}$ of a 16×16 CIOQ switch with $B^i$ = 3 ℓ, 5 ℓ, 7 ℓ, 9 ℓ, and 11 ℓ, and $B^o$ = 11 ℓ cells ( ℓ =16 cells) under correlated input traffic.

# Chapter 3

# Packet-Based LCF/MUF Matching Algorithm

## 3.1 Introduction

The selection of packets to be transmitted in each time slot from input ports to output ports is accomplished using a scheduling algorithm, which is the key component in achieving high performance using a switch. Most of previous scheduling algorithms found in open literature were designed for fixed-length packets (or cells). These proposed algorithms configure a set of non-conflict connections of the switching fabric between input ports and output ports in each time slot and re-configure all connections in the next slot. Hence, in the current Internet environment, segmenting packets of variable-lengths into cells and re-assembling the cells into the original packets are needed before and after packets transmissions across the switching fabric because cells belonging to various packets may interleave with one another. As a result, such a process can be time consuming and may not be easily implemented at very high speed. Therefore, a scheduling algorithm designed for fixed-length packets has to be modified to make it useful for switching Internet packets. There are some algorithms [52], [53] derived from parallel iterative matching (PIM) algorithm [27] that can handle variable-length packets. A variable-length packet will be scheduled as a whole and transmitted continuously in the switch rather than schedule them on a cell-by-cell basis. In [52], the authors showed that an algorithm using a rotating priority (round-robin) arbitration and a masking operation can achieve 100% throughput with a single iteration for uniform IP traffic and perform better packet latency and similar cell latency, compared with the well-known *i*SLIP cell-based scheduling algorithm. In [53], the authors proposed a burst-based PIM (BPIM) algorithm for an input queued (IQ) switch with multiple input queues. The maximum throughput using single-iteration BPIM scheduling exceeds 0.9. However, these algorithms were developed for IQ switches in achieving high throughput but they are difficult to provide quality of service (QoS). In [57], a distributed packet fair queuing (D-PFQ) architecture was proposed to handle variable-length packets with advanced QoS requirements. Buffers are placed at input ports, output ports, and the crossbar switching fabric. An important advantage of the D-PFQ architecture is that no global synchronization is necessary. Numerical results showed that the D-PFQ architecture provides service that closely approximates an output queued (OQ) switch employing fair queuing with modest speedup. However, the total number of schedulers is equal to $N^2+3N$ for an $N \times N$ switch, meaning that a large number of states have to be maintained which may become the performance bottleneck for a large-capacity switch.

In this chapter, we present the packet-based least cushion first/most urgent first (PB-LCF/MUF) algorithm for combined input and output queued (CIOQ) switches and evaluate its performance via computer simulations. Since cushion calculation is complicated, to simplify the implementation of the PB-LCF/MUF algorithm, an approximate cushion is adopted again. In order to provide QoS guarantee to different applications, our design is applicable to a network environment where packets are classified into multiple priority classes. Results show that the performance of a CIOQ switch with a speedup factor of 5 which adopts the single-iteration PB-LCF/MUF algorithm is close to that of an OQ switch under the weighted round robin (WRR) service discipline. The single-iteration version not only simplifies the matching procedure but also avoids out-of-order packet delivery from input ports to output ports.

## 3.2 The Packet-based LCF/MUF Matching Algorithm

The architecture of the considered CIOQ switch is the same as mentioned in Section 2.3. Because of that our design is applicable to a network environment where packets are classified into $K$ priority classes. At input port $i$, there are $K$ queues denoted by $VOQ_{i,j,k}$, $1 <= k <= K$, for storing packets destined to output port $j$ ($1 <= j <= N$). As a result, there are $N \times K$ queues maintained at each input port. A new arrival packet is placed at the tail of the appropriate queue depending on its destination and class and then becomes eligible to be scheduled for delivery only after the whole packet arrives completely. Although packets are of variable lengths, we assume that every packet can be fragmented into an integral number of smaller fixed-length cells. A CIOQ switch with a speedup factor of $S$ is able to make $S$ cell transfers from each input to outputs during each time slot. In other words, a slot is further divided into $S$ equal sub-slots for the switching fabric and scheduling is performed at the beginning of each sub-slot. To eliminate the possibility of interleaving the cells of multiple packets to save the reassembling effort, a packet will be delivered in consecutive sub-slots as a whole until it is completely delivered. At each output port $j$, there are $K$ queues, denoted by $OQ_{j,k}$, $1 <= k <= K$, for storing packets. At the same time, the service discipline studied in this chapter is WRR because it is a scheme that can be easily implemented and is able to provide QoS guarantee.

In the LCF/MUF algorithm (described in section 2.2) which handles fixed-length packets (or cells), the cushion $C(x_{i,j})$ of a packet $x_{i,j}$ holding by some input port $i$ that is destined to output port $j$ is defined to be the number of packets currently residing in output port $j$ that will leave the shadow OQ switch earlier than the packet $x_{i,j}$. For the PB-LCF/MUF algorithm, the cushion of a packet of variable-length destined to output port $j$ can be defined as the total transmission time of all packets currently at output port $j$ that will leave the shadow OQ switch earlier than the packet. As mentioned before, to implement the LCF/MUF algorithm, a switch has to know the cushion of all packets and their relative departure order in the emulated OQ switch. An approximate cushion is used to simplify implementation of the PB-LCF/MUF algorithm since the calculation of real cushion is complicated. Our choice of the approximate cushion is simply the number of packets currently queued at the destination output port in the CIOQ switch. For example, the approximate cushion of a packet with class $k$ waiting for scheduling to output port $j$ is equal to the number of packets queued at $OQ_{j,k}$.

### 3.2.1 Single-iteration Matching

The single-iteration PB-LCF/MUF scheduling algorithm is described below. It consists of three phases.

**Phase 1**. Input port $i$ sends a request to output port $j$ for all $i$ and $j$ $(1 <= i, j <= N)$. The request contains the arrival times of the HOL packets of all the $K$ queues. The arrival time is set to infinity if a queue is empty. Note that if input port $i$ is matched to output port $j$ in the previous sub-slot and there are more cells for the same packet, then the request sent by input port $i$ to output port $j$ will be the same as that sent in the previous sub-slot.

**Phase 2**. Output port $j$ sends a grant message back to input port $i$ if it was matched to input port $i$ in the previous sub-slot and there are more cells of the same packet waiting to be delivered to output port $j$. Otherwise, it calculates the cushions for the HOL packets contained in the requests and selects the request with the least cushion. If there is a tie, then the packet with the earliest arrival time wins the contention. It then sends a grant message back to the input port which sent the selected request in Phase 1. The cushion is contained in the grant message.

**Phase 3**. Input port $i$ selects the grant sent by output port $j$ if it was matched to output port $j$ in the previous sub-slot and there are more cells of the same packet waiting to be delivered to output port $j$. Otherwise, it selects the grant with the least cushion. If there is a tie, then the packet with the earliest arrival time is selected. Assume that the grant message sent by output port $k$ is selected. Input port $i$ sends a cell of the HOL packet to output port $k$.

It is worth noting that the proposed algorithm causes no cell interleaving of packets and therefore no re-assembly buffer at output ports is needed. Moreover, the departure order of packets can also be maintained at output ports with the single-iteration version. The reason is explained as follows. Consider, for example, the $2 \times 2$ CIOQ switch shown in Fig. 3.1. Assume that input port 1 is matched to output port 1 in the previous sub-slot and there are more cells of packet $z$ waiting to be delivered. Let packets $x$ and $y$ be of the same class (different from the class of packet $z$) at different input ports. Assume that both $x$ and $y$ are HOL packets and destined for the same output port 2 and $x$ arrived earlier than $y$. Then, in current matching sub-slot, output port 1 will grant input port 1 for packet $z$ and output port 2 will grant input port 1 for packet *x* in phase 2. Consequently, packet *y* will be blocked until packet *x* has been scheduled. Therefore, packet out-of-order will not happen at any output port.

### 3.2.2 Multiple-iteration Matching

To improve the throughput performance of the switching fabric, the above matching procedure can be repeated for multiple iterations. After each iteration, an input port stops sending requests once it has sent an accept message to a certain output port. Similarly, an output port stops calculating the cushions and sending a grant message once it has received an accept message. The subsequent iterations will try to match the rest of the input and output ports that remain unmatched in previous iterations. Obviously, the throughput increases as the number of iteration increases. However, multiple iterations may result in out-of-order delivery of packets from input ports to output ports. Moreover, our simulation results show that multiple-iteration matching only yields negligible performance improvement over single-iteration matching if the switching fabric is speeded up by a factor of 2.

## 3.3 Input Traffic Model

To evaluate the performance of the CIOQ switch using the proposed matching algorithm, we measure the latency of the packets which are simultaneously fed to both the shadow OQ switch and the CIOQ switch under a practical input traffic model. The traffic model is characterized by a 2-state Markov Modulated Bernoulli Process (2-MMBP) [21],[56] alternating between active and idle states. The traffic source will generate a cell every time slot when it is in the active state. A packet consists of the cells generated by consecutive active states. We assume there is no correlation between different packets and that the destination of each packet is uniformly distributed among the output ports and every packet is equally likely to be any traffic class. According to the real IP networks traffic data collected in [58] from the wide area network, the distribution of packet size has two masses traffic at 40 bytes (about a third) due to TCP acknowledgment packets and 552 bytes (about 22%) due to maximum transmission unit (MTU) limitations at many routers. Other prominent packet sizes include 44 bytes (about 3%), 72 bytes (about 4.1%), 185 bytes (about 2.7%), 576 bytes (about 3.6%), and 1500 bytes (about 1.5%), due to Ethernet MTU. So, we approximately model the period of active state with distributions of 33% at 40 bytes, 22% at 552 bytes, 30% between 40 bytes and 552bytes, and 15% between 552 bytes and 1500 bytes. The cumulative distribution function of this traffic model is shown in Fig. 3.2. Additionally, we convert the IP packet in bytes to the corresponding number of ATM cells using AAL5 null encapsulation technique [59]. As a result, the average packet size is about 8.295 ATM cells.

## 3.4 Numerical Results

For performance evaluations, we use $d(x)=|d_{oq}(x)-d_{cioq}(x)|$ defined in Section 2.5 as the deviation index of packet $x$ with variable-length as a criterion again. Here $d_{oq}(x)$ and $d_{cioq}(x)$ denote, respectively, the departure times of packet $x$ for the shadow OQ switch and the CIOQ switch. Given a fixed $d$, we measure the percentage of packets that has a deviation index smaller than or equal to $d$. This percentage is denoted by $P_d$. For example, $P_{d=0}$ equals 100% means that the CIOQ switch exactly emulates the shadow OQ switch.

We perform simulations for a CIOQ switch with different switch sizes and speedup factors. Simulations are performed for $N = 16$, 32, and 64 with $K = 3$ or 8. First of all, we show the performance curves for $N = 32$ and $K = 3$. The weights for the three traffic classes are chosen to be $w_1 = 4$, $w_2 = 3$, and $w_3 = 1$. Figure 3.3 shows the results for a $32 \times 32$ CIOQ switch with a speedup factor of 2 and adopts the proposed single-iteration PB-LCF/MUF algorithm. It can be seen that the performance degrades dramatically as the offered load increases for the class 1 traffic. The value of $P_{d=0}$ is about 42.77% under an offered load of 0.9 which obviously is not acceptable. To avoid drawing too many curves, which would make the figure less comprehensible, the performances of class 2 and class 3 traffics (which are similar to that for class 1 traffic) are not shown in Fig. 3.3. Instead, we include the performance curves of the CIOQ switch, with a speedup factor of 2, using the packet-based FIRM (PB-FIRM) algorithm with packet re-ordering buffers at output ports for comparison. The PB-FIRM is modified from the FIRM algorithm [37] which was proposed for cell-based scheduling. The matching procedures of PB-FIRM algorithm are described below：

(1) Keep all the matches in the last time slot which are still valid at the current time slot.

(2) Each unmatched input sends a request to every output for which it has a buffered packet.

(3) Every output sends a grant to the input port which sent the request that appears next in a fixed, round-robin schedule. If the grant is accepted in step 4, the round-robin pointer is incremented (mod *N*) to one location beyond the granted input. If the grant is not accepted, pointer is placed to the granted input.

(4) Every input accepts the grant that appears next in a fixed, round-robin schedule. The accept pointer is incremented (mod *N*) to one location beyond the accepted output.

In Fig. 3.3, it can be seen that the PB-LCF/MUF algorithm performs better than the PB-FIRM algorithm. We also performed simulations for the CIOQ switch using the packet-based *i*SLIP (PB-*i*SLIP) algorithm which is modified from the cell-based *i*SLIP algorithm [36]. The performances of PB-*i*SLIP algorithm is worse than that of the PB-FIRM algorithm and is not shown in this thesis.

As mentioned before, to increase the throughput of the switching fabric, one can perform the matching procedure iteratively to approach a maximal matching. But multiple-iteration will cause out-of-order packet delivery and increase the complexity of the matching procedure. In our experiments, the matching algorithm was performed with one, two, or three iterations and, if needed, packet re-ordering buffers are provided at output ports. The results are shown in Fig. 3.4 (again, for a speedup factor of 2). As can be seen in Fig. 3.4, multiple-iteration matching does not result in noticeable performance improvement, which indicates that one iteration is enough for the PB-LCF/MUF algorithm to find a maximal matching under any traffic load.

One method to both increase throughput and maintain packet order is to speed up the switch fabric. This method is feasible for moderate speedup factors. We plot in Fig. 3.5 the performance curves for class 1 traffic with speedup factor $S = 1, 2, 3, 4, 5,$ and 6. It can be seen that, for $S = 1$ which means no speedup, the system performance degrades seriously as the traffic load increases. The reason is that the single-iteration PB-LCF/MUF algorithm is not a maximum matching. In fact, it is not even a maximal matching. However, the total number of matches achieved in a time slot increases (and hence system performance improves) dramatically with speedup. The value of $P_{d=0}$ is larger than 90% under traffic load up to 0.9 when the speedup factor is increased to 5. As mentioned before, we also performed simulations for $N = 16$ and 64 and the results show that a speedup factor of 5 still yields satisfactory performance. Fig. 3.6 shows the results of $N = 64$ for various speedup factors. Similar results were obtained for eight traffic classes with various weights. Therefore, we conclude that the CIOQ switch with a speedup factor of 5 which adopts the proposed single-iteration approximate PB-LCF/MUF algorithm can closely emulate an OQ switch.

## 3.5 Summary

In this chapter, we have proposed a packet-based least cushion first/most urgent first scheduling algorithm for CIOQ switches that handle variable-length packets in current Internet environment. Since the proposed algorithm requires only one iteration, the packet re-ordering problem is eliminated. The PB-LCF/MUF algorithm delivers each packet as a whole and thus no cell interleaving of packets happened at output ports. Therefore, no re-assembly buffer is needed. Numerical results obtained from computer simulations show that our proposed algorithm yields satisfactory performance when the speedup factor is 5 for various switch sizes and traffic classes. We believe that, with a modest speedup, a CIOQ switch can exactly emulate an OQ switch even for variable-length packets.

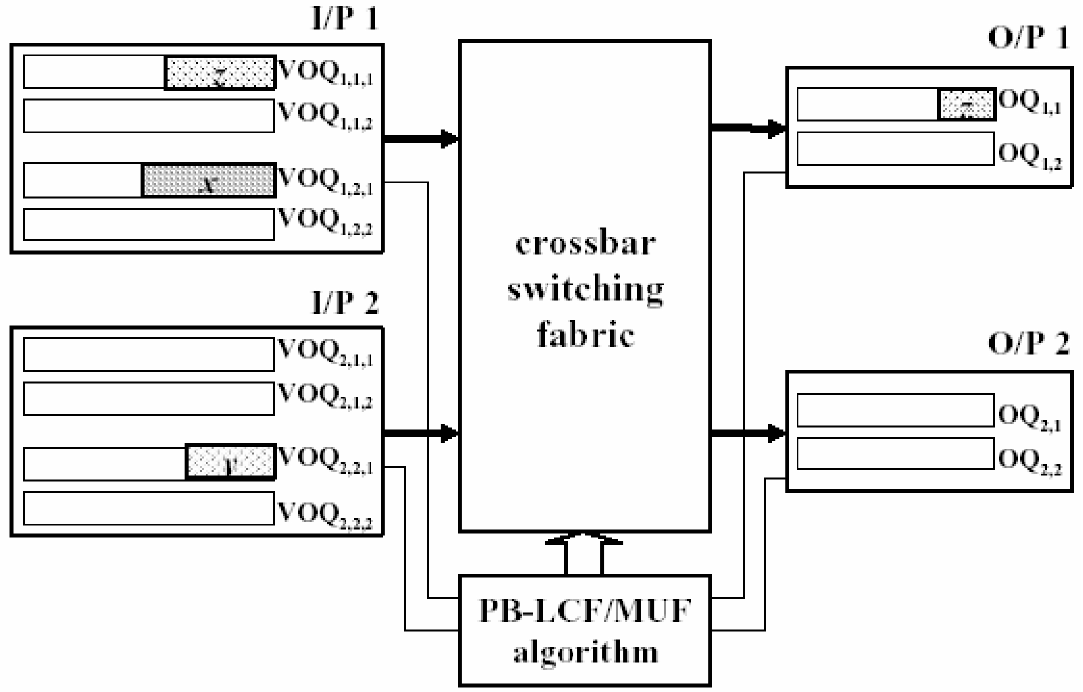


Fig. 3.1. Example of maintaining packet order in a *2×2* CIOQ switch.

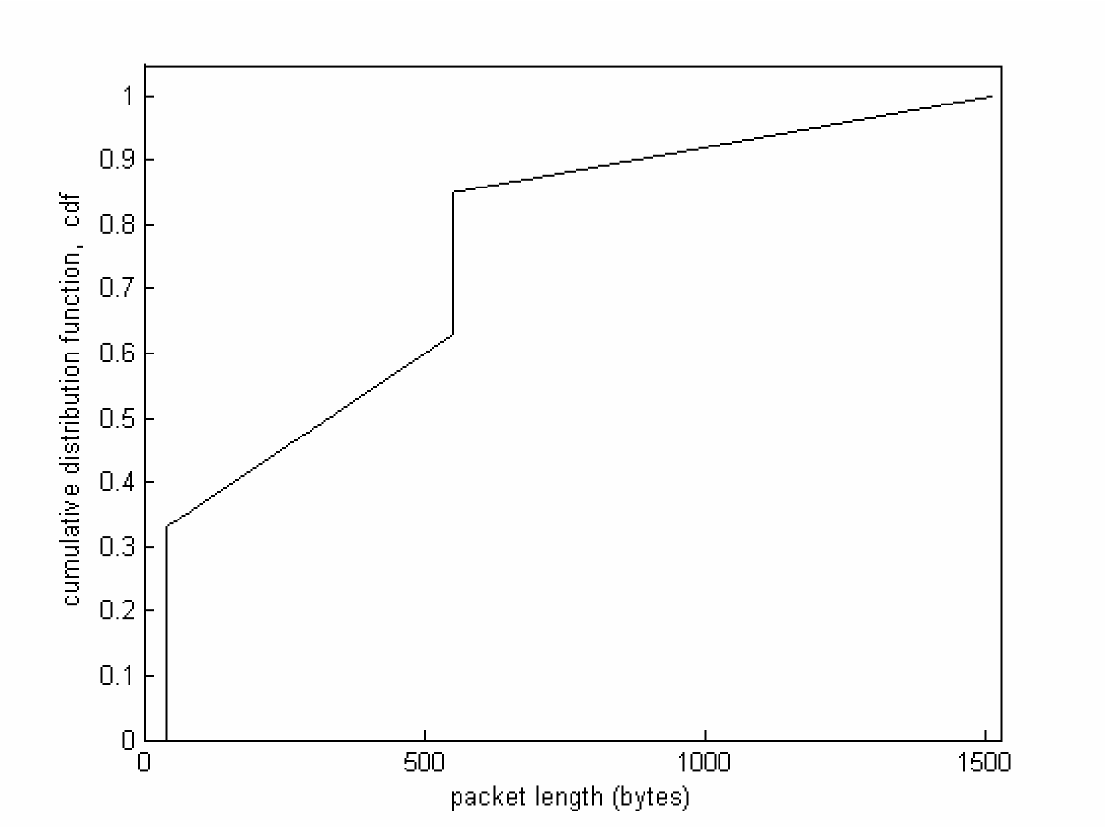


Fig. 3.2. Cumulative distribution function for active-period traffic of a 2-MMBP input traffic model.

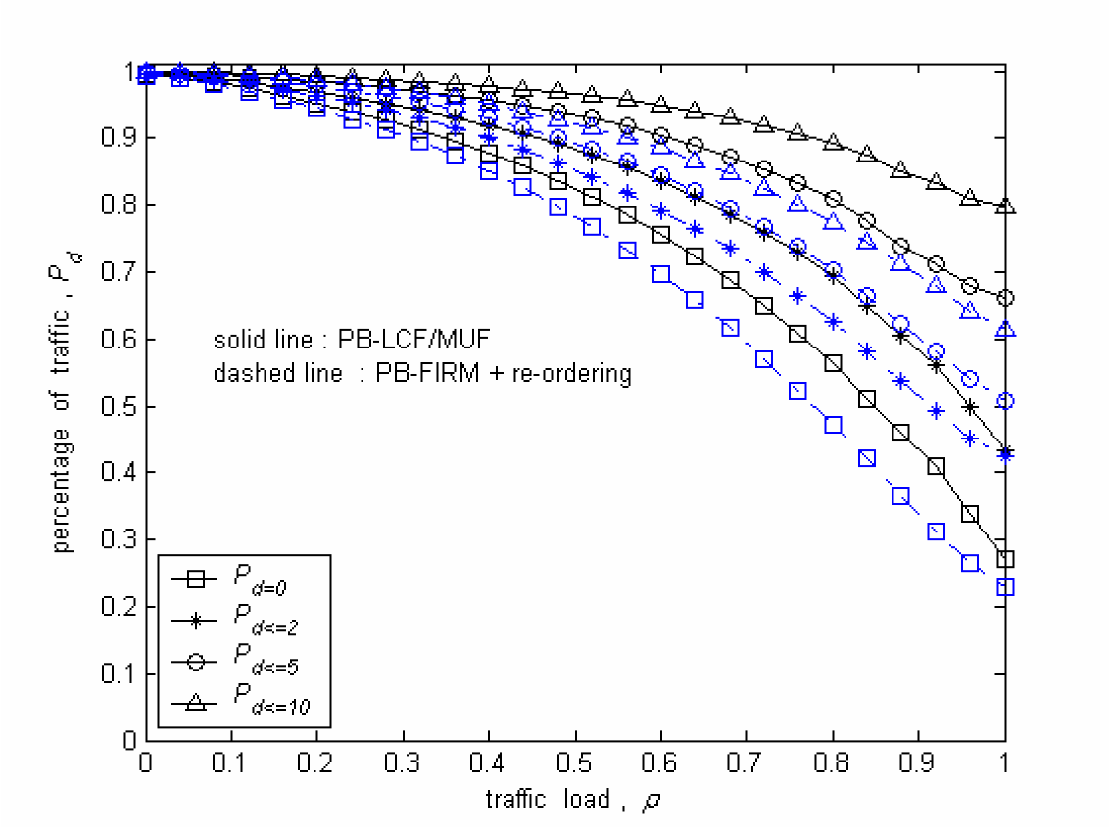


Fig. 3.3. Performance comparison of the proposed PB-LCF/MUF algorithm and the FIRM algorithm for a *32×32* CIOQ switch.

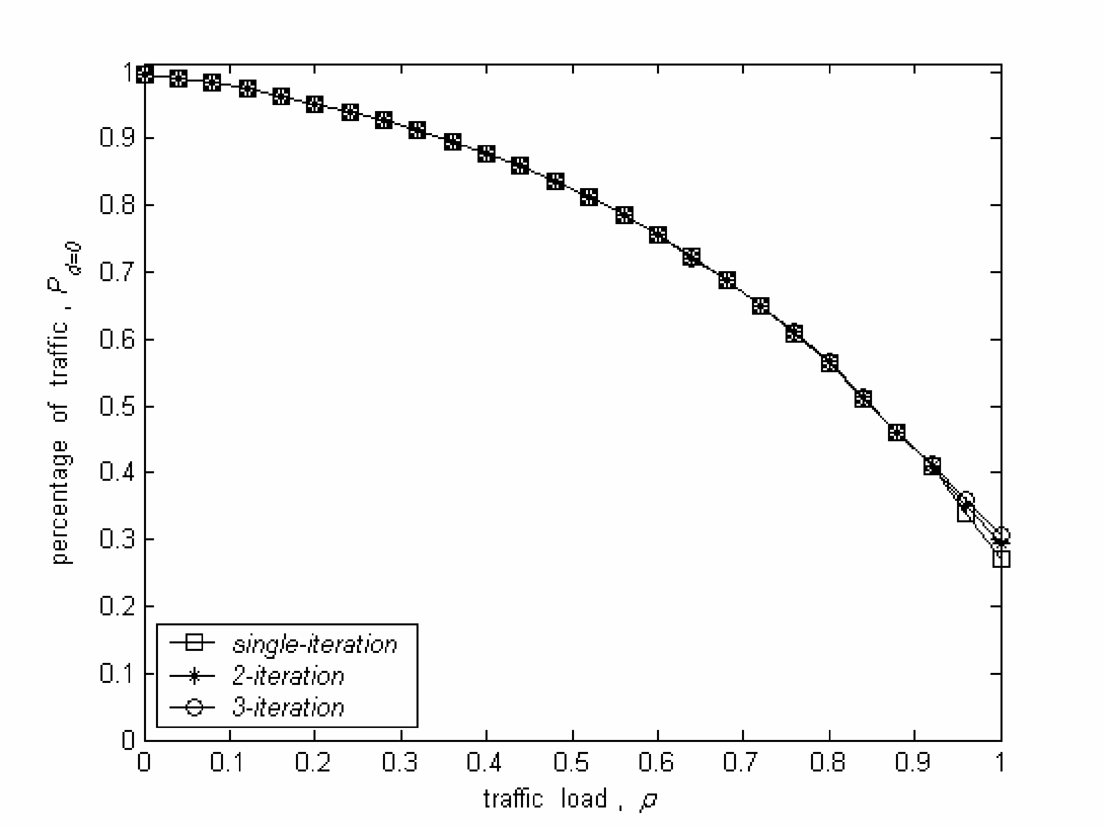


Fig. 3.4. Performance of the class 1 traffic under multiple-iteration matching procedure with packet re-ordering at output ports.

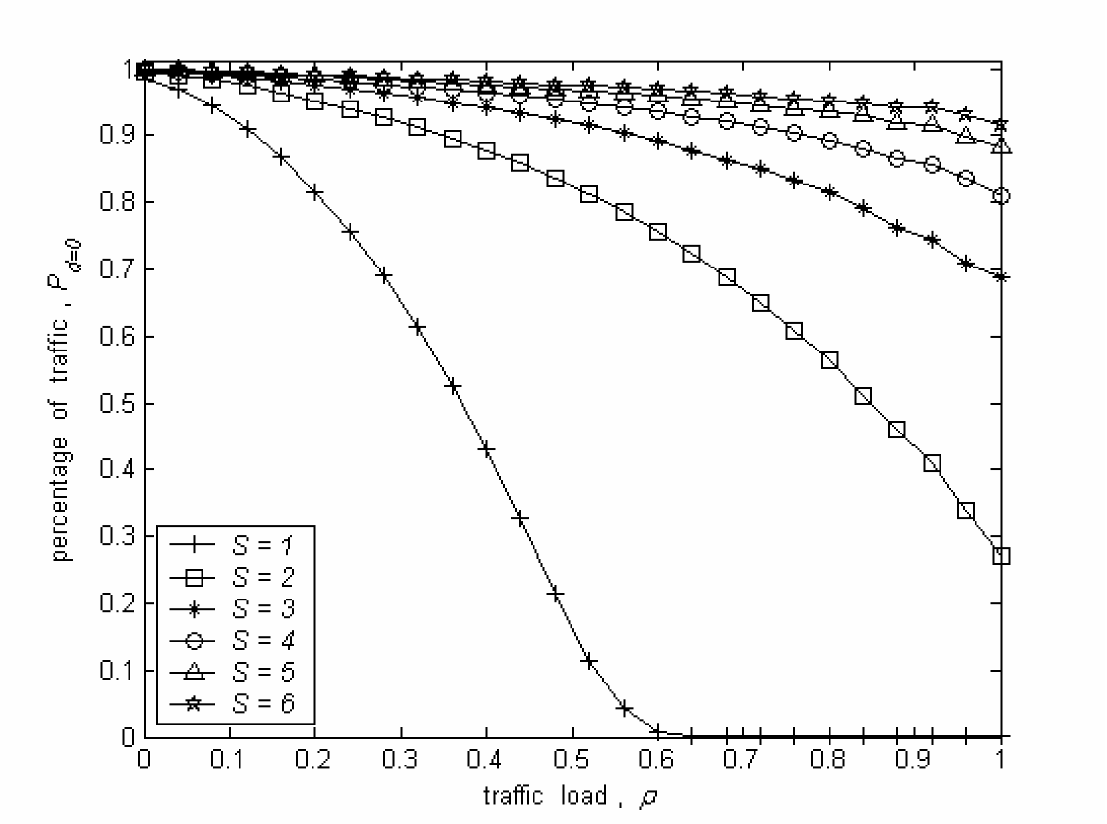


Fig. 3.5. Performance of the class 1 traffic as a function of offered load for various speedup factors.

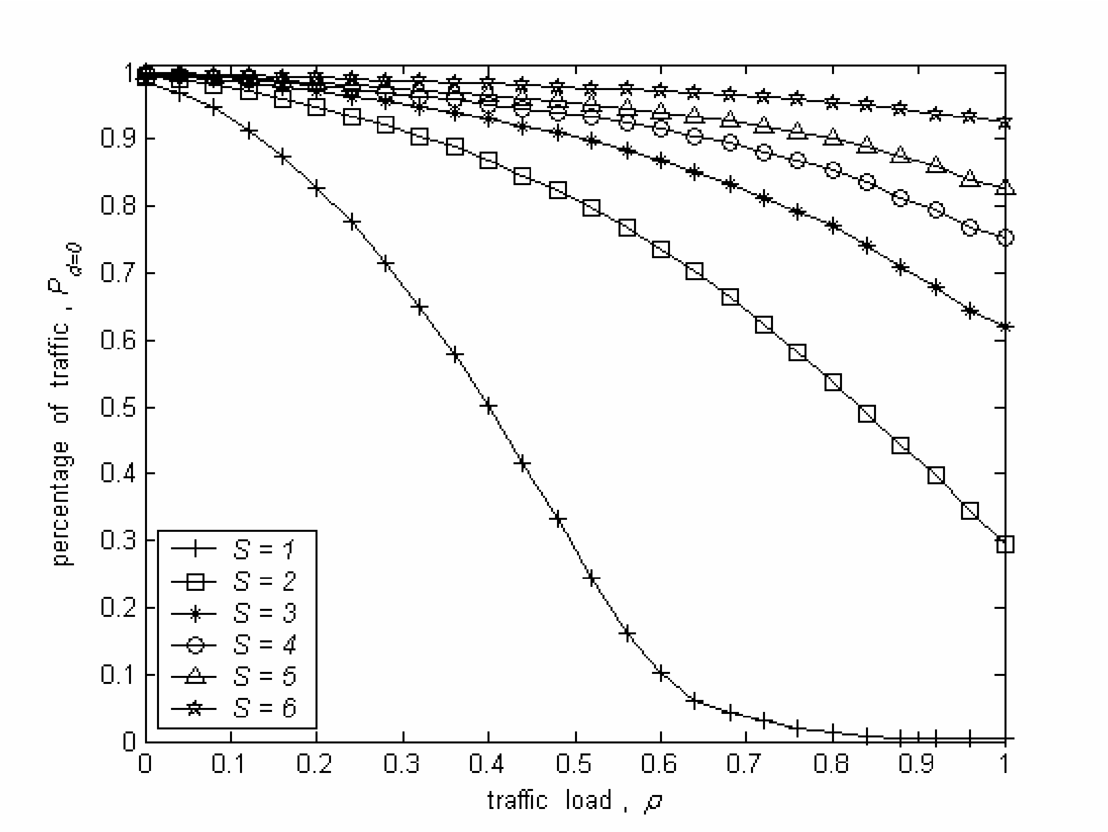


Fig. 3.6 Performance of a *64×64* CIOQ switch for various speedup factors.

# Chapter 4
# Output Initiated Fair Scheduling Algorithm

## 4.1 Introduction

In order to provide QoS guarantee, a switch may have to maintain a large number of queues. However, when considering the cost of hardware implementation, the number of queues is a decisive factor. It is well known that input queued (IQ) switches suffer from head-of-line (HOL) blocking which limits the maximum throughput to 0.586 under uniform traffic [20] and output queued (OQ) switches have serious scalability problem. One possible approach to improve the performance of an IQ switch is to use virtual output queuing (VOQ) combined with a parallel matching algorithm (e.g., PIM[27], WPIM[34], LPF[35], *i*SLIP[36], and FIRM[37]). Moreover, in order to provide QoS guarantee to different applications, our study is applicable to a network environment where packets are classified into multiple priority classes. Accordingly, the number of queues maintained at each input port is $N \times K$ for an $N \times N$ switch applied VOQ queuing structure with $K$ traffic classes which means high complexity when $N$ and $K$ are large. To reduce the complexity and make a system feasible, it is important to reduce the number of queues maintained by the switch. In this chapter, we propose an architecture that requires $K$ queues, one for each traffic class, per port. Furthermore, we assume that the switch handles variable-length packets and, for convenience, every packet can be segmented into an integral number of fixed-length cells. Time is divides into slots so that the duration of a slot is equal to the transmission time of a cell. A packet is not eligible to be delivered to its destined output port before it completely arrives at an input port. There are two choices to place the queues: at input ports or output ports.

Consider the situation where queues are placed at input ports. In this case, several algorithms [52][53][54][60] derived from the PIM algorithm are used to make a set of non-conflict connections of the switching fabric between input ports and output ports for packet delivery. Based on these algorithms, starvation may happen if multiple connection requests are sent from input ports to output ports. Fig. 4.1 illustrates an example for a $2 \times 2$ switch under the traffics with the same class ($K$ = 1). In this example, the first packet (marked as packet $x$) which arrived at input port 2 is of length 3 (cells) and all the other packets are of length 2. All packets, except the one marked as packet $y$, that arrive at input port 1 are destined to output port 1 and all packets that arrive at input port 2 are destined to output port 2. There is no idle slot for input port 2 and only one idle slot (i.e., the first slot) for input port 1. Assume that each unmatched input port sends a connection request to every output port if there is a packet waiting to be delivered to that output port. An output port selects (say, randomly) a request and sends a grant message back to the input port. Upon receiving grant messages, an input port picks one and sends the packet to the corresponding output port. A matched input-output pair remains connected until a packet is completely delivered. It is not hard to see that, with such a matching procedure, packet $y$ will never be serviced if output port 2 selects the request sent by input port 2 at the beginning of slot 4.

As mentioned above, our study is applicable to a network environment where packets are classified into multiple priority classes. In addition to the potential starvation problem, a switch may not be able to achieve weighted fairness among traffic classes if queues are placed at input ports. Fairness in resource allocation is very important to support the need for diverse applications and QoS requirements and thus is studied in this chapter. Many fair queuing algorithms [6]-[10],[27],[34],[54],[60]-[62] have been developed to schedule packets on shared resource. When all output ports have identical weights, one can easily modify the dual round robin matching (DRRM) [62] algorithm to become weighted DRRM (WDRRM, will be described later) to achieve fairness. Furthermore, one can use WFQ, PGPS, and WF$^2$Q algorithms etc. to achieve this goal, but they are proposed adopted at output ports and they are difficult to implement. More recently, an iterative fair scheduling (*i*FS) scheme [60] was proposed for IQ switches to support fair bandwidth distribution among traffic and achieve asymptotically 100% throughput. It maintains VOQs at input ports and thus it is not cost-effective in hardware implementation. Since *i*FS algorithm is based on virtual time [6][8], it is difficult to maintain the virtual transmission time of each packet. In [54], the authors propose fair scheduling based on deficit round robin algorithm [61] for IQ switches to make it feasible in implementation. However, it is not possible to achieve weighted fairness among traffic classes if different output ports have different weights. Although one can use weighted round robin service discipline with different weights at each output port in an OQ switch. But the OQ switch has to run *N* times faster than the line rate. In our proposal, we place queues at output ports and design a parallel matching algorithm initiated by output ports. Details of the proposed output initiated parallel matching (OIPM) algorithm are described in the following section. The proposed OIPM algorithm guarantees starvation free, in-order packet delivery, and achieves weighted fairness among traffic classes required by output ports. At the same time, the switching fabric only runs as fast as the line rate. We also evaluate the performance of our proposed OIPM algorithm via computer simulations. Results show that its throughput performance is close to that of an input buffered switch where queues are placed at input ports. The maximum throughput is 0.520 under uniform traffic which is only slightly smaller than 0.586. To increase the throughput of the switch, one can speed up the switching fabric and add a FIFO queue at each output port. Simulation results show that a speedup factor of 2 is enough to achieve satisfactory performance.

Consider an $N \times N$ input buffered switch with *K* classes of packets. Our proposed switch architecture is shown in Fig. 4.2. Every input port *i* has a shared memory *SM(i)* to store packets arrived at input link. Every output port *j* maintains *K* queues, one for each traffic class. Assume that a class *k* packet, destined to output port *j*, completely arrives at input port *i* in slot *n* and then is stored in the shared memory *SM(i)*. At the beginning of slot *n*+1, input port *i* sends a notification, denoted by *record(i, k, addr, len)*, to output port *j*, where *i* is the input port the packet arrives at, *k* denotes the traffic class of the packet, *addr* represents the starting address of *SM(i)* that stores the packet, and *len* indicates the length (in cells) of the packet. Upon receiving the notification, output port *j* places it to the end of queue *k*. Notice that output port *j* may receive multiple notifications and, if so, will place all the notifications to their appropriate queues.

## 4.2 Output Initiated Parallel Matching (OIPM) Algorithm

The matching procedure, described below, begins after all the notifications are processed in each time slot.

**Step 1**: Each unmatched output port *j* selects one HOL notification from the *K* queues based on the service discipline and sends the notification back to the corresponding input port. For convenience, a notification sent by an output port is called a request.

**Step 2**: An unmatched input port selects a request, if any, based on the round robin scheme. If the request sent by output port *j* is selected by input port *i*, then set input port *i* and output port *j* as matched. The two values *addr* and *len* are extracted from the request and used to read out the packet for delivery.

In current Internet environment, a matching algorithm can efficiently handle variable-length packet for the QoS requirement is a must. In our proposed algorithm, when an input port and an output port are matched, the connection is kept until a complete packet is delivered. Once the packet is completely delivered, the input-output connection pair is reset as unmatched. Consequently, the OIPM algorithm causes no cell interleaving of packets under variable-length packet traffics. Moreover, our proposed OIPM algorithm needs only request-grant matching phases, it is comparable less than request-grant-accept phases required in PIM-similar matching algorithms. That means the OIPM algorithm takes less arbitration time and is much feasible in implementation.

Notice that, although packets are buffered at input ports, queues are maintained at output ports. Each output port can send a request based on the separate fairness service discipline. As such, it is able to achieve weighted fairness among traffic classes even different output ports have different weights. Moreover, packets will be delivered in order because notifications are processed with the First-Come-First-Serve (FCFS) scheme. Finally, no starvation will happen because the same request will be repeatedly sent by an output port before it is selected.

Since each output sends just one request at each time and the request may be denied because of competition, the maximum throughput of our proposed OIPM algorithm is only 0.520 which is slightly smaller than 0.586. To increase the throughput of the switch, one can speed up the switching fabric. In this situation, the switch needs to add a FIFO queue at each output port. Packets transported to output ports are all buffered at the FIFO queues while they are waiting to be transmitted to the outgoing links. Our simulation results show that a speedup factor of 2 is enough to achieve satisfactory performance.

## 4.3 Numerical Results

In this section, we evaluate the performance, via computer simulations, of the proposed OIPM algorithm for the weighted round robin (WRR) service discipline. In our experiments, the traffic model is characterized by a 2-state Markov Modulated Bernoulli Process (2-MMBP) alternating between active and idle states. The traffic source will generate a cell every time slot when it is in the active state. A packet consists of the cells generated by consecutive active states. We assume that there is no correlation between different packets. Moreover, the destination of each packet is uniformly distributed among the output ports and every packet is equally likely to be any traffic class.

As mentioned above, the WDRRM algorithm can achieve fairness when all output ports have identical weights. Before describing the WDRRM algorithm, we briefly review the dual round robin matching (DRRM) algorithm proposed in [62]. In the DRRM algorithm, each input maintains *N* VOQs for *N* output ports. An input arbiter at each input selects a nonempty VOQ according to the round-robin service discipline. After the selection, each input port sends one request, if any, to an output arbiter. An output arbiter at each output receives up to *N* requests and chooses one of them based on the round-robin service discipline and sends a grant to the chosen input port. For the WDRRM algorithm modified in this study, each input port maintains *K* queues for the *K* traffic classes. For starvation free, we assume that only one request is selected from *K* queues based on the WRR scheme and sent from an input port to an output port. An output port will select a request, if there are multiple, based on the round robin scheme and sent a grant message back to the corresponding input port.

Simulations are performed for $N = 32$, assuming that the mean packet length is 8 cells. Figs. 4.3 illustrates the throughput performance of the WDRRM algorithm and our proposed OIPM algorithm for $K = 3$. The service discipline is weighted round robin with $w_1 = 5$, $w_2 = 3$, and $w_3 = 2$.

It can be seen that both OIPM and WDRRM algorithms achieve weighted fairness among traffic classes. As mentioned above, the WDRRM algorithm is not able to achieve weighted fairness if different output ports have different weights. The curves shown in Fig. 4.4 are for the case where output ports 1 to 16 have weights $w_{11} = 5$, $w_{12} = 3$, and $w_{13} = 2$ and output ports 17 to 32 have weights $w_{21} = 4$, $w_{22} = 4$, and $w_{23} = 2$. As can be seen in this figure, our proposed OIPM algorithm still achieves weighted fairness.

When the switching fabric runs as fast as the input/output line rate, the maximum throughput of our proposed OIPM algorithm is only 0.520. Next, we study the effects of speedup factors and results are shown in Fig. 4.5. The packets of different classes arrived at each input port have the same arrival rate and the destination of each packet is uniformly distributed among the output ports. The incoming traffic arrived at input ports may be overloaded up to twice the output departure rate for testing the fairness. Output ports 1 to 16 have weights $w_{11} = 5$, $w_{12} = 3$, and $w_{13} = 2$ and output ports 17 to 32 have weights $w_{21} = 4$, $w_{22} = 4$, and $w_{23} = 2$. We found that our proposal achieves approximate 100% throughput and guarantees fairness when the switching fabric runs with a speedup factor of 2.

## 4.4 Summary

We have presented a novel output initiated parallel matching algorithm for large-scale input buffered switched. In our proposed architecture, packets are buffered at input ports and queues are maintained at output ports. The advantages of maintaining queues at output ports include much fewer queues compared with VOQ switch, weighted fairness among traffic classes, in-order packet delivery, and starvation free. Through computer simulations, we found that the maximum throughput of our proposed OIPM algorithm is around 0.520. The throughput can be improved to approximate 100% when the switch runs with a speedup factor of 2.

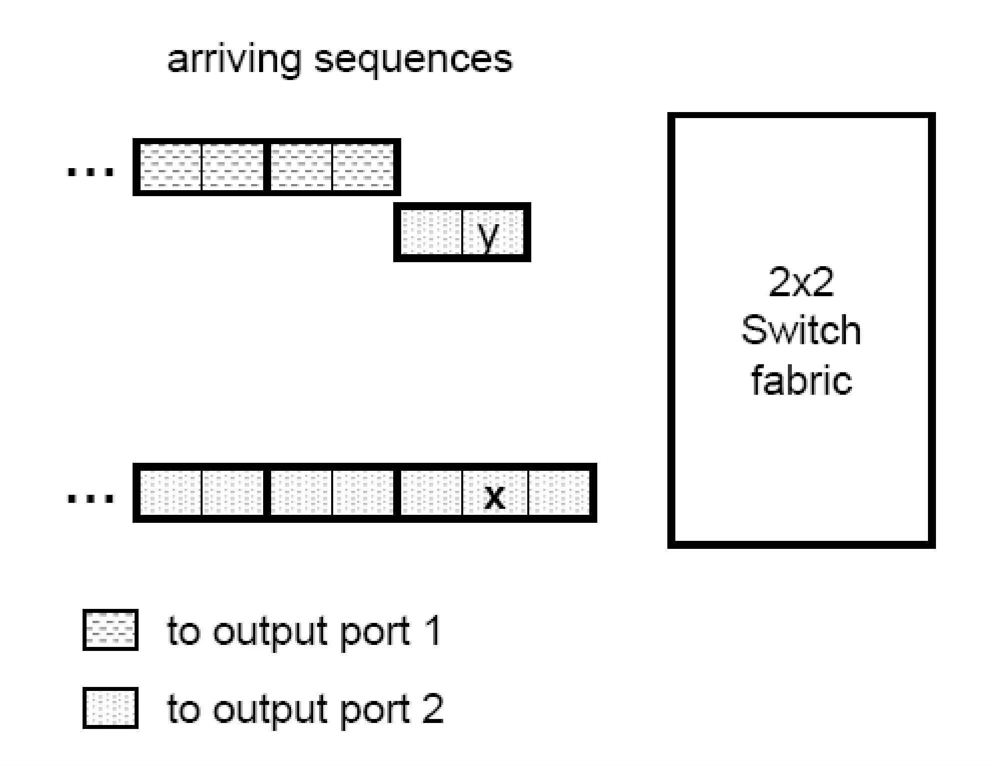


Fig. 4.1 Example of starvation when queues are placed at input ports.

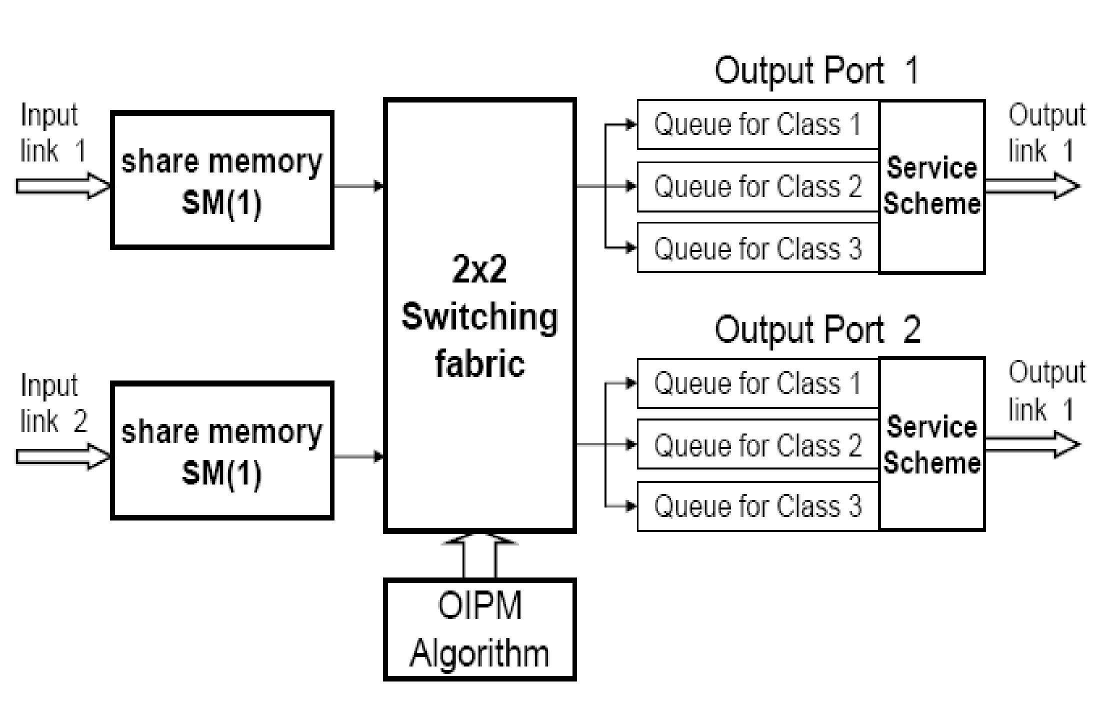


Fig. 4.2 The architecture of a 2x2 input buffered switch using OIPM algorithm with three traffic classes ($K = 3$).

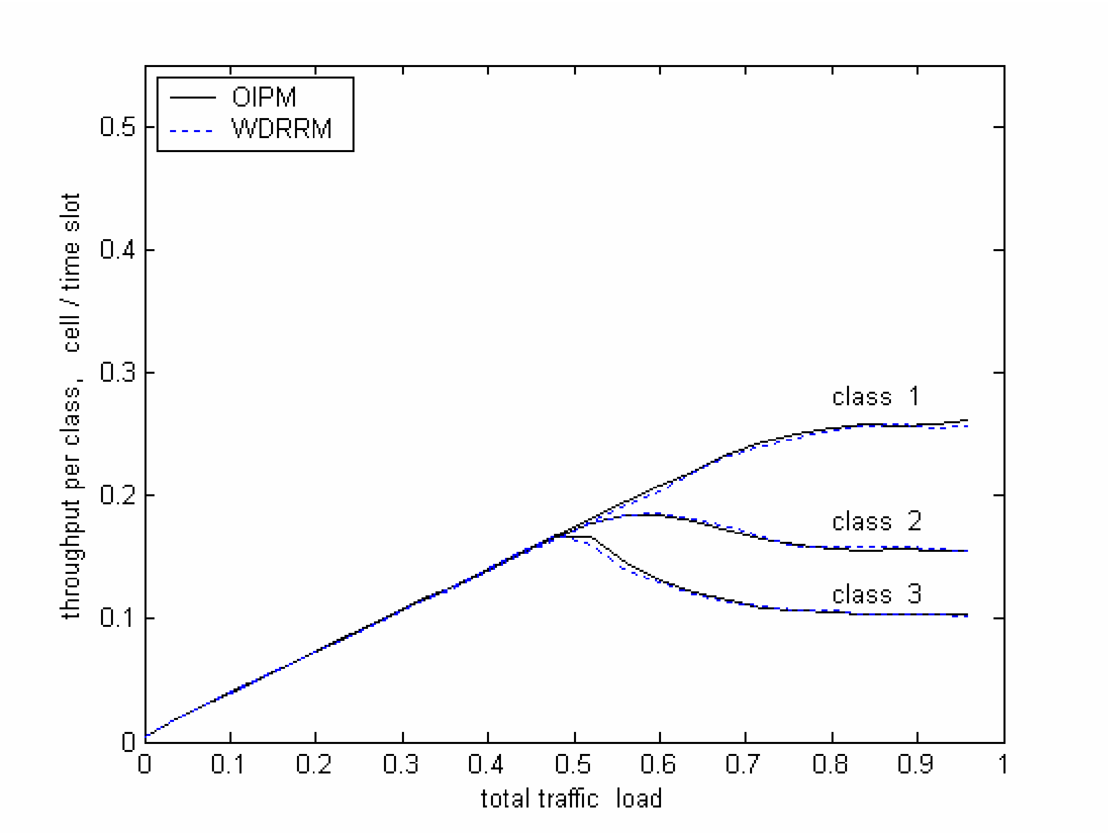


Fig. 4.3 Weighted fairness of the proposed OIPM and the WDRRM algorithm when all output ports have identical weights.

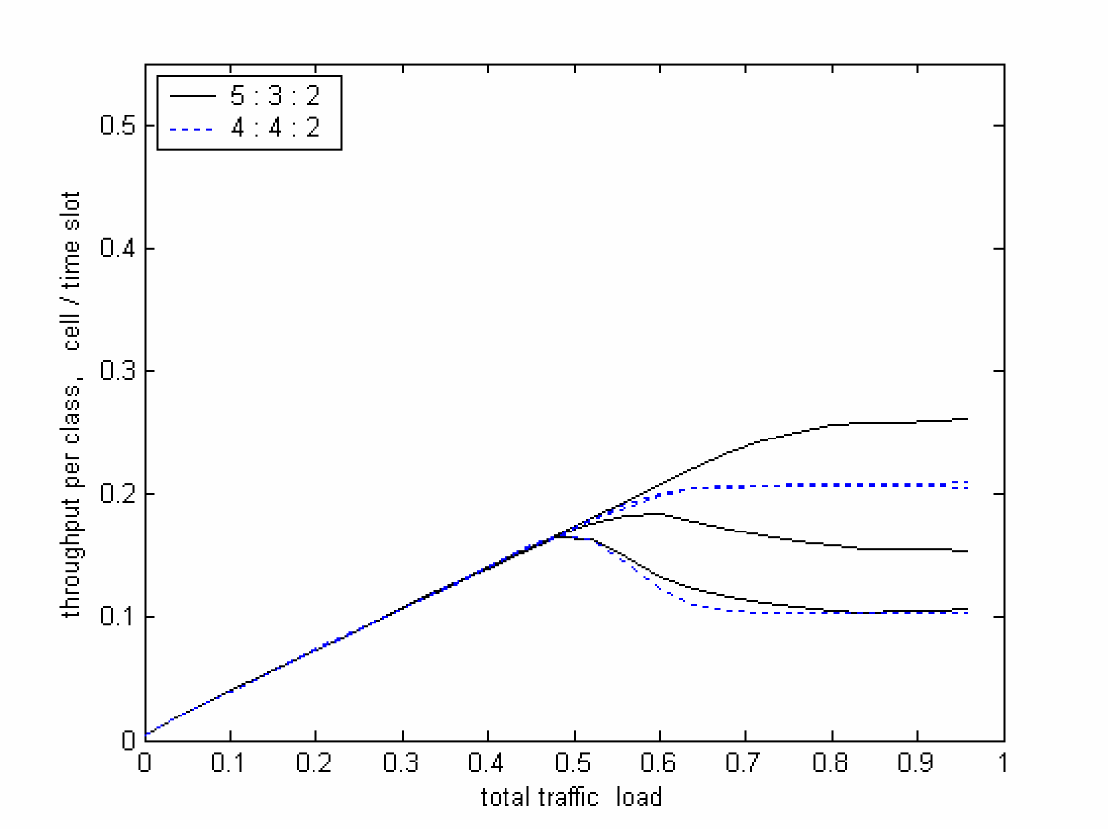


Fig. 4.4 Weighted fairness of the proposed OIPM algorithm when different output ports have different weights.

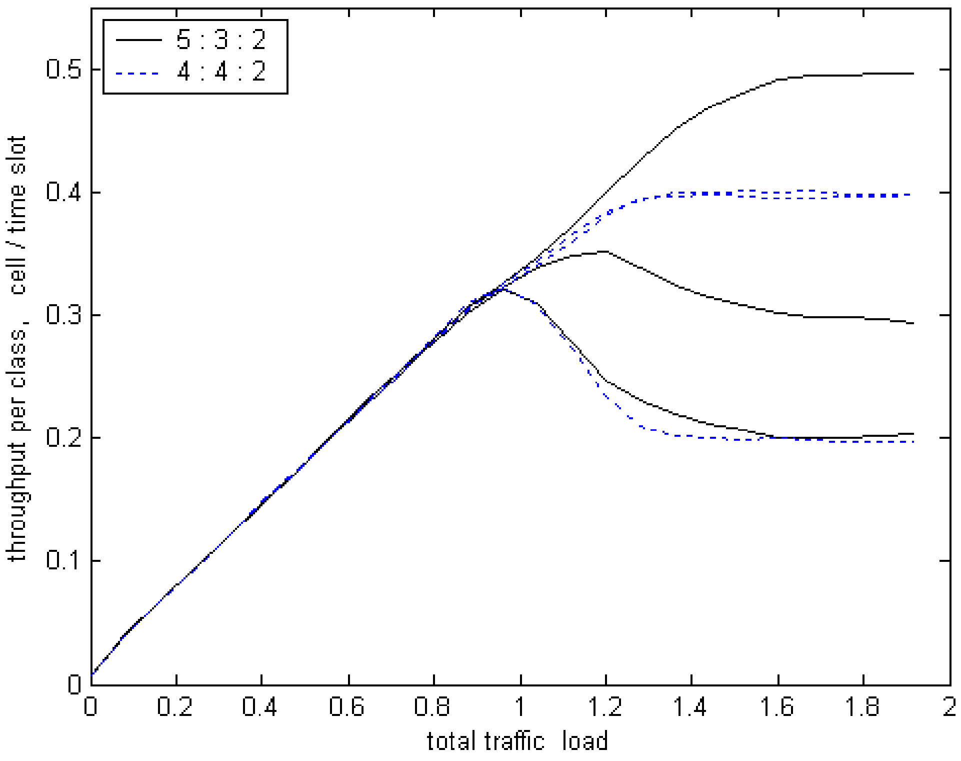


Fig. 4.5. Weighted fairness of the proposed OIPM algorithm when the switching fabric runs with a speedup factor of 2.

# Chapter 5
# Conclusions

In the first two part of this thesis, we have proposed two approximate LCF/MUF matching algorithms for the CIOQ switch with a speedup factor of 2 under fixed-length packet traffics and with a speedup factor of 5 under variable-length packet traffics, respectively. Our proposal makes the LCF/MUF algorithm much more feasible for implementation because it needs only single-iteration matching and the calculation of cushions of packets is largely simplified. Numerical results show that the performance of the proposed approximate matching is very close to that of the LCF/MUF algorithm under uniform, non-uniform, and correlated traffic models for offered load up to 0.9. Moreover, our simulation results show that the performance of the proposed approximate LCF/MUF algorithm is significantly better than that of a simple matching algorithm such as FIRM or *i*SLIP. The price paid for the LCF/MUF algorithm is higher hardware cost because it requires cushion calculations.

We have also investigated the performance of a CIOQ switch with finite input and output buffers since the buffer size at each input and output port of the exact emulated CIOQ switch is assumed to be of infinite size. We found, via computer simulations, that a CIOQ switch with sufficient input and output buffers can mimic an OQ switch quite well under both uniform and correlated traffics.

Based on the extensive simulation results, we believe that, with a moderate speedup, a CIOQ switch can exactly emulate an OQ switch even for variable-length packets. However, we are not able to determine a necessary and sufficient speedup number for exact emulation when using our proposal. Rigorous mathematical proof of exact emulation obviously is an interesting further research topic. Possible realization of the proposed PB-LCF/MUF algorithm is another one that can be further studied.

In the last part of this thesis, we presented a novel output initiated parallel matching algorithm for large-scale input buffered switches. In our proposed architecture, packets are buffered at input ports and queues are maintained at output ports. The advantages of maintaining queues at output ports include much fewer queues compared with VOQ switch, weighted fairness among traffic classes, in-order packet delivery, and starvation free. Through computer simulations, we found that the maximum throughput of our proposed OIPM algorithm is around 0.520 which is slightly smaller than 0.586. One can speedup the switching fabric to improve system performance. The throughput can be improved to approximately 100% when the switching fabric runs with a speedup factor of 2.

# Bibliography

1 D. Ferrari, “Client requirements for real-time communication services,” *IEEE communications magazine*, 28(11), pp. 65-72, Nov. 1990.

2 J. Kurose, “On computing per-session performance bounds in high-speed multi-hop computer networks,” *Proc. ACM Sigmetrics ’92*, 1992.

3 H. Zhang and E. Knightly, “Providing end-to-end statistical performance guarantees with interval dependent stochastic models,” *Proc. ACM Sigmetrics ’94*, pp. 211-220, Nashville, TN, May 1994.

4 D. Ferrari and D. Verma, “A scheme for real-time channel establishment in wide-area networks,” *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 368-379, Apr. 1990.

5 Q. Zheng and K. Shin, “On the ability of establishing real-time channels in point-to-point packet-switcing networks,” *IEEE Trans. Commun.*, pp. 1096-1105, Mar. 1994.

6 L. Zhang, “Virtual clock: A new traffic control algorithm for packet switching networks,” *Proc. ACM SIGCOMM ’90*, pp. 19-29, Philadelphia, PA, Sep. 1990.

7 A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” *Proc. ACM SIGCOMM ’89*, pp. 3-12.

8 A. Parekh, and R. Gallager, “A generalized processor sharing approach to flow control – the single node case,” *Proc. IEEE INFOCOM ’92*.

9 S. Golestani, “A self-clocked fair queueing scheme for broadband applications,” *Proc. IEEE INFOCOM ’94*, pp. 636-646.

10 J. C. R. Bennett and H. Zhang, “WF2Q: Worst-case fair weighted fair queueing,” *Proc. IEEE INFOCOM ’96*, pp. 120-128. San Francisco, CA, Mar. 1996.

11 D. Verma, H. Zhang, and D. Ferrari, “Guaranteeing delay jitter bounds in packet switching networks,” *Proc. Tricomm ’ 91*, pp. 35-46, Chapel Hill, North Carolina, Apr. 1991.

12 S. Goleatani, “A stop-and-go queuing framework for congestion management,” *Proc. ACM SIGCOMM ’94*, pp8-18, Philadelphia, PA, Sep. 1990.

13 C. Kalmanek, H. Kanakia, and S. Keshav, “Rate controlled servers for very hogh-speed networks,” *Proc. IEEE GLOBCOM ’90*, pp. 300.3.1-300.3.9, San Diego, California, Dec. 1990.

14 H. Zhang and D. Ferrari, “Rate-controlled static priority queuing,” *Proc. IEEE INFOCOM ’93*, pp. 227-236, San Francisco, California, Apr. 1993.

15 H. Zhang, “Service disciplines for guaranteed performance service in packet-switching networks,” *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374-1396, Oct. 1995

16 F. M. Chiussi, D. A. Khotimsky, and S. Krishnan, “Generalized inverse multiplexing of switched ATM connections,” *Proc. IEEE GLOBECOM ’98*, The Bridge to Global Integration, Sydney, Australia, Nov. 1998.

17 S. Iyer, A. Awadallah, and N. McKeown, “Analysis of a packet switch with memories running slower than the line-rate,” *Proc. IEEE INFOCOM 2000*, pp. 529-537.

18 S. Iyer and N. McKeown, “Making parallel switches practical,” *Proc. IEEE INFOCOM 2001,* pp. 1680-1687.

19 D. Khotimsky and S. Krishnan, “Stability analysis of a parallel packet switch with bufferless input demultiplexors,” *Proc. IEEE ICC 2001*, pp. 100-111.

20 M. Karol, M. Hluchyj, and S. Morgan, “Input verse output queuing on a space division switch,” *IEEE Trans. Commun.*, vol. 35, pp. 1347-1763, Dec. 1987.

21 S. Q. Li, “Performance of a nonblocking space-division packet switch with correlated input traffic,” *Proc. IEEE GLOBECOM ’89*, pp. 1754-1763.

22 M. Chen and N. D. Georganas, “A fast alforithm for multi-channel/port traffic scheduling,” *Proc. IEEE Supercom/ICC ’94*, pp. 96-100.

23 A. Huang and S. Knauer, “Starlite: A wideband digital switch,” *Proc. IEEE GLOBECOM ’84*, pp. 121-125.

24 M. Karol and M. Hluchyi, “Queueing in high-performance packet switching,” *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1587-1597, Dec. 1988.

25 Y. Tamir and G. Franier, “High performance multi-queue buffers for VLSI communication switches,” *Proc. 15th Annu. Symp. Comput. Arch.*, June 1988, pp. 343-354.

26 M. Ali and H. Nguyen, “ A neural network implementation of an input access scheme in a high-speed packet switch,” *Proc. IEEE GLOBECOM ’89*, pp. 1192-1196.

27 T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, “High speed switch scheduling for local area networks,” *IEEE/ACM Trans. Comput. Syst.*, vol. 11, no.4, pp. 319-352, Nov. 1993.

28 R. O. LaMaire and D. N. Serpanos, “Two-dimensional round-robin schedulers for packet switches with multiple input queues,” *IEEE/ACM Trans. Networking*, vol. 1, pp. 471-482, Oct. 1993.

29 N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, “The Tiny tera: A small high-bandwidth packet switch core,” *IEEE Micro.,* Vol. 17, pp.

26-33, Jan.-Feb. 1997.

30 Y. Tamir and H.-C. Chi, “Symmetric crossbar arbiters for VLSI communication switches,” *IEEE Trans. Parallel Dist. Syst.*, vol. 4, pp. 13-27, 1993.

31 N. McKeown, V. Anantharam, and J. Walrand, “Achieving 100% throughput in an input-queued switch,” *Proc. IEEE INFOCOM ‘96*, pp. 296-302.

32 J. E. Hopcroft and R. M. Karp, “An n5/2 algorithm for maximum matching in bipartite graphs,” *Society for Industrial and Applied Mathematics J. Comput.*, vol. 2, pp.225-231, 1973.

33 R. E. Tarjan, “Data structures and network algorithms,” Bell Labs, 1983.

34 D. Stiliadis and A. Verma, “Providing bandwidth guarantees in an input-buffered crossbar switch,” *Proc. IEEE INFOCOM ‘95*, pp. 960-968, 1995.

35 A. Mekkittikul and N. McKeown, “A practical scheduling algorithm to achieve 100% throughput in input-queued switches,” *Proc. IEEE INFOCOM ‘98*, pp. 792-799, 1998.

36 N. McKeown, “The *i*SLIP scheduling algorithms for input-queued switches,” *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-201, Apr. 1999.

37 D. N. Serpanos and P. I. Antoniadis, “FIRM: a class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues,” *Proc. IEEE INFOCOM 2000*, pp. 548-555.

38 A. Charny, “Providing QoS guarantees in input-queued crossbar switches with speedup,” Ph.D. dissertation, Aug. 1998, MIT.

39 J. G. Dai and B. Prabhakar, “The throughput of data switches with and without speedup,” *Proc. IEEE INFOCOM 2000.*

40 P. Giaccone, E. Leonardi, B. Prabhakar, and D. Shah, “Delay performance of high-speed packet switches with low speedup,” *Proc. IEEE GLOBECOM 2002*, Taipei, Taiwan, Nov. 2002.

41 M. Yang and S. Q. Zheng, “Efficient scheduling for CIOQ switches with space division multiplexing speedup,” *Proc. IEEE INFOCOM 2003*.

42 C. Y. Chang, A. J. Paulraj, and T. Kailath, “A broadband packet switch architecture with input and output queuing,” *Proc. IEEE GLOBECOM ’94*, pp.448-452.

43 I. Iliadis and W. E. Denzel, “Performance of packet switches with input and output queuing,” *Proc. IEEE ICC ’90*, Atlanta, GA, Apr. 1990, pp.747-53.

44 A. L. Gupta and N. D. Georganas, “Analysis of a packet switch with input and output buffers and speed constraints,” *Proc. IEEE INFOCOM ’91*, Bal Harbour, FL, Apr. 1991, pp.694-700.

45 Y. Oie, M. Murata, K. Kubota, and H. Miyahara, “Effect of speedup in

nonblocking packet switch," *Proc. IEEE ICC ’89*, Boston, MA, Jun. 1989, pp.410-14.

46 A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-queued crossbars with speedup," *Proc. IEEE IWQoS ’98*, pp. 235-244.

47 P. Krishan, N. S. Patel, A. Charny, and R. Simcoe, "On the speedup required for work-conserving crossbar switches," *Proc. IEEE IWQoS ‘98*, Napa, California, USA, pp. 225-234, May 1998.

48 I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," *Proc. IEEE IWQoS ‘98*, Napa, California, USA, pp. 218-224, May 1998.

49 S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output qeueueing with a combined input/output-queued switch," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, pp. 1030-1039, Jun. 1999.

50 T. H. Lee, Y. W. Kuo, and J. C. Huang, "Quality of service guarantee in a combined input output queued switch," *IEICE Trans. Commun.*, vol. E83-B, no. 2, pp. 190-195, Feb. 2000.

51 M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet scheduling in input-queued cell-based switches," *Proc. IEEE INFOCOM 2001*, pp. 1085-1094.

52 S. H. Moon and D. K. Sung, "High-performance variable-length packet scheduling algorithm for IP traffic," *Proc. IEEE GLOBECOM 2001*, pp. 2666-2670.

53 G. Nong and M. Hamdi, "Burst-based scheduling algorithms for non-blocking ATM switches with multiple input queues," *IEEE Communications Lett.*, vol. 4, no. 6, Jun. 2002.

54 X. Zhang and L. N. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *IEEE J. Select. Areas Commun.*, vol. 21, no. 4, May. 2003.

55 R. Rojas-Cessa, E. Oki, and H. J. Chao, "CIXOB-*k*:Combined input-crosspoint-output buffered packet switch," *Proc. IEEE GLOBECOM 2001*, pp. 2654-2660.

56 S. C. Liew, "Performance of various input-buffered and output-buffered ATM switch design principles under bursty traffic : simulation study," *IEEE Trans. Commun.*, vol. 42, pp. 1371-1379, Feb/Mar/Apr 1994.

57 C. S. Donpaul, H. Zhang, "Implementing distributed packet fair queuing in a scalable switch architecture," *Proc. IEEE INFOCOM ‘98*, pp. 282-290.

58 http://www.nlnar.net/NA/Learn/packetsizes.html.

59 J. Heinanen, “Multiprotocol encapsulation over ATM adaptation layer 5,” RFC 1483, Jul. 1993.

60 N. Ni and L. N. Bhuyan, “Fair scheduling and buffer management in internet routers,” *Proc. IEEE INFOCOM*, New York, pp. 1141-1150, Jun. 2002.

61 M. Shreedhar and G. Varghese, “Efficient fair queuing using deficit round robin,” *Proc. IEEE SIGCOMM*, Boston, MA, Aug. 1995.

62 H. J. Chao and J. S. Park, “Centralized contention resolution schemes for a large-capacity optical ATM switch,” *Proc. IEEE ATM Workshop*, pp. 11-16, 1998.

# Vita

Ying-Che Kuo received the B.S. and M.S. degrees in Automatic Control Engineering from Feng Chia University, Taichung, Taiwan, ROC, in 1986 and 1988, respectively. He is currently pursing the P.h.D. degree in Communication Engineering at National Chiao Tung University, Hsinchu, Taiwan, ROC.

His principal areas of interest include scheduling and switch architecture.

# Publication List

## A. International Journal papers

[1] T. H. Lee and Y. C. Kuo, “Performance Evaluation of Combined Input Output Queued Switch with Finite Input and Output Buffers,” in LNCS (Lecture Note in Computer Science) book of Springer Verlag, Part 1, pp. 203-214, 2002.

[2] T. H. Lee and Y. C. Kuo, “A Parallel Matching Algorithm for CIOQ Switches with Multiple Traffic Classes,” IEE Proceedings on Communications, vol. 150, no. 5, pp. 354-360, 2003.

[3] T. H. Lee and Y. C. Kuo, “Packet-Based Scheduling Algorithm for CIOQ Switches with Multiple Traffic Classes,” accepted for publication in Elsevier Journal on Computer Communications.

## B. International Conference papers

[1] T. H. Lee and Y. C. Kuo, “Performance Evaluation of Combined Input Output Queued Switch with Finite Input and Output Buffers,” The 16th International Conference on Information Networking (ICOIN-16), Cheju Island, Korea, 2002.

[2] T. H. Lee and Y. C. Kuo, “A Matching Algorithm for CIOQ Switches with Multiple Levels of Service,” The 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Montreal, Canada, 2003.