CHAPTER 5

BASICS OF PACKET SWITCHING

An Internet Protocol (IP) router is a vital network node in today's packet switching network. It consists of multiple input/output ports. Packets from different places arrive at the input ports of the IP router. They are delivered to appropriate output ports by a switch fabric according to a forwarding table, which is updated by routing protocols. In the packet switch network, packets from various input ports may destine for the same output port simultaneously, resulting in output port contention. How to arbitrate and schedule packets when contention arises is an important and challenging issue in designing a high-performance scalable packet switch.

Traditional telephony networks use circuit switching techniques to establish connections. In the circuit switching scheme, there is usually a centralized processor that determines all the connections between input and output ports. Each connection lasts for an average of 3 minutes. For an $N \times N$ switch, there are at most N connections simultaneously and the time required to make each connection is 180 seconds divided by N. For instance, if N is 1000, the time to make a connection is at most 180 ms, which is quite relaxed for most of switch fabrics using current technology, such as CMOS crosspoint switch chips. However, for IP routers, the time needed to configure the input–output connections is much more stringent, and it is normally based on a fixed-length time slot. For instance, it could be as small as 64 bytes to cope with the smallest packet length of 40 bytes. For a 10 Gbit/s line, the slot time is about 50 ns. As the line bit rate increases and the number of the switch ports increases, the time needed for each connection is further reduced. As a result, it is impractical to employ centralized connection processors to establish connections between the inputs and the outputs.

Thus, for IP routers, we do not use a centralized connection processor to establish connections. But rather, a self-routing scheme is used to establish input–output connections in a distributed manner. In other words, the switch fabric has intelligence to route packets to proper output ports, based on the physical output port addresses attached in front of each packet. One of switch fabrics that have self-routing capability is, for instance, the banyan switch.

In addition to routing packets in the IP router, another important function is to resolve the output port contention when more than one packet is destined for the same output port at the same time. There are several contention-resolution schemes that have been proposed since the first packet switch architecture was proposed in early 1980s. One way to resolve the contention is to allow all packets that are destined for the same output port to arrive at the output port simultaneously. Since only one packet can be transmitted via the output link at a time, the remaining packets are queued at the output port. A switch with such architecture is called the output-buffered switch. For such a scheme, there is a need to operate the switch fabric and the memory at N times of the line speed, where N is the switch size. As the line speed or the switch port number increases, this scheme becomes practically impossible to implement. However, if all packets are not allowed to go to the same output port at a time, a scheduling scheme (or called arbitration scheme) is required to arbitrate the packets that are destined for the same output port. Packets that lose contention have to wait at the input buffer. A switch with such architecture is called the input-buffered switch.

The way of handling output contention will impact the switch performance, complexity, and implementation cost. Some schemes are implemented in a centralized manner and some in a distributed manner. The latter usually allows the switch to be scaled in both line rate and switch size while the former is for a smaller size switch and is usually less complex. In addition to scheduling packets at the inputs when resolving output port contention, packet scheduling is also implemented at the output of the switch. It schedules packet transmission order according to their priorities and/or allocated bandwidth to meet each connection's delay/throughput (i.e., quality of service) requirements. Packet scheduling to achieve quality of service requirements is discussed extensively in Chapter 4.

5.1 FUNDAMENTAL SWITCHING CONCEPT

Before discussing switching principles and architectures, some basic concepts in packet switching are first presented.

Switching and Routing. The basic goals for switching and routing are the same, both try to transfer information from one part to another. However, routing normally represents a large-scale perspective, where information is exchanged between two nodes that can be separated by a large distance in the network. While switching normally refers to the information exchanged within a network node. Moreover, routing usually needs the cooperation of other network nodes, and is based on a routing protocol, while switching is only a function of a single device, and is based on a forwarding table, switching architectures, and scheduling algorithms. Figures 5.1 and 5.2 illustrate examples for switching and routing.

Unicast and Multicast. Majority of network connections are point-to-point, which is called unicast. As a result, traffic that enters an input port of a switch fabric is only destined for an output port. However, for applications of video/audio conferences and data broadcasting, traffic from one source is sent to several different destinations. To support multicast, extra switching mechanism is required to copy data from one input port to multiple output ports. The multicast in a switch fabric is illustrated in Figure 5.3.

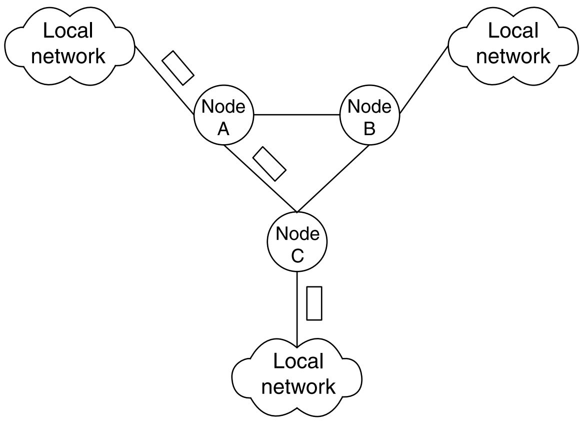


Figure 5.1 Typical routing example.

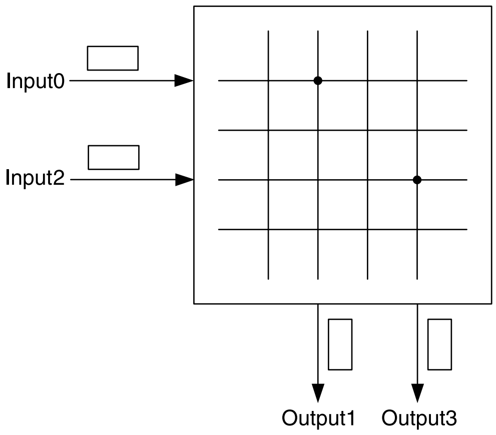


Figure 5.2 Typical switching example.

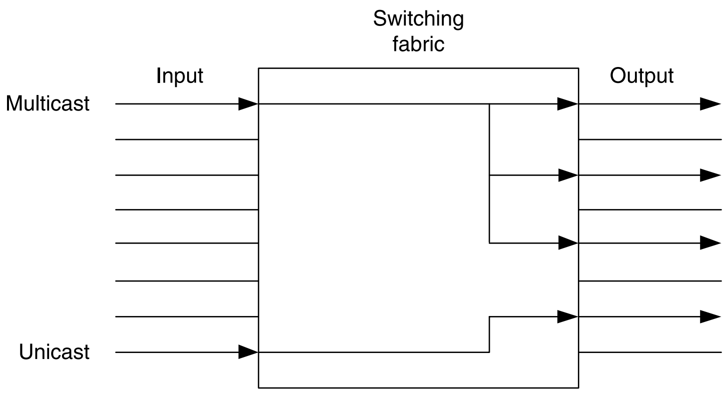


Figure 5.3 Multicast in switching fabric.

Throughput and Speedup. The throughput of a switch fabric is defined as the ratio of the average aggregated output rate to the average aggregated input rate when all the input ports carry 100% traffic at line rate. It is a positive value no more than one.

A speedup of k means that the internal forwarding rate of the switch fabric is k times the input line rate. So when a speedup exceeds one, buffers must be used at output ports. Speedup allows a switch fabric to forward more than one packet at the same time slot to the output port to alleviate output port contention, resulting in a higher throughput.

Blocking and Output Contention. Here we refer to blocking and output contention in space-division switching, since in time-division switching, traffic is multiplexed to avoid blocking.

In space-division switching, the main issue is the matching of input–output ports. Nonblocking means that when there is a request generated between an idle input port and an idle output port, a connection can always be set up. Note that an idle port refers to a port not connected nor requested. On the other hand, blocking means that it is possible that no connection can be set up between the idle input/output pair. Figure 5.4 shows an internal blocking in a three-stage Clos switch, when the idle input port 9 requests the idle output port 4 or 6, a connection cannot be set up because of internal blocking. Another example is shown in Figure 5.5, in a delta network, input port 0 is requesting output port 5 and input port 2 is requesting output port 4, but both need the same output port in the second stage to set up connections, which causes internal blocking. A typical nonblocking switch architecture is a crossbar switch.

Another problem that degrades the throughput of a switching fabric is the output contention. It is not because of the design of switching architecture, but due to the nature of bursty IP traffic. If more than one input port requests the same output port, only one can be granted, so the others have to wait. This is depicted in Figure 5.6. The output contention greatly affects the overall utilization of the switching fabric.

Cell-mode Switching and Packet-mode Switching. In ATM networks, information is packed into fixed-size cells. So in an ATM switch, scheduling takes place per cell

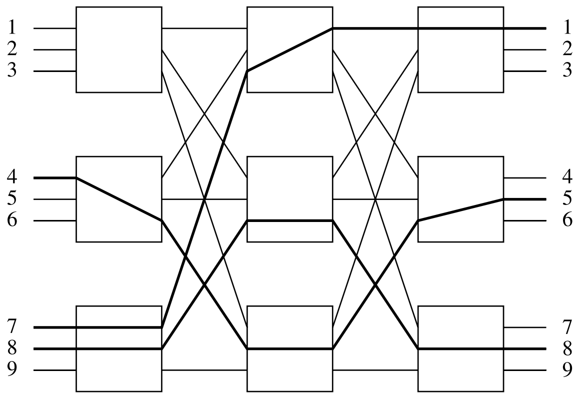


Figure 5.4 Internal blocking in a Clos network caused by a new connection between input 9 and output 4 or 6.

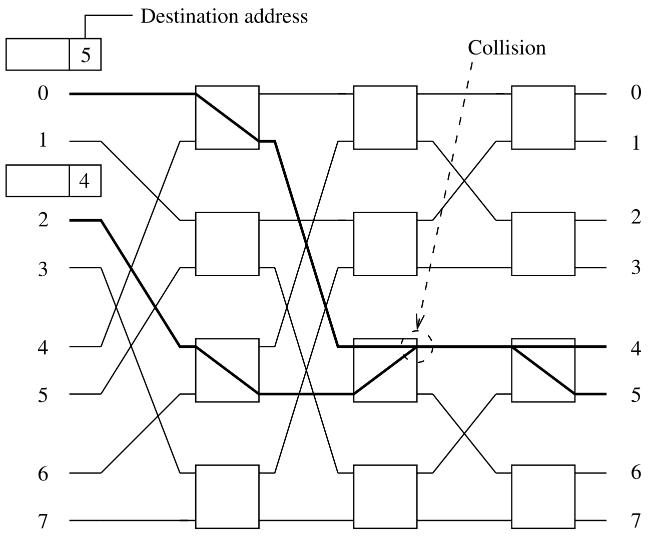


Figure 5.5 Internal blocking in a delta network.

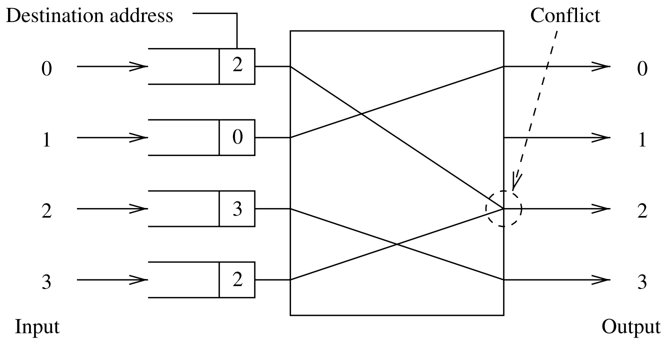
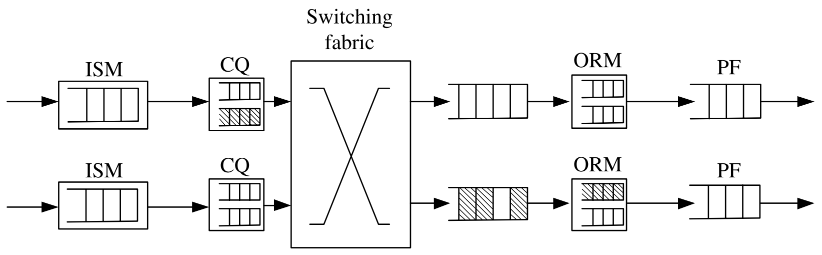


Figure 5.6 Output port contention.

time, which is called the time slot. But in IP networks, variable-length packets are adopted, and so traditional scheduling algorithms in the switching fabric of IP routers operate per packet basis. It is called pure packet-mode switching. However, because of the random distribution of packet lengths, the scheduler becomes too complex to design in large-scale routers.

To implement the ATM cell switching technology in IP routers, a popular architecture [1] has been proposed, as shown in Figure 5.7. In each input port, an input segmentation module (ISM) is employed to convert the variable-size IP packet into several fixed-size cells.[1] Proper stuffing is used when the remaining packet is shorter than a whole cell. Then the cells are buffered in the cell queues (CQs), for example,

[1]Cell is defined to be a fixed-length data unit used in the switch fabric. It consists of a cell header, used for routing in the switch fabric and payload. It needs not necessarily to be an ATM cell.



ISM: Input segmentation module
ORM: Output reassembly module
CQ: Virtual output cell queue
PF: Packet FIFO queue

Figure 5.7 Input queuing cell switch with segmentation and reassembly.

virtual output queues (VOQs), each associated with each output. More details about VOQs can be found in Chapter 7. Specific scheduling algorithms are implemented in selecting cells out of VOQs and switching them to correct output port. The output reassembly module (ORM) is required in each output port to reassemble cells of the same packet together. After the ORM, a packet FIFO (PF) is used to buffer the reassembled packets when the ORM has a speedup. This switching strategy is called cell-mode switching in IP routers.

Recent studies by Marsan et al. [2] and Ganjali et al. [3] have presented another scheduling strategy, packet-mode switching. A packet is divided into fixed-size cells like in cell-mode, but cells of the same packet are scheduled contiguously, not independently. The advantages of packet-mode switching are that the reassembly of packets become much easier. It also benefits from lower average packet delay as compared to cell-mode scheduling when the variance of packet length distribution is small.

5.2 SWITCH FABRIC CLASSIFICATION

The switch fabrics can be classified based on their switching techniques into two groups: time-division switching (TDS) and space-division switching (SDS). TDS is further divided into shared-memory type and shared-medium type. SDS is further divided into single-path switches and multiple-path switches, which are in turn further divided into several other types, as illustrated in Figure 5.8.

5.2.1 Time-Division Switching

Cells from different inputs are multiplexed and forwarded through a data path connecting all inputs and outputs. Typical time-division switching structures are the shared-memory switch (see Chapter 6) and the shared-medium switch. The time-division switching structure is limited by the internal communication bandwidth that should reach the aggregated forwarding bandwidth of all the input ports. However, this class of switch provides an advantage. Since every cell flows across the single communication structure, it can be easily extended to support multicast/broadcast operations.

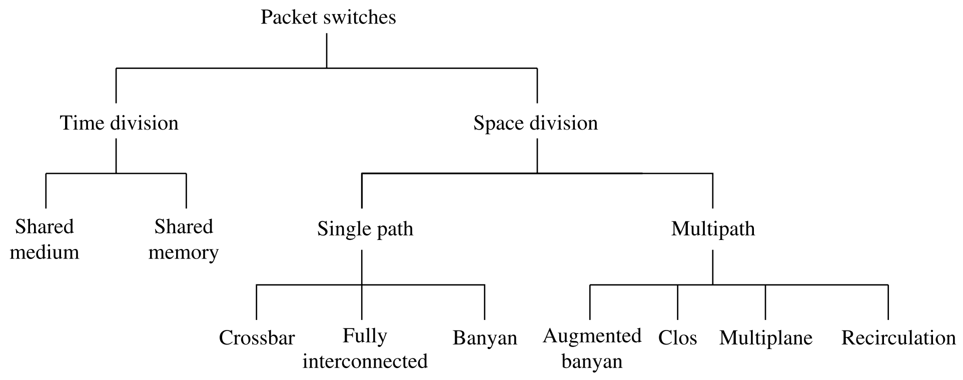


Figure 5.8 Classification of switching architecture.

In a shared-medium switch, shown in Figure 5.9, cells arriving at input ports are time-division multiplexed into a common high-speed medium, such as a bus or a ring, of bandwidth equal to N times the input line rate. An address filter (AF) and an output FIFO buffer are used in each output line that connects to the shared medium. The AF examines the header of each incoming cell and then accepts only the cells destined for itself. In this way, the multiplexed incoming cells are demultiplexed into correct output ports. Due to the separate FIFO buffer in each output port, the memory utilization of shared-medium switch is lower than the shared-memory structure. For a shared-medium switch, the switch size is limited by the memory read/write access time, within which N incoming and 1 outgoing cells in a time slot need to be accessed. Suppose the time for a cell slot is T_{cell} and the memory access time is T_{mem}, the number of switch size N can be calculated using the following equation:

$$(N + 1) \le \frac{T_{\text{cell}}}{T_{\text{mem}}}$$

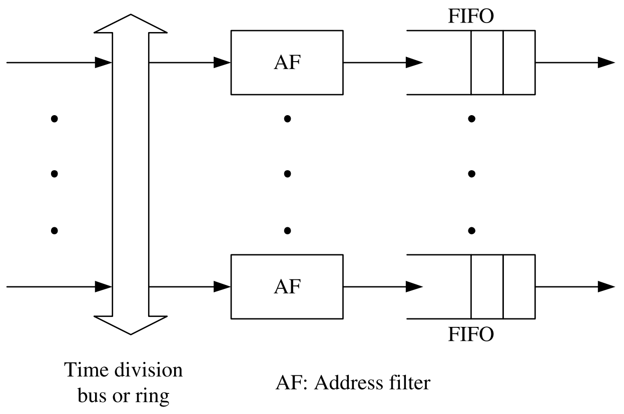


Figure 5.9 Shared-medium switching architecture.

For example, the cell slot time for a 64-byte packet at a 10 G interface is

$$T_{\text{cell}} = \frac{\text{Packetsize}}{\text{throughput}} = (64 \times 8)/(10 \times 10^9) = 51.2\,\text{ns}.$$

If $T_{\text{mem}} = 4\,\text{ns}$, a total of $\lfloor(51.2/4) - 1\rfloor = 11$ ports can be supported. In a shared-memory switch (see Chapter 6), same as shared-medium switches, cells from all the input ports are time-division multiplexed into the central logic. However, they are not forwarded immediately and flood to all the outputs as in the shared-medium structure. Instead, cells are buffered in a central memory, and then scheduled to the demultiplexer where cells destined for different output ports are separated. The shared memory structure is better in memory utilization than the separate FIFO structure in the shared-medium structure, but requires twice memory speed.

5.2.2 Space-Division Switching

The space-division switching (SDS) stands for the structure where multiple data paths are available between the input and the output ports and cells of the different input-output connections can be forwarded concurrently, when no blocking is present. The total switching capacity is then the product of the bandwidth of each path and the number of paths that can transmit cells simultaneously. However, in practice, the capacity of the switch is restricted by physical implementation constraints such as the device pin count and backplane connections.

The SDS switches are classified based on the number of available paths between any input–output pair. In single-path switches, only one path exists for any input-output pair, while in multiple-path switches there is more than one. The former has simpler routing control than the latter, but the latter has better connection flexibility and fault tolerance.

Single-Path Switches. In Figure 5.8, single path switches are classified into crossbar-based switches, fully interconnected switches, and banyan-based switches.

Crossbar Switch. A 4 × 4 crossbar switch is shown in Figure 5.10, where horizontal lines represent the inputs to the switch, and vertical lines represent the outputs. Basically, an $N \times N$ crossbar switch consists of a two-dimensional array of N^2 crosspoints, each corresponding to an input–output pair. Each crosspoint has two possible states: cross (default) and bar. A connection between input port i and output port j is established by setting the (i,j)th crosspoint switch to the bar state while letting other crosspoints along the connection remain in the cross state. The bar state of a crosspoint can be triggered individually by each incoming cell when its destination matches with the output address. No global information about other cells and their destinations is required. This property is called the self-routing property, by which the control complexity is significantly reduced in the switching fabric as the control function is distributed among all crosspoints. The crossbar switch allows the transferring of up to N cells with different input ports and different output destinations in the same time slot.

Crossbar switches have three attractive properties: internally nonblocking, simple in architecture, and modular. However, its number of the crosspoints grows as a function of N^2. If constructing larger multistage switch with small crossbar switch modules, the complexity can be improved to $O(N \log N)$. There is no internal blocking in a crossbar switch. As shown in Figure 5.11, there are four requests, input port 1 to

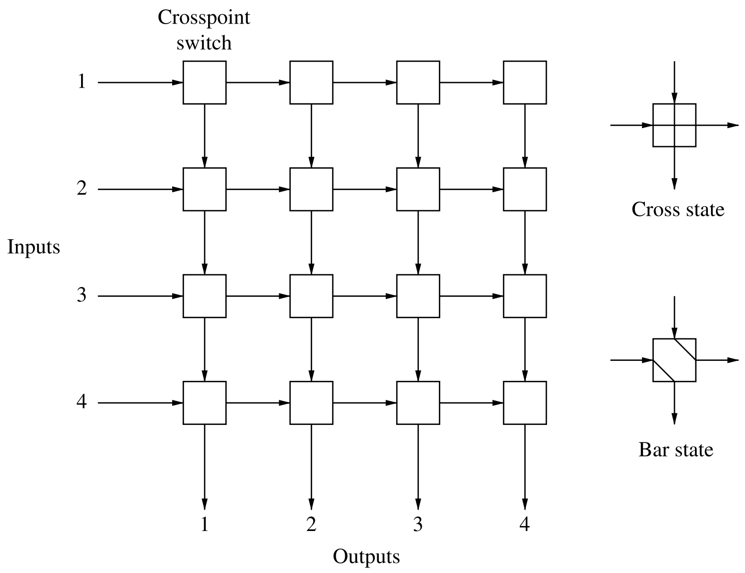


Figure 5.10 4 × 4 crossbar switch.

output port 3, input port 2 to output port 4, input port 3 to output 1 and input port 4 to output 3. Except the last request, the first three requests can be switched at the same time without internal blocking. However, there is a contention between the first and the last request marked by a dotted circle in Figure 5.11.

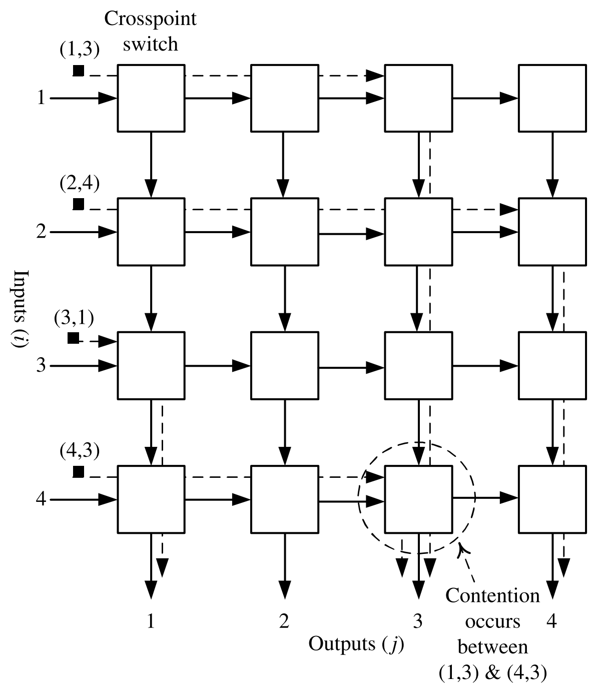


Figure 5.11 Switch examples for a 4 × 4 crossbar switch.

There are several possible locations for the buffers in a crossbar switch, such as (1) at the crosspoints in the switch fabric, (2) at the inputs of the switch, and (3) at the inputs and outputs of the switch. Each one has its advantages and disadvantages; they are detailed in Section 5.3.

Fully interconnected switches. Another kind of single-path space-division switches are the fully interconnected switches. In a fully interconnected switch, the complete connectivity between inputs and outputs is usually accomplished by means of N separate broadcast buses from every input port to all output ports, as shown in Figure 5.12. N separate buffers are required in such a switch, one at each output port. However, if each of these N output buffers in the fully interconnected switch is partitioned and dedicated to each input line, yielding N^2 dedicated buffers, it becomes topologically identical with the crosspoint-buffered switch and thus provides exactly the same performance and implementation complexity.

The fully interconnected switch operates in a similar manner to the shared medium switch. A cell from any input port is broadcast to every output port. The difference is that multiple cells from several input ports can be simultaneously broadcasted to every output port. Therefore, separate cell filters and dedicated buffers, one for each output port, are required to filter out the misdelivered cells and to temporarily store the properly destined cells.

Moreover, the fully interconnected switch is different from the shared medium switch in that the speed-up requirement caused by a sequential transmission over the shared medium is replaced by space overhead of the total N^2 separate broadcast buses. This is considered a disadvantage of the switch type. The advantages of the fully interconnected switch lie in its simple and nonblocking structure, similar to the crossbar-based switch.

Banyan-based switches. The banyan-based switches are also the single-path space-division switches. They are a family of self-routing switches constructed from 2×2 switching elements with a single path between any input–output pair. As shown in Figure 5.13, there are three isomorphic topologies: *delta*, *omega*, and

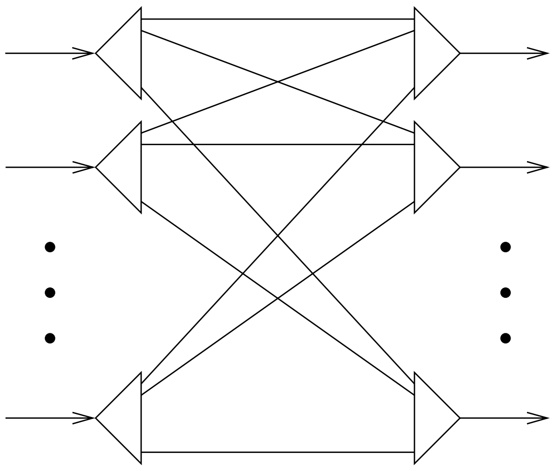


Figure 5.12 A fully interconnected switch.

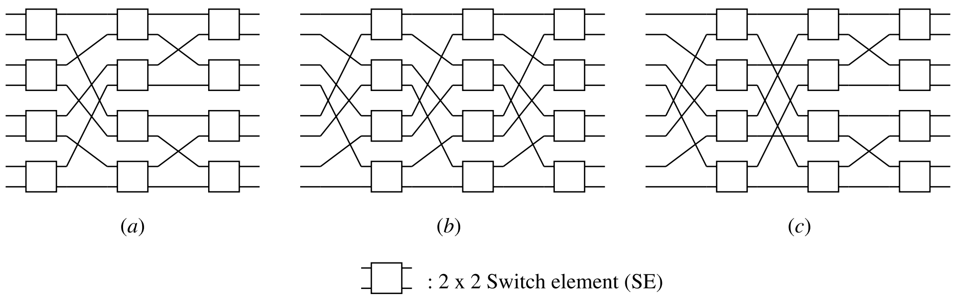


Figure 5.13 Three different topologies of banyan-based switches (*a*) Delta network; (*b*) Omega network; (*c*) Banyan network.

banyan networks, belonging to banyan-based family. All of them offer an equivalent performance. The detailed description of banyan switches is in Section 5.4.

Multiple-Path Switches. Multiple path switches are classified as *augmented banyan* switches, *Clos* switches, *multiplane switches*, and *recirculation* switches, as shown in Figure 5.14.

Augmented Banyan Switches. In a regular $N \times N$ banyan switch, cells pass through $\log N$ stages of switching elements before reaching their destinations. The augmented banyan switch, as illustrated in Figure 5.14*a*, refers to the banyan switch that has more stages than the regular banyan switch. In the regular banyan type switch, once a cell is deflected to an incorrect link and thus deviates from a predetermined unique path, the cell is not guaranteed to reach its requested output. Here, in the augmented banyan switch, deflected cells are provided more chances to be routed to their destinations

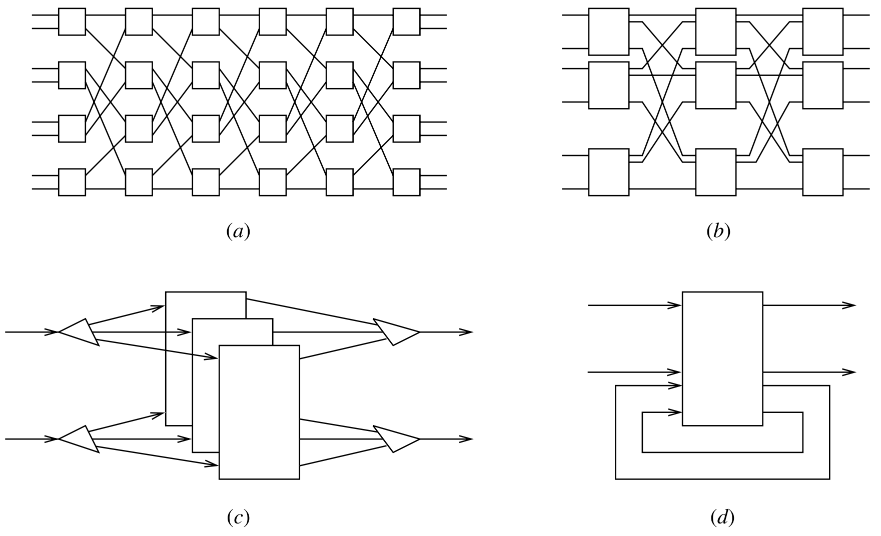


Figure 5.14 Multiple-path space-division switches (*a*) Augmented Banyan; (*b*) 3-stage clos; (*c*) Multiplane; (*d*) Recirculation.

again by using later augmented stages. When the deflected cells do not reach their destinations after the last stage, they will be discarded.

The advantage of the augmented banyan switch is that by adding augmented stages, the cell loss rate is reduced. The performance of the switch is improved. The disadvantage of this switch type is its complicated routing scheme. Cells are examined at every augmented stage to determine whether they have arrived at their requested output ports. If so, they will be sent to the output interface module. Otherwise, they are routed to the next stage and will be examined again. Another disadvantage is that the number of augmented stages needs to be sufficiently large to satisfy desired performance. Adding each augmented stage to the switch causes increased hardware complexity. The tandem banyan switch [4] and dual shuffle exchange switch [5] are examples of the augmented banyan switches.

Three-Stage Clos Switches. The structure of a three-stage Clos switch, as shown in Figure 5.14*b*, consists of three stages of switch modules. The first stage is used to distribute traffic, and in the middle stage, several parallel switch modules are built to provide multiple paths through the switches, and at last in the third stage cells from different switch modules of the middle stage are switched to the correct output ports. More details are described in Section 5.4.

Multiplane Switches. Figure 5.14*c* shows a multiplane switch. They refer to the switches that have multiple, usually identical, switch planes. Multiplane switches are mainly proposed as a way to improve system throughput. By using some mechanisms to distribute the incoming traffic loading, cell collisions within the switches can be reduced. Additionally, more than one cell can be transmitted to the same output port by using each switch plane so the output lines are not necessary to operate at a faster speed than that of the input lines. Another advantage of the multiplane switches is that they are used as a means of achieving reliability since the loss of a complete switch plane will reduce the capacity but not the connectivity of the switches. The parallel banyan switch and the Sunshine switch [6] are examples of the multiplane switches.

Recirculation Switches. Recirculation switches, as shown in Figure 5.14*d*, are designed to handle the output port contention problem. By recirculating the cells that did not make it to their output ports during the current time slot back to the input ports via a set of recirculation paths, the cell loss rate can be reduced. This results in system throughput improvement. The disadvantage of the recirculation switches is that they require a larger size switch to accommodate the recirculation ports. Also, recirculation may cause out-of-sequence errors. Some mechanisms are needed to preserve the cell sequence among the cells in the same connection. The representative recirculation switches are the Starlite switch [7] and the Sunshine switch [6].

5.3 BUFFERING STRATEGY IN SWITCHING FABRICS

Since the burst arrival of traffic are likely to contend for the output ports and/or internal links, buffers are required to temporarily store cells that lose the contention. Various buffering strategies are described below.

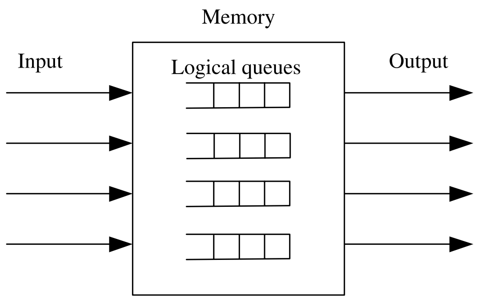


Figure 5.15 Shared-memory queuing structure.

5.3.1 Shared-Memory Queuing

The shared-memory queuing structure is shown by Figure 5.15, where a memory unit is used to store cells from all input ports. The memory contains N logical queues, one per output port, and each queue stores the cells destined for the corresponding output. It is thus said the memory is shared by all the output ports and the utilization of memory is maximized. However, the disadvantage is that in each time slot the memory must accommodate N concurrent write accesses and N concurrent read accesses, which limits the switch to a small size. Details are described in Chapter 6.

5.3.2 Output Queuing (OQ)

Another queuing strategy shown in Figure 5.16 is output queuing. Under this structure, cells are immediately forwarded to the destined output ports once they arrive at the inputs. Since the arrival rate of each output port is larger than the output line rate, buffers are needed at the output ports. The output queuing structure benefits from better QoS control ability, as the destining cells are always available at the output ports to be scheduled based on their priority levels, allocated bandwidth, and QoS requirements. However, this structure suffers from memory speed constraint and thus switch size limitation. This is because there can be up to N cells from different input ports destined for the same output port at the same cell time. The memory utilization is not as good as the shared-memory queuing's.

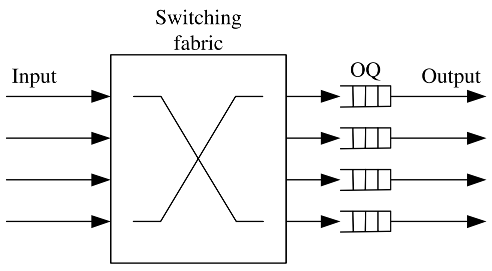


Figure 5.16 Output queuing structure.

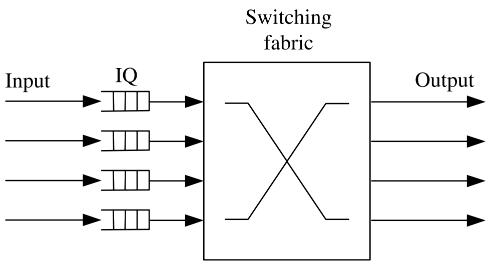


Figure 5.17 Input queuing structure.

5.3.3 Input Queuing

Due to the switch size limitation of the output queuing structure, the input queuing structure has attracted much more attention in recent years. As shown in Figure 5.17, pure input queuing has one FIFO queue per input port to buffer cells. A scheduling algorithm is required to match input ports with output ports in each time slot, as only one cell heading to an output port can be forwarded per time slot. One problem of input queuing is the so-called head-of-line (HOL) blocking. For instance, in Figure 5.18 input port 0 cannot send its head-of-line cell to output port 2 because of losing output contention to input port 3. However, the cell queued at input port 0 behind the HOL cell cannot be transmitted even if the destined output port 1 is idle. This severely degrades the switch throughput. It is proved that under uniform traffic arrival, the throughput of an input buffered switch is at most 58.6 percent [8, 9]. Several solutions such as channel grouping and windowing were proposed to increase the throughput.

5.3.4 Virtual Output Queuing (VOQ)

To eliminate HOL blocking, the virtual output queuing (VOQ) structure has been widely used. In Figure 5.19, same as the input queuing structure, cells are buffered at input ports. But the input buffer is divided into N logical queues, called VOQ, storing the cells destined for the associated output port. The HOL cell of each VOQ is scheduled in current cell time, comparing to the input queuing structure, only one HOL cell of each input can be forwarded. However, since the logical queue number is N times the input queuing structure, scheduling algorithm is much more sophisticated. It was a very hot research topic, and many scheduling algorithms have been proposed [1, 10–13]. The main function is to

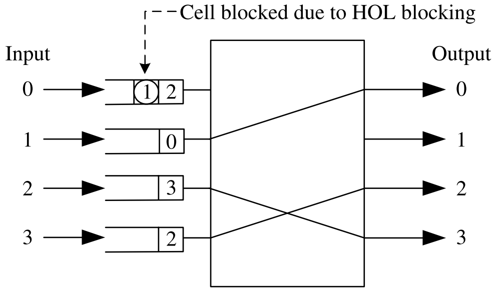


Figure 5.18 HOL blocking in input queuing structure.

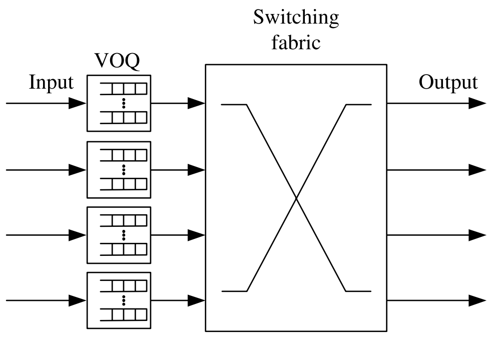


Figure 5.19 VOQ structure.

select an input–output matching from an $N \times N$ request matrix. Both delay/throughput performance and implementation complexity must be considered. More details can be found in Chapter 7.

5.3.5 Combined Input and Output Queuing

Another way to alleviate HOL blocking is to allow each output port to receive more than one cell in each time slot. That is to have the switch fabric operate S times of the line rate, where $1 < S < N$. Since the switch fabric operates faster than line rate, buffers must be placed at input and output ports to cope with the speed discrepancy between the line rate and the switch fabric speed.

In the original combined input and output queuing space (CIOQ) structure shown in Figure 5.20, each input and output port has a FIFO queue. It is proved that with a speedup of four, the CIOQ switch can achieve 99 percent throughput under independent uniform input traffic [14, 15].

An improved CIOQ structure [16, 17], shown in Figure 5.21, has N VOQ queues in the input buffer, while each output port still has a single FIFO queue. It is proved that under a slightly constrained traffic, this CIOQ structure can emulate a FIFO output queue (FIFO-OQ) switch with a speedup of 2 with bounded delay difference [18]. Note that the FIFO-OQ switch provides the best delay/throughput. More details can be found in Chapter 7.

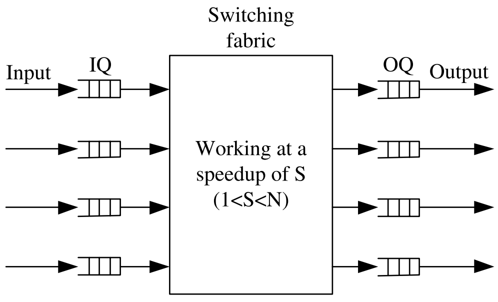


Figure 5.20 Combined input and output queuing structure with FIFO queue.

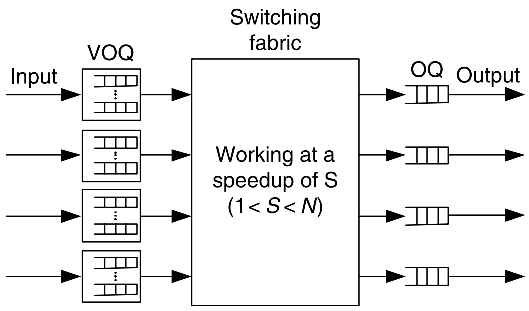


Figure 5.21 Combined input and output queuing structure with VOQ.

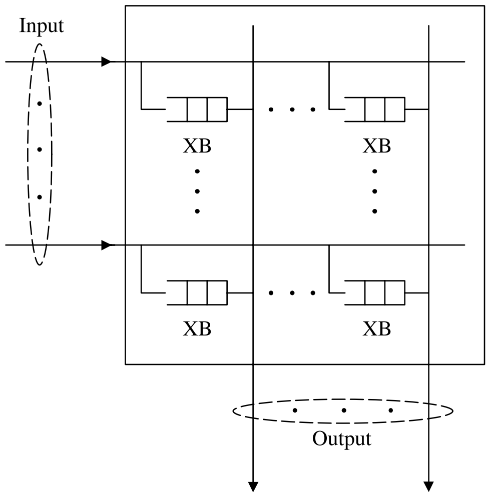


Figure 5.22 Crosspoint buffered crossbar structure.

5.3.6 Crosspoint Queuing

For a crossbar switch fabric, a buffer can be placed at each crosspoint, as shown in Figure 5.22. Incoming cells are first placed in the corresponding crosspoint buffer (XB), waiting to be transmitted to the output ports. An arbitrator of each output selects a cell among the XBs on the same column based on some scheduling scheme, for example, round-robin. This switch can achieve the same performance as the OQ switches because of no HOL blocking. However, since there are N^2 discrete buffers, the sharing effect is very poor. With limited chip space, only a few cells can be implemented. As a result, this switch usually combines VOQs at each input. Details can be found in Chapter 11.

5.4 MULTIPLANE SWITCHING AND MULTISTAGE SWITCHING

A large-scale switch is usually built with multiple switch modules (either in chips or boards) interconnected in a different structures. They are partitioned into single-stage or multistage switches.

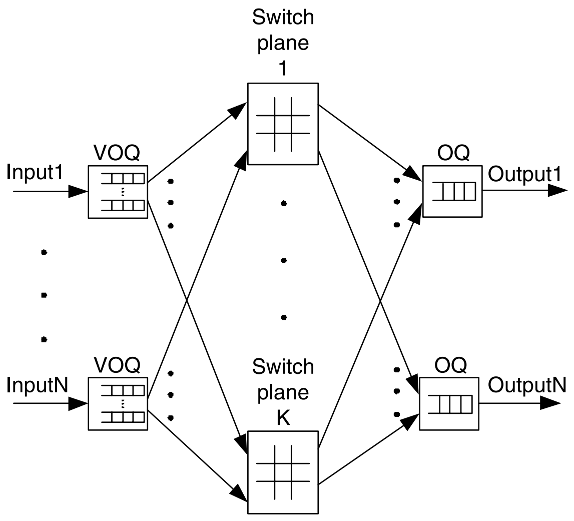


Figure 5.23 Parallel packet switching structure with crossbar.

Parallel Packet Switch. Figure 5.23 shows a single-stage switch fabric with multiple identical switch modules connected in parallel, which is called parallel packet switch (PPS) [19–21]. As a cell stream enters the switching fabric, it is distributed to K parallel planes. The challenging issue is how to find K cells from the N VOQs in each time slot such that the throughput is maximized. Since the center switch fabric does not have buffers, there shall be no cell out of order concern. It is proved that given a speedup of 2, the PPS structure can emulate a first-come-first-serve (FCFS) output-queued (OQ) switch; given a speedup of 3, the PPS can emulate any QoS queuing discipline, and even with no speedup, the PPS can mimic the FCFS OQ switch within a delay bound.

However, since the PPS use single-stage switch fabric, it cannot support large port count, while multi-stage switching can [22, 23]. The multistage switch structures connect the input and output ports in multiple stages and can be divided into two categories. One is the fully connected multistage network, where in every two adjacent stages, each module in the first stage connects to every module in the next stage. The other is the partially connected multistage network, where each module in the first stage only connects to a subset of the modules in the next stage.

Banyan-Based Switch. An example of the partially connected multistage network is the banyan network as shown in Figure 5.24. A typical example of the fully connected multistage structure is the Clos switch structure, as shown in Figure 5.25.

An $N \times N$ banyan network consists of $\log_b N$ stages of $b \times b$ switching elements. Each stage has multiple $b \times b$ switching elements whose inputs and outputs are labeled 0 to $b - 1$. The interstage connection patterns are such that only one path exists between any input of the first stage and any output of the last stage.

The banyan network has self-routing capability. When a cell enters in one of the input ports, it uses the output address as the routing information when traversing all the stages. For example, in Figure 5.24, a cell in input port 1 destined for output port 2, ('010') is

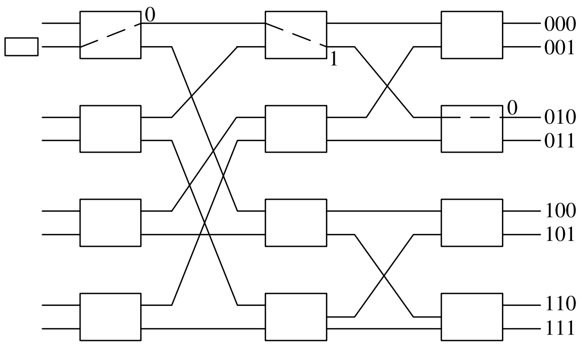


Figure 5.24 Banyan switching structure.

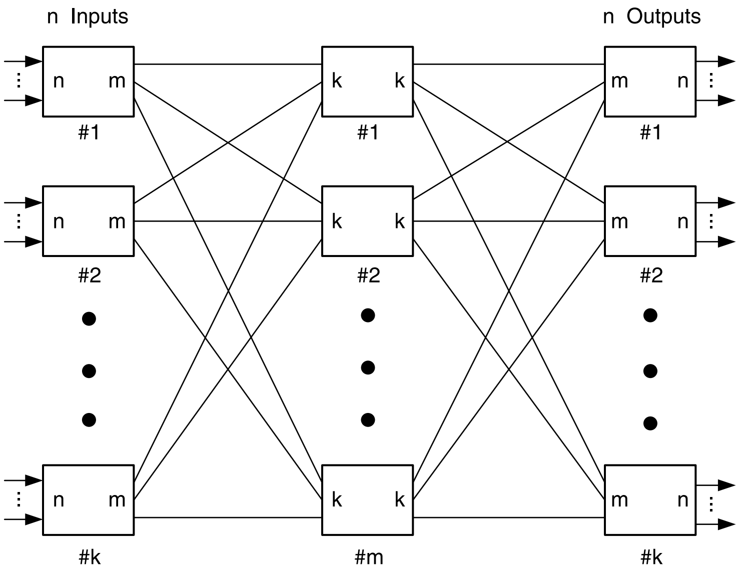


Figure 5.25 Clos switching structure.

rounded to the top link (0) of the first switch element, to the down link (1) of the second switch element, and to the top link (0) of the third switch element.

The banyan-based switch provides several advantages. It has fewer switch elements than the crossbar-based and the fully interconnected switch, $O(N \log N)$ versus $O(N^2)$. Self-routing property is also an attractive feature in that no control mechanism is needed for routing cells. Routing information is contained within each cell and it is used while the cell is routed along the path.

However, the main drawback of the banyan-based switch is that it is an internally blocking switch. Its performance degrades rapidly as the size of the switch increases. The performance may be improved if layer switching elements are employed (i.e., $b > 2$). This leads to the class of *delta-based* switches.

Clos-Network Switch. A crossbar switch is strictly *nonblocking* in that a path is always available to connect an idle input port to an idle output port. This is not always true for the Clos switch. Figure 5.26 shows a three-stage Clos switch with $N = 9, n = 3$, and $m = 3$. The bold lines indicate the existing connections. It can be seen that input port 9 cannot be connected to either output port 4 or 6, even though both of these output lines are idle. However, by rearranging the existing connections, the new connection request can be accommodated. Thus, the Clos network is rearrangably nonblocking.

By increasing the value of m (the number of middle-stage switch modules), the probability of blocking is reduced. To find the value of m needed for a strictly nonblocking three-stage switch, let us refer to Figure 5.27.

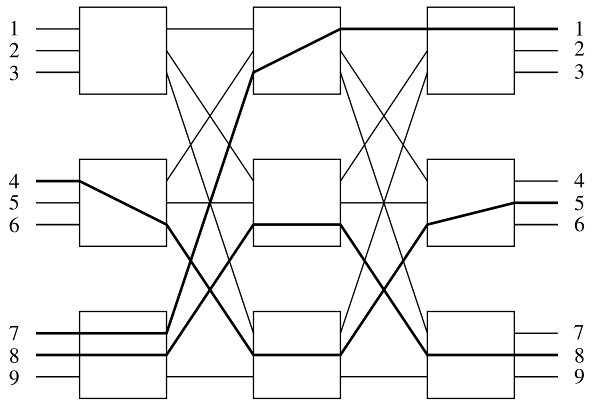


Figure 5.26 Example of internal blocking in a three-stage Clos switch.

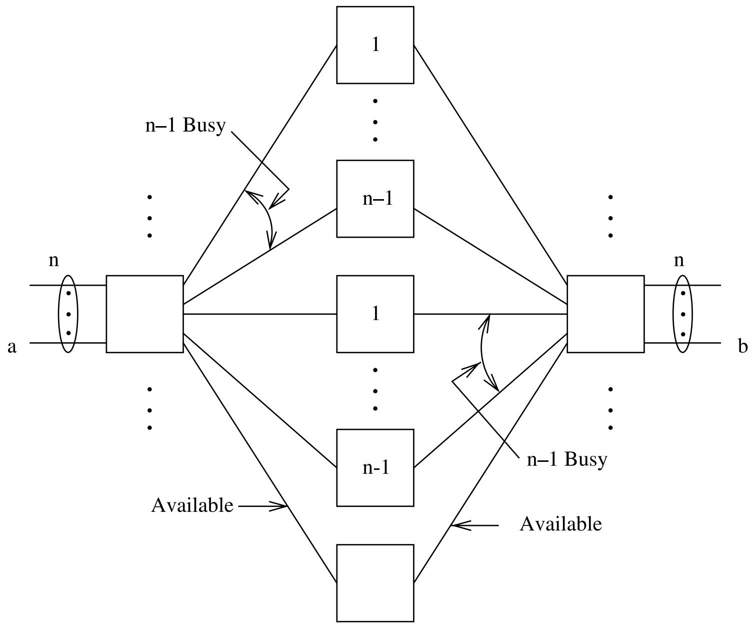


Figure 5.27 Nonblocking condition for a three-stage Clos switch.

We wish to establish a path from input port a to output port b. The worst case situation for blocking occurs if all of the remaining $n-1$ input lines and $n-1$ output lines are busy and are connected to different middle-stage switch modules. Thus a total of $(n-1)+(n-1)=2n-2$ middle-stage switch modules are unavailable for creating a path from a to b. However, if one more middle-stage switch module exists, an appropriate link must be available for the connection. Thus, a three-stage Clos switch will be nonblocking if

$$m \geq (2n-2)+1=2n-1$$

The total number of crosspoints N_x in a three-stage Clos switch when it is symmetric (i.e., $m=k$) is

$$N_x = 2Nm + m\left(\frac{N}{n}\right)^2$$

Substituting $m=2n-1$ into N_x, we obtain

$$N_x = 2N(2n-1)+(2n-1)\left(\frac{N}{n}\right)^2$$

for a nonblocking switch. For a large switch size, n is large. We can approximate

$$N_x \simeq 2N(2n)+2n\left(\frac{N}{n}\right)^2 = 4Nn + 2\left(\frac{N^2}{n}\right)$$

To optimize the number of crosspoints, differentiate N_x with respect to n and set the result to 0. The result will be $n \simeq (N/2)^{1/2}$. Substituting into N_x,

$$N_x = 4\sqrt{2}N^{3/2} = O(N^{3/2})$$

The three-stage Clos switch provides an advantage in that it reduces the hardware complexity from $O(N^2)$ in the case of the crossbar switch to $O(N^{3/2})$ while the switch could be designed to be nonblocking. Furthermore, it also provides more reliability since there is more than one possible path through the switch to connect any input port to any output port. The main disadvantage of this switch type is that some fast and intelligent mechanism is needed to rearrange the connections in every time slot so that internal blocking can be avoided. This would be the bottleneck when the switch size becomes large. In practice, it is difficult to avoid the internal blocking although the switch itself is nonblocking. Once the contention on the internal links occurs, the throughput is degraded. This can be improved by either increasing the number of internal links between switch modules or increasing the bandwidth of internal links so that more than one cell from the input module that are destined to the same third-stage module can be routed.

5.5 PERFORMANCE OF BASIC SWITCHES

This section describes performance of three basic switches: input-buffered, output-buffered, and completely shared-buffer.

5.5.1 Traffic Model

Bernoulli Arrival Process and Random Traffic. Cells arrive at each input in a slot-by-slot manner. Under Bernoulli arrival process, the probability that there is a cell arriving in each time slot is identical and is independent of any other slot. This probability is referred as the offered load of the input. If each cell is equally likely to be destined for any output, the traffic becomes uniformly distributed among the switch.

Consider the FIFO service discipline at each input. Only the HOL cells contend for access to the switch outputs. If every cell is destined for a different output, the switch fabric allows each to pass through to its respective output. If k HOL cells are destined for the same output, one is allowed to pass through the switch, and the other $k - 1$ must wait until the next time slot. While one cell is waiting its turn for access to an output, other cells may be queued behind it and blocked from possibly reaching idle outputs. A Markov model can be established to evaluate the saturated throughput when N is small.[2] When N is large, the slot-by-slot number of HOL cells destined for a particular output becomes a Poisson process. As described in Section 5.5.2, the HOL blocking limits the maximum throughput to 0.586 when $N \to \infty$.

On–Off Model and Bursty Traffic. In the bursty traffic model, each input alternates between active and idle periods of geometrically distributed duration. Figure 5.28 considers an on–off source model in which an arrival process to an input port alternates between on (active) and off (idle) periods. A traffic source continues sending cells in every time slot during the on period, but stops sending cells in the idle period. The duration of the active period called a burst is determined by a random variable $X \in \{1, 2, \ldots\}$, which is assumed to be geometrically distributed with a mean of β cells (i.e., the mean burst length). Similarly, the duration of the off period is determined by a random variable $Y \in \{0, 1, 2, \ldots\}$, which is also assumed to be geometrically distributed with a mean of α cells. Define q as the probability of starting a new burst (on period) in each time slot, and p as the probability of terminating a burst. Cells arriving at an input port are assumed to be destined for the same output. Thus, the probability that the on period lasts for a duration of i time slots is

$$\Pr\{X = i\} = p(1 - p)^{i-1}, \quad i \geq 1$$

and we have the mean burst length

$$\beta = E[X] = \sum_{i=1}^{\infty} i \Pr\{X = i\} = 1/p.$$

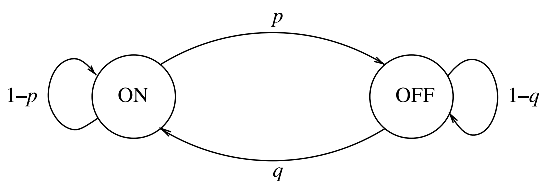


Figure 5.28 On–off source model.

[2]Consider the state as the different destination combinations among the HOL cells.

The probability that an off period lasts for j time slots is

$$\Pr\{Y = j\} = q(1-q)^{j-1}, \quad j \geq 0$$

and we have

$$\alpha = E[Y] = \sum_{j=0}^{\infty} j \Pr\{Y = j\} = (1-q)/q.$$

The offered load ρ is the portion of time that a time slot is active:

$$\rho = \frac{1/p}{1/p + \sum_{j=0}^{\infty} jq(1-q)^j} = \frac{q}{q+p-pq}.$$

5.5.2 Input-Buffered Switches

We consider first-in-first-out (FIFO) buffers in evaluating the performance of input queuing. We assume that only the cells at the head of the buffers can contend for their destined outputs. If there are more than one cell contending the same output, only one of them is allowed to pass through the switch and the others have to wait until the next time slot. When a head-of-line (HOL) cell loses contention, at the same moment it may also block some cells behind it to reach idle outputs. As a result, the maximum throughput of the switch is limited and cannot be 100 percent.

To determine the maximum throughput, we assume that all the input queues are saturated. That is, there are always cells waiting in each input buffer, and whenever a cell is transmitted through the switch, a new cell immediately replaces it at the head of the input queue. If there are k cells waiting at the heads of input queues addressed to the same output, one of them will be selected at random to pass through the switch. In other words, each of the head-of-line (HOL) cells has equal probability $(1/k)$ of being selected.

Consider all N cells at the heads of input buffers at time slot m. Depending on the destinations, they can be classified into N groups. Some groups may have more than one cell, and some may have none. For those which have at least a cell, one of the cells will be selected to pass through the switch, and the remaining cells have to stay until the next time slot. Denote B_m^i as the number of remaining cells destined for output i in the mth time slot, and B^i the random variable in the steady state [8, 9]. Also denote A_m^i as the number of cells moving to the heads of the input queues during the mth time slot and destined for output i, and A^i the corresponding steady-state random variable. Note that a cell can only move to the head of an input queue if the HOL cell in the previous time slot was removed from that queue for transmission to an output. Hence, the state transition of B_m^i can be represented by:

$$B_m^i = \max\left(0, B_{m-1}^i + A_m^i - 1\right). \tag{5.1}$$

We assume that each new cell arrival to the head of an input queue has the equal probability $1/N$ of being destined for any given output. As a result, A_m^i has the following binomial distribution:

$$\Pr\left[A_m^i = k\right] = \binom{F_{m-1}}{k} (1/N)^k (1-1/N)^{F_{m-1}-k},$$
$$k = 0, 1, \ldots, F_{m-1}, \tag{5.2}$$

where

$$F_{m-1} \triangleq N - \sum_{i=1}^{N} B_{m-1}^{i}. \tag{5.3}$$

F_{m-1} represents the total number of cells transmitted through the switch during the $(m-1)$st time slot, which in saturation is equal to the total number of input queues which have a new cell moving into the HOL position in the mth time slot. That is,

$$F_{m-1} = \sum_{i=1}^{N} A_{m}^{i}. \tag{5.4}$$

When $N \to \infty$, A_m^i has a Poisson distribution with rate $\rho_m^i = F_{m-1}/N$. In steady state, $A_m^i \to A^i$ also has a Poisson distribution. The rate is $\rho_0 = \overline{F}/N$, where $\overline{F}$ is the average of the number of cells passing through the switch, and ρ_0 is the utilization of output lines (i.e., the normalized switch throughput). The state transition of B^i is driven by the Markov process same as the $M/D/1$ queues in the steady state. Using the results for the mean steady-state queue size for an $M/D/1$ queue, for $N \to \infty$ we have

$$\overline{B^i} = \frac{\rho_0^2}{2(1-\rho_0)}. \tag{5.5}$$

In the steady state, (5.3) becomes

$$\overline{F} = N - \sum_{i=1}^{N} \overline{B^i}. \tag{5.6}$$

By symmetricity, $\overline{B^i}$ is equal for all i. In particular,

$$\overline{B^i} = \frac{1}{N}\sum_{i=1}^{N} \overline{B^i} = 1 - \frac{\overline{F}}{N} = 1 - \rho_0. \tag{5.7}$$

It follows from (5.5) and (5.7) that $\rho_0 = \left(2 - \sqrt{2}\right) = 0.586$.

When N is finite and small, the switch throughput can be calculated by modeling the system as a Markov chain. The numerical results are given in Table 5.1. Note that the throughput rapidly converges to 0.586 as N increases.

The results also imply that, if the input rate is greater than 0.586, the switch will become saturated, and the throughput will be 0.586. If the input rate is less than 0.586, the throughput will be 100 percent, and every cell will get through the switch after some delay. To characterize the delay, a discrete-time *Geom*$/G/1$ queuing model can be used to obtain an exact formula of the expected waiting time for $N \to \infty$.

The result is based on the following two assumptions: (1) The arrival process to each input queue is a Bernoulli process. That is, the probability that a cell arrives in each time slot is identical and independent of any other slot. We denote this probability as p, and call it the offered load. (2) Each cell is equally likely to be destined for any one output. The 'service time' for a cell at the head of line consists of the waiting time until it gets

TABLE 5.1 The Maximum Throughput Achievable Using Input Queuing with FIFO Buffers

N	Throughput
1	1.0000
2	0.7500
3	0.6825
4	0.6553
5	0.6399
6	0.6302
7	0.6234
8	0.6184
∞	0.586

selected, plus one time slot for its transmission through the switch. As $N \to \infty$, and in the steady state, the number of cells arriving to the heads of input queues, and addressed to a particular output (say j) has the Poisson distribution with rate p. Hence, the service time distribution for the discrete-time $Geom/G/1$ model is the delay distribution of another queuing system: a discrete-time $M/D/1$ queue with customers served in random order.

Using standard results for a discrete-time $Geom/G/1$ queue, we obtain the mean cell waiting time for input queuing with FIFO buffers

$$\overline{W} = \frac{p\overline{S\,(S-1)}}{2\left(1-p\overline{S}\right)} + \overline{S} - 1, \tag{5.8}$$

where S is the random variable of the service time obtained from the $M/D/1$ model. The mean cell waiting time for input queuing with FIFO buffers is computed and shown in Figure 5.29 for $N \to \infty$.

5.5.3 Output-Buffered Switches

With output queuing, cells are only buffered at outputs, at each of which a separate FIFO is maintained. Consider a particular (i.e., tagged) output queue. Define the random variable A as the number of cell arrivals destined for the tagged output in a given time slot. In [8, 9], based on the same assumptions as in Section 5.5.2 on the arrivals, we have

$$a_k \overset{\triangle}{=} \Pr[A = k] = \binom{N}{k} (p/N)^k \,(1 - p/N)^{N-k}\,, \qquad k = 0, 1, 2, \ldots. \tag{5.9}$$

When $N \to \infty$, (5.9) becomes

$$a_k \overset{\triangle}{=} \Pr[A = k] = \frac{p^k e^{-p}}{k!}$$
$$k = 0, 1, 2, \ldots, N = \infty \tag{5.10}$$

Denote Q_m as the number of cells in the tagged queue at the end of the mth time slot, A_m the number of cell arrivals during the mth time slot, and b the capacity of the output buffer,

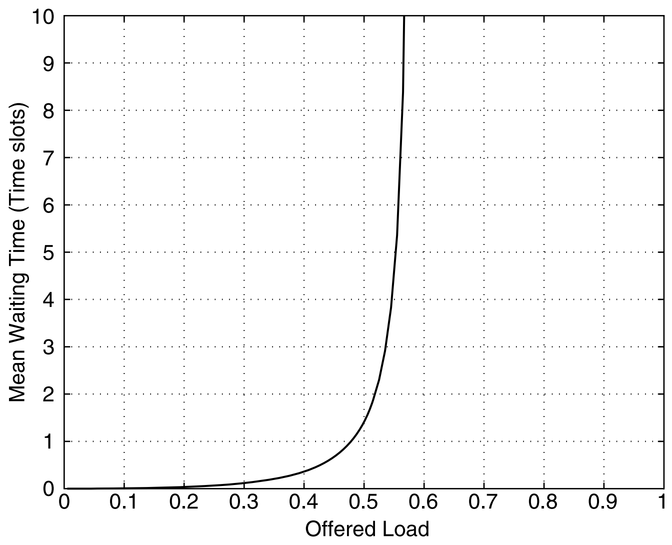


Figure 5.29 The mean waiting time for input queuing with FIFO buffers for the limiting case for $N = \infty$.

we have

$$Q_m = \min\{\max(0, Q_{m-1} + A_m - 1), b\}. \tag{5.11}$$

If $Q_{m-1} = 0$ and $A_m > 0$, there is no cell waiting at the beginning of the mth time slot, but we have A_m cells arriving. We assume that one of the arriving cells is immediately transmitted during the mth time slot; that is, a cell goes through the switch without any delay.

For finite N and finite b, this can be modeled as a finite-state, discrete-time Markov chain with state transition probabilities $P_{ij} \overset{\triangle}{=} \Pr[Q_m = j \mid Q_{m-1} = i]$ in the following:

$$P_{ij} = \begin{cases} a_0 + 1 & i = 0,\ j = 0 \\ a_0 & 1 \le i \le b,\ j = i - 1 \\ a_{j-i+1} & 1 \le j \le b - 1,\ 0 \le i \le j \\ \displaystyle\sum_{m=j-i+1}^{N} a_m & j = b,\ 0 \le i \le j \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

where a_k is given by (5.9) and (5.10) for a finite N and $N \to \infty$, respectively. The steady-state queue size can be obtained recursively from the following Markov chain balance equations:

$$\begin{cases} q_1 \overset{\triangle}{=} \Pr[Q = 1] = \dfrac{(1 - a_0 - a_1)}{a_0} \cdot q_0 \\ q_n \overset{\triangle}{=} \Pr[Q = n] = \dfrac{(1 - a_1)}{a_0} \cdot q_{n-1} - \displaystyle\sum_{k=2}^{n} \dfrac{a_k}{a_0} \cdot q_{n-k}, \quad 2 \le n \le b, \end{cases}$$

where

$$q_0 \overset{\triangle}{=} \Pr[Q = 0] = \frac{1}{1 + \sum_{n=1}^{b} q_n / q_0}.$$

No cell will be transmitted on the tagged output line during the mth time slot if, and only if, $Q_{m-1} = 0$ and $A_m = 0$. Therefore, the switch throughput ρ_0 is represented as:

$$\rho_0 = 1 - q_0 a_0.$$

A cell will be lost if, when emerging from the switch fabric, it finds the output buffer already containing b cells. The cell loss probability can be calculated as follows:

$$\Pr[\text{cell loss}] = 1 - \frac{\rho_0}{p},$$

where p is the offered load.

Figures 5.30*a* and 5.30*b* show the cell loss probability for output queuing as a function of the output buffer size b for various switch size N and offered loads $p = 0.8$ and 0.9. At the 80 percent offered load, a buffer size of $b = 28$ is good enough to keep the cell loss probability below 10^{-6} for arbitrarily large N. The $N \to \infty$ curve can be a close upper bound for finite $N > 32$. Figure 5.31 shows the cell loss performance when $N \to \infty$ against the output buffer size b for offered loads p varying from 0.70 to 0.95.

Output queuing achieves the optimal throughput-delay performance. Cells are delayed unless it is unavoidable when two or more cells arriving on different inputs are destined for the same output. With Little's result, the mean waiting time $\overline{W}$ can be obtained as follows:

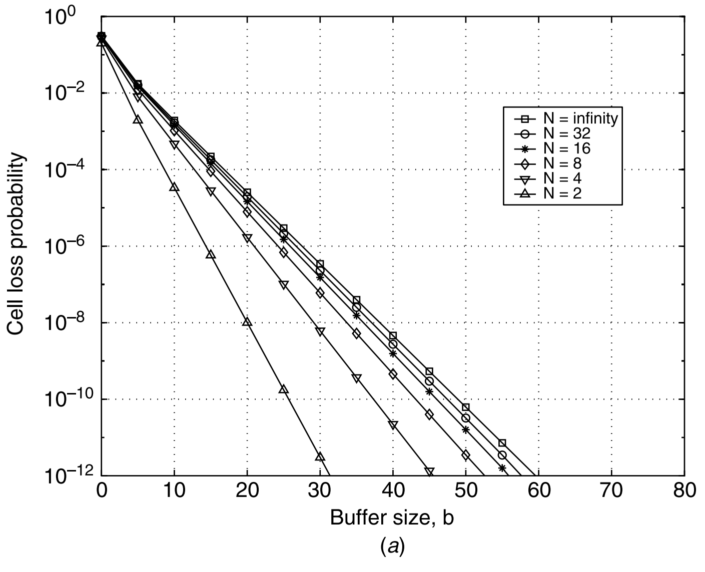
$$\overline{W} = \frac{\overline{Q}}{\rho_0} = \frac{\sum_{n=1}^{b} n q_n}{1 - q_0 a_0}.$$

Figure 5.32 shows the numerical results for the mean waiting time as a function of the offered load p for $N \to \infty$ and various values of the output buffer size b. When $N \to \infty$ and $b \to \infty$, the mean waiting time is obtained from the $M/D/1$ queue as follows:

$$\overline{W} = \frac{p}{2(1-p)}.$$

5.5.4 Completely Shared-Buffered Switches

With completely buffer sharing, all cells are stored in a common buffer shared by all inputs and outputs. One can expect that it will need less buffer to achieve a given cell loss probability, due to the statistical nature of cell arrivals. Output queuing can be maintained logically with linked lists, so that no cells will be blocked from reaching idle outputs, and



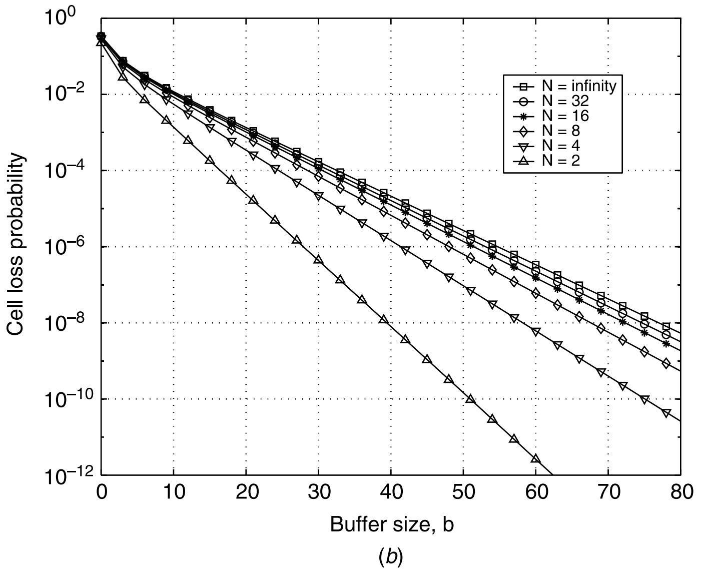


Figure 5.30 Cell loss probability for output queuing as a function of the buffer size b and the switch size N, for offered loads (a) $p = 0.8$ and (b) $p = 0.9$.

we still can achieve the optimal throughput-delay performance as with dedicated output queuing.

In the following, we will take a look on how it improves the cell loss performance. Denote Q_m^i as the number of cells destined for output i in the buffer at the end of the mth time slot. The total number of cells in the shared buffer at the end of the mth time slot is $\sum_{i=1}^{N} Q_m^i$. If the buffer size is infinite, then

$$Q_m^i = \max\left\{0, Q_{m-1}^i + A_m^i - 1\right\}, \tag{5.13}$$

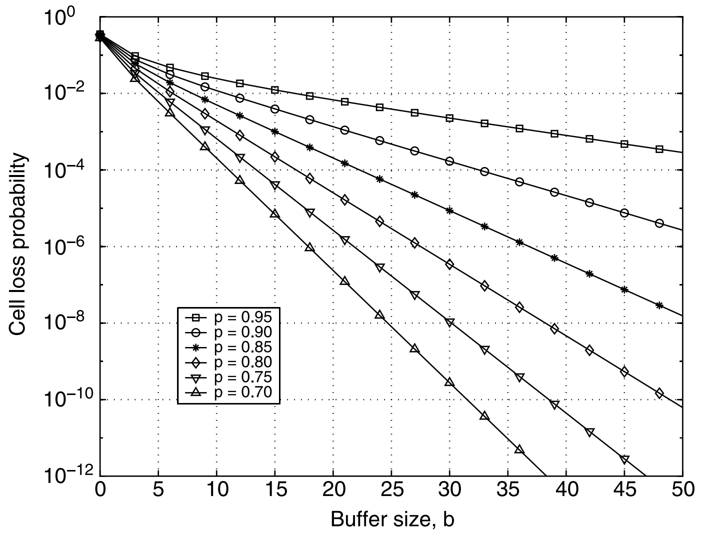


Figure 5.31 Cell loss probability for output queuing as a function of the buffer size b and offered loads varying from $p = 0.70$ to $p = 0.95$, for the limiting case of $N = \infty$.

where A_m^i is the number of cells addressed to output i that arrive during the mth time slot.

With a finite buffer size, cell arrivals may fill up the shared buffer and the resulting buffer overflow makes (5.13) only an approximation. However, we are only interested in the region of low cell loss probability (e.g., less than 10^{-6}), in which this approximation is still good.

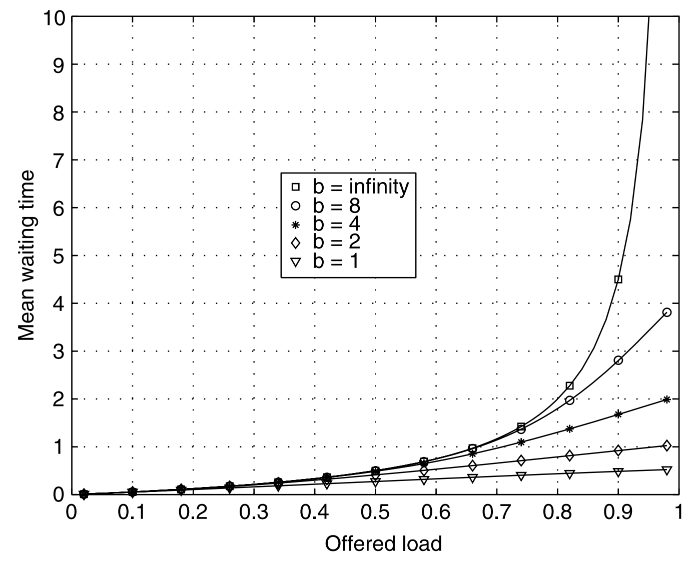


Figure 5.32 Mean waiting time for output queuing as a function of the offered load p, for $N = \infty$ and output FIFO sizes varying from $b = 1$ to $b = \infty$.

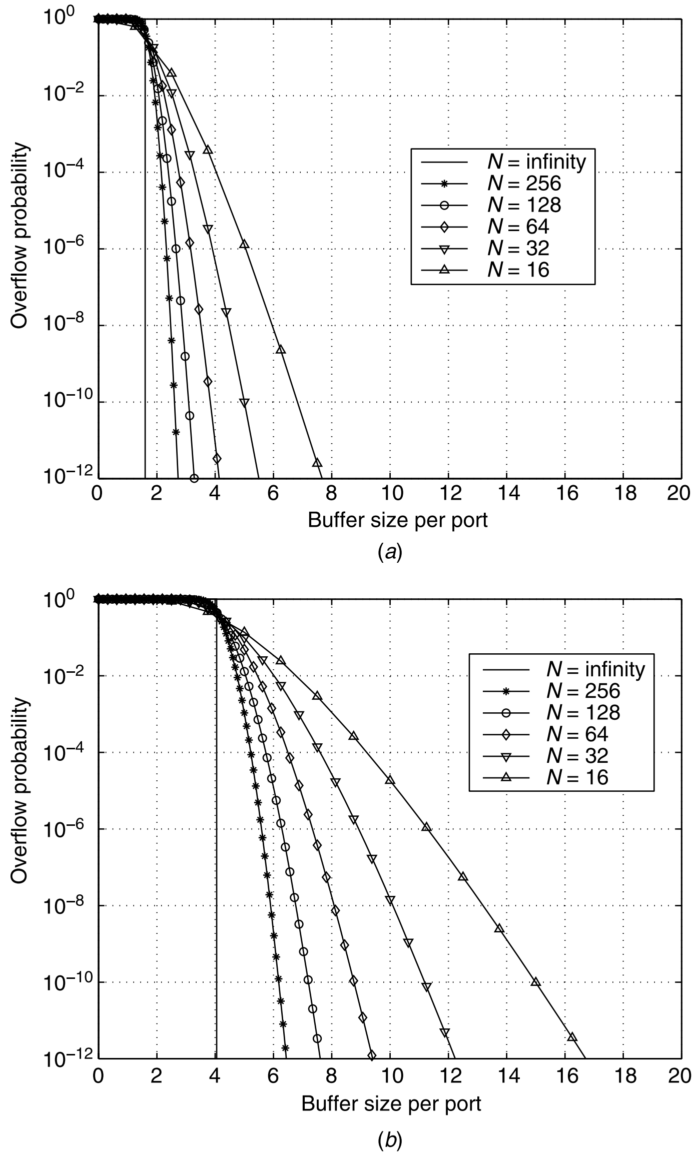


Figure 5.33 Cell loss probability for completely shared buffering as a function of the buffer size per output b and the switch size N, for offered loads (a) $p = 0.8$ and (b) $p = 0.9$.

When N is finite, A^i, which is the number of cell arrivals destined for output i in the steady state, is not independent of A^j $(j \neq i)$. This is because, at most N cells arrive to the switch, and a large number of cells arriving for one output implies a small number for the remaining outputs. As N goes to infinity, however, A^i becomes an independent Poisson random variable (with mean value p). Then, Q^i, which is the number of cells in the buffer that are destined for output i in the steady state also becomes independent of Q_j $(j \neq i)$. We will use the Poisson and independence assumption for finite N. These the approximations are good for $N \geq 16$.

Therefore we model the steady-state distribution of $\sum_{i=1}^{N} Q^i$, the number of cells in the buffer, as the N fold convolution of N $M/D/1$ queues. With the assumption of an infinite buffer size, we then approximate the cell loss probability by the overflow probability $\Pr\left[\sum_{i=1}^{N} Q^i \geq Nb\right]$. Figure 5.33*a* and 5.33*b* show the numerical results.

REFERENCES

[1] F. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and E. Neri, “Packet scheduling in input-queued cell-based switches,” in *Proc. IEEE INFOCOM’01*, Anchorage, Alaska, vol. 2, pp. 1085–1094 (Apr. 2001).

[2] F. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, “Packet-mode scheduling in input-queued cell-based switches,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 666–678 (Oct. 2002).

[3] Y. Ganjali, A. Keshavarzian, and D. Shah, “Input queued switches: cell switching vs. packet switching,” in *Proc. IEEE INFOCOM’03*, San Francisco, California, vol. 3, pp. 1651–1658 (Mar. 2003).

[4] F. A. Tobagi, T. K. Kwok, and F. M. Chiussi, “Architecture, performance and implementation of the tandem banyan fast packet switch,” *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1173–1193 (Oct. 1991).

[5] S. C. Liew and T. T. Lee, “$N \log N$ dual shuffle-exchange network with error-correcting routing,” in *Proc. IEEE ICC’92*, vol. 1, Chicago, Illinois, pp. 1173–1193 (June 1992).

[6] J. N. Giacopelli, J. J. Hickey, W. S. Marcus, W. D. Sincoskie, and M. Littlewood, “Sunshine: a high-performance self-routing broadband packet switch architecture,” *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1289–1298 (Oct. 1991).

[7] A. Huang and S. Knauer, “STARLITE: a wideband digital switch,” in *Proc. IEEE GLOBECOM’84*, Atlanta, Georgia, pp. 121–125 (Dec. 1984).

[8] M. J. Karol, M. Hluchyj, and S. Morgan, “Input vs. output queuing on a space-division packet switch,” *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347–1356 (Dec. 1987).

[9] M. J. Karol, M. Hluchyj, and S. Morgan, “Input vs. output queuing on a space-division packet switch,” *IEEE Transactions on Communications*, vol. 35, issue 12 (Dec. 1987).

[10] N. McKeown, “Scheduling algorithms for input-queued cell switches,” Ph.D. Thesis, UC Berkeley, May 1995.

[11] N. McKeown, “iSLIP: a scheduling algorithm for input-queued switches,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201 (Apr. 1999).

[12] H. J. Chao, “Saturn: a terabit packet switch using Dual Round-Robin,” *IEEE Communications Magazine*, vol. 38, issue 12, pp. 78–84 (Dec. 2000).

[13] D. N. Serpanos and P. I. Antoniadis, “FIRM: a class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues,” in *Proc. IEEE INFOCOM’00*, Tel Aviv, Israel, pp. 548–554 (Mar. 2000).

[14] A. L. Gupta and N. D. Georganas, “Analysis of a packet switch with input and output buffers and speed constraints,” in *Proc. IEEE INFOCOM’91*, Bal Harbour, Florida, vol. 2, pp. 694–700 (Apr. 1991).

[15] J. S.-C. Chen and T. E. Stern, “Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch,” *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, pp. 439–449 (Apr. 1991).

[16] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030–1039 (June 1999).

[17] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Automatica*, vol. 35, no. 12 (Dec. 1999).

[18] S. Iyer and N. McKeown, "Using constraint sets to achieve delay bounds in CIOQ switches," *IEEE Communications Letters*, vol. 7, no. 6, pp. 275–277 (June 2003).

[19] S. Iyer, A. Awadallah, and N. McKeown, "Analysis of a packet switch with memories running slower than the line-rate," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Israel, vol. 2, pp. 529–537 (Mar. 2000).

[20] S. Iyer and N. McKeown, "Making parallel packet switches practical," in *Proc. IEEE INFOCOM'01*, Anchorage, Alaska, vol. 3, pp. 1680–1687 (Apr. 2001).

[21] S. Iyer and N. McKeown, "Analysis of the parallel packet switch architecture," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 314–324 (Apr. 2003).

[22] H. J. Chao, Z. Jing, and S. Y. Liew, "Matching algorithms for three-stage bufferless Clos network switches," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 46–54 (Oct. 2003).

[23] A. Jajszczyk, "Nonblocking, repackable, and rearrangeable Clos networks: fifty years of the theory evolution," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 28–33 (Oct. 2003).