a constant rate of cell delivery. This is important because delay-sensitive applications, such as a voice communication, cannot tolerate long delays among their cells. *Cell-delay variations* can occur both at sources and within a network. In any case, a destination user must be able to adapt to cell-delay variations.

In order to maintain a constant bit rate (CBR), a certain algorithm is used in ATM networks. Let J_i be the cell i delay variation that a given application can tolerate. Let c be the constant rate at which cells are delivered to users. Thus, the interval between two similar points of two consecutive arriving cells is $\frac{1}{c}$. Let t_i be the time at which the ith cell is received. Then, the variable delay for the ith cell is given by

$$J_i = J_{i-1} - \left(t_i - (t_{i-1} + \frac{1}{c})\right). \tag{12.25}$$

The analysis bases J_1 as the first value and starts recursively, using i, where $i \geq 2$. For example, for the second cell, we have:

$$J_2 = J_1 - \left(t_2 - (t_1 + \frac{1}{c})\right). \tag{12.26}$$

The cell-delay variation can be reduced by increasing both the available network resources and the data rate at the user network interface. The cell delay at the network level can be lower, because cells are small and fixed size and have a fixed header format. For ATM networks, as well as other types of networks, resource allocation has to be controlled to avoid congestion. Variable cell delay can occur because of the processing overhead in the ATM layer and the physical layer. The encapsulation process at the ATM layer and the overhead bits added at the physical layer lead to variable cell delays.

The key factor that makes it extremely difficult to control wide-area ATM networks is the fact that the cell-transmission time in ATM networks is quite small; hence, feedback control may not be able to react quickly, making the use of adaptive reservation systems for congestion control ineffective. The key factors that influence traffic management in ATM networks are latency and cell-delay variation. The network propagation delay can sometimes be comparable to cell-transmission times. Hence, a source cannot use implicit feedback control, as the feedback control signals arrive too late for any appropriate cell-retransmission strategy.

Resource Management Using Virtual Paths

The main resource management in ATM networks involves the proper use of virtual paths. Several *virtual channel connections* (VCC) are grouped into a *virtual path connection* (VPC). ATM networks provide an aggregate capacity for all virtual channels

within a virtual path. The primary parameters for traffic management in ATM networks are *cell-loss ratio*, *cell-transfer delay*, and *cell-delay variation*. If a virtual channel passes through multiple virtual paths, the performance on that virtual channel depends on the performance of all the virtual paths that incorporate the virtual channel.

Different VCCs within a VPC may have different QoS requirements. The aggregate capacity and performance characteristics for a virtual path should be set to meet the requirements of the most demanding VCCs. There are two different approaches to resource allocation for virtual paths. In the first approach, an ATM network can set the capacity of the virtual path to meet the peak data rates of all VCCs within that VPC. However, this may lead to an underutilization of network resources. The second approach follows the idea of *statistical multiplexing*. An ATM network can set the capacity of a VPC to be greater than or equal to the average capacity of all its VCCs. With this approach, the total capacity of the virtual path becomes smaller than the aggregate peak data rate, leading to more efficient utilization of network resources. The second approach works best when a number of VCCs are grouped into a VPC on the basis of similar quality of service.

Connection Admission Control

When requesting a new VCC or VPC in ATM networks, a user must associate the connection with a particular QoS. Generally, a user can choose from a range of QoS classes that the network supports. An ATM network accepts the connection only if it can meet the QoS requirements without compromising on the QoS of existing connections. Once the network accepts a connection, the user and the network enter into a traffic contract. The network continues to provide the QoS as long as the user does not violate the traffic contract. The traffic contract is characterized by four parameters: *peak cell rate* (PCR), *cell-delay variation* (CDV), *sustainable cell rate* (SCR), and *burst tolerance*.

The peak cell rate of a connection is the maximum rate at which a source generates cells within a connection. The sustainable cell rate represents the upper bound on the average cell rate associated with a given connection. Burst tolerance represents the maximum variability of the cell-arrival rate from a SCR. For a constant bit rate (CBR) source, only the first two parameters are relevant. The cell-delay variation may cause an increase in the cell rate from its peak value, because cell delays may cause cells to clump up, thereby accounting for an increase in the cell rate from its peak value.

For variable bit rate (VBR) sources, all four parameters are relevant. A future flow pattern can be predicted by knowing the SCR and burst tolerance. A network characterizes each connection based on these four parameters. The admission-control decision is based primarily on users' QoS requirements. In an ATM connection, a user

is also allowed to assign a cell-loss priority (CLP). The CLP bit in the header indicates either normal priority (CLP = 0 or 1) or high priority (CLP = 0).

Usage Parameter Control

Once a network's admission-control function accepts a connection, the *usage parameter control* scheme is used to determine whether the connection confirms the traffic contract. The usage parameter control mechanism is normally accompanied by a QoS traffic-shaping scheme. The usage parameter control involves the control of the peak cell rate and the associated cell-delay variation. The traffic flows are monitored to check whether the flow violates the contract by exceeding the peak cell rate, subject to the maximum specified cell-delay variation.

The CLP header bit defined in Section 2.4.2 is used for tagging or discarding noncompliant cells. Those cells that were tagged and cells with CLP = 1 are classified as low-priority cells. If the network becomes congested, these cells are dropped first, in order to provide better performance to high-priority cells. Usage parameter control can also provide a mechanism of traffic policing, whereby cells that violate the traffic contract are either tagged or discarded.

12.4.5 Cell Scheduling and QoS

ATM networks provide a *guaranteed frame rate* (GFR) service that requires the network to handle frames in addition to ATM cells. With GFR, all cells of a frame have the same CLP bit setting. Frames with CLP = 1 setting are low-priority frames. This service provides a minimum guarantee to a connection by setting up GFR virtual channels. The minimum capacity is ensured only for frames with CLP = 0. The implementation involves *tagging* and *policing*.

The tagging process is used to identify frames that violate the GFR traffic contract. The ATM network sets the CLP bit to 1 on all cells in a noncompliant frame. Tagged cells are given a lower priority in the buffer-management and scheduling stages. The buffer management is done when congestion occurs. In such a case, all tagged cells are discarded to give preference to untagged cells. The buffering scheme is set up such that an untagged cell can replace a tagged cell in the buffer when resources are limited.

As a QoS component, a scheduler gives preference to untagged cells over tagged cells. Scheduling among router queues ensures the minimum rate requirements for each virtual channel. The traffic on each connection is monitored all the time. For a frame with confirmed contract terms, all cells in the frame must be confirmed. Frames that violate the traffic contract are either tagged to a lower priority or discarded. Therefore,

this process filters frames that may cause congestion by overloading the network. Frames are then checked to determine whether they qualify for guaranteed delivery.

12.5 Summary

Methods of providing QoS are divided into two broad categories: *integrated services* and *differentiated services*. Integrated services provide QoS to individual applications and flow records. QoS protocols in this category include traffic shaping and packet scheduling. Traffic shaping regulates the spacing between incoming packets. Two traffic-shaping algorithms are *leaky bucket* and *token bucket*. In a leaky-bucket traffic shaper, traffic is regulated at a constant rate, much like the flow of water from a leaky bucket. In the token-bucket algorithm, enough tokens are assigned to each incoming packet. If the bucket becomes empty, packets stay in the buffer until the required number of tokens is generated.

Packet scheduling involves managing packets in queues. Several scheduling techniques *FIFO*, *priority queueing*, *fair queueing*, and *weighted fair queueing* (WFQ). WFQ is an improvement over fair queueing, in which each flow i is assigned a weight w_i. Another version of WFQ is *deficit round-robin*, in which each flow i is allocated b_i bits in each round of service.

The differentiated services (DiffServ) approach is based on providing QoS support to a broad class of applications. The *traffic conditioner*, one of the main features of a DiffServ node, protects the DiffServ domain. The traffic conditioner includes four major components: *meter*, *marker*, *shaper*, and *dropper*.

Resource allocation in networks can be classified by various approaches: fixed versus adaptive, router based versus host based, window based versus rate based. Finally a few methods of ATM resource control were presented.

The next chapter discusses the internal architecture of switch fabrics. This topic is the basis of discussion in several later chapters as well.

12.6 Exercises

1. In a leaky-bucket traffic-shaping scheme, $w = 4$ grants are initially allocated to a session. The count is restored back to 4 every $4/g$ seconds. Assume a Poisson arrival of packets to the system.
   (a) Develop the first five balance equations for the corresponding Markovian process.
   (b) Sketch a Markov chain representing the number of grants allocated to packets, and indicate as many transition values as you have computed.

2. A small router is equipped with the leaky-bucket traffic-shaping scheme; the capacity of the grant buffer is set on window size $w = 2$. Packet arrival during $1/g$ uniformly distributed, with four possibilities of $k = 0, 1, 2$, and 3 packets. Consider only non-queueing states with the assumption of $P_0 = 0.008$.
   (a) Find the probability that k packets arrive during $1/g$, denoted by $P_X(k)$.
   (b) Find the probabilities that 0, or 1 grants are left in the grant buffer.
   (c) Find all the transition probabilities for three states 0, 1, and 2.
   (d) Sketch a Markov chain representing the number of grants allocated to packets for the first three states.

3. A router attached to mail server is responsible only for receiving and smoothing e-mail. The router is equipped with the leaky-bucket traffic-shaping scheme, whereby the capacity of the grant buffer is set on window size $w = 4$. Packets arrive at $\lambda = 20$ packets per second, and grantarrive at $g = 30$ packets per second. Packet arrival during $1/g$ is distributed according to Poisson. Consider only non-queueing states with the assumption of $P_0 = 0.007$.
   (a) Find the probability that k packets arrive during $1/g$, denoted by $P_X(k)$.
   (b) Find the probability that 0, or 1 grants are left in the grant buffer.
   (c) Find all the transition probabilities for four states 0, 1, 2, and 3.
   (d) Sketch a Markov chain representing the number of grants allocated to packets for the first four states.

4. Consider a token-bucket traffic shaper. Let the bucket size be b bits and the token arrival rate be v b/s; let the maximum output data rate be z b/s.
   (a) Derive an equation for T_b, the time consumed for a flow to be transmitted at the maximum rate.
   (b) Find T_b when the bucket size is $b = 0.5$ Mb, $v = 10$ Mb/s, and $z = 100$ Mb/s.

5. Does nonpreemptive priority queueing change the packet transmission orders if bit-by-bit round-robin service is used?

6. Derive an expression for a priority scheduler to present the mean residual service time, r_i, used in Equation (12.10). (*Hint:* Use μ_i.)

7. Assume that all components of a three-flow ($n = 3$) priority scheduler are identical: $\lambda_i = \lambda = 0.2$/ms, $1/\mu_i = 1/\mu = 1$ms, and $r_i = r = 0.5$ms. Find the total system delay for a packet passing each of the three queues 1, 2, and 3 and the server denoted by $E[T_1]$, $E[T_2]$, and $E[T_3]$, respectively. Do the following.
   (a) Use a nonpreemptive priority scheduler for the task.
   (b) Use a preemptive priority scheduler for the task.
   (c) Comment on results obtained in parts (a) and (b).

8. We want to compare the impact of an increased number of inputs on total delay in priority schedulers. Assume that all components of a three-flow ($n = 3$) and a four flow ($n = 4$) priority scheduler are identical: $\lambda_i = \lambda = 0.2$/ms, $1/\mu_i = 1/\mu =$ 1ms, and $r_i = r = 0.5$ms. Find the total system delay for a packet passing queue 3 denoted by $E[T_3]$. Do the following
   (a) Use a nonpreemptive priority scheduler for the task.
   (b) Use a preemptive priority scheduler for the task.
   (c) Justify the results obtained in parts (a) and (b).

9. Suppose that four flows are processed in a router that accepts only equal-size packets. Packets in these flows arrive at the following virtual clock times:
   Flow 2: 4, 5, 6, 7, 9
   Flow 2: 1, 6, 9, 12, 14
   Flow 3: 1, 4, 8, 10, 12
   Flow 4: 2, 4, 5, 6, 12
   (a) For each packet, give the virtual clock count at which it is transmitted, using fair queueing. If the arrival times of two packets are the same, the smaller flow number is selected.
   (b) Now consider weighted fair queueing, whereby flows 1, 2, 3, and 4 are given 10 percent, 20 percent, 30 percent, and 40 percent of the output capacity, respectively. For each packet, give the virtual clock count at which it is transmittted.

10. Consider the router interfaces in Figures 3.11 and 3.16. Explain where and how fair queueing can be implemented.

11. We define the timing parameters s_i, f_i, and a_i for a weighted fair-queueing scheduler similar to what we did for the fair-queueing scheduler. Derive a relationship among these parameters, taking into consideration the weight of each flow (packet), w_i.

12. In a nonpreemptive priority queueing scheme, a low-priority flow is guaranteed to receive 10 percent of the total bit rate s of the transmission link.
   (a) How much better does this low-priority flow perform?
   (b) What would be the impact on the performance of the high-priority flows?

13. Each output port processor unit of a router has four inputs designated to four different flows. The unit receives the packets in the following order during a period

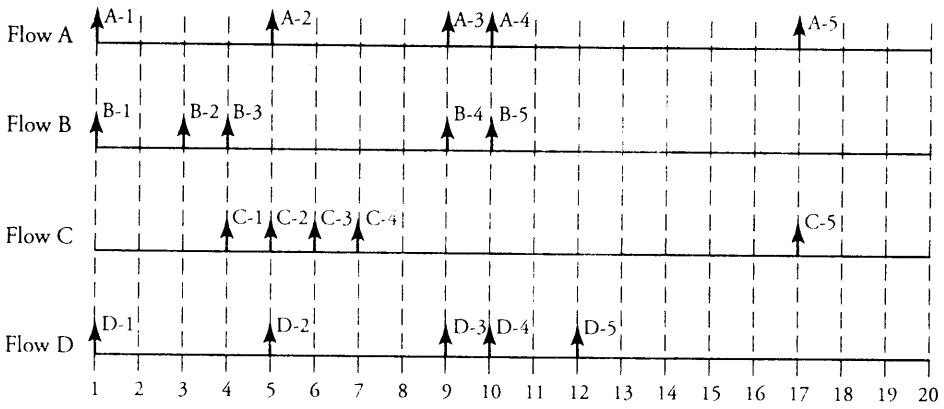

Figure 12.16 Exercise 14 comparison of priority queueing, fair queueing, and weighted fair queueing on processing of four flows

in which the output port is busy but all queues are empty. Give the order in which the packets are transmitted.

Flow: 1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 4

Packet: 1, 2, 3, 4, 5, 6, 7, ,8, 9, 10, 11, 12

Packet size: 110, 110, 110, 100, 100, 100, 100, 200, 200, 240, 240, 240

(a) Assume a fair queueing scheduler.

(b) Assume a weighted fair-queueing scheduler with flow $i \in \{1, 2, 3, 4\}$ having weights $w_i \in \{10\%, 20\%, 30\%, 40\%\}$ of the output capacity, respectively.

14. In Figure 12.16, flows A, B, C, and D are processed at one of the outputs of a router. For each packet, the time it arrives and its label are indicated in the figure. Specify the order of packets transmitted at this output, using the following three schedulers. All schedulers scan the flows, starting from flow A.

(a) *Priority queueing*. Priorities decrease from line A to line D.

(b) *Fair queueing*. If the arrival times of two packets are the same, the smaller flow number is selected.

(c) *Weighted fair queueing*, where flows A, B, C, and D are given 10 percent, 20 percent, 30 percent, and 40 percent, respectively, of the output capacity.

15. Figure 12.17 shows four flows (A, B, C, and D) to be processed at one of the outputs of a router. For each packet, the time it arrives and its label are indicated

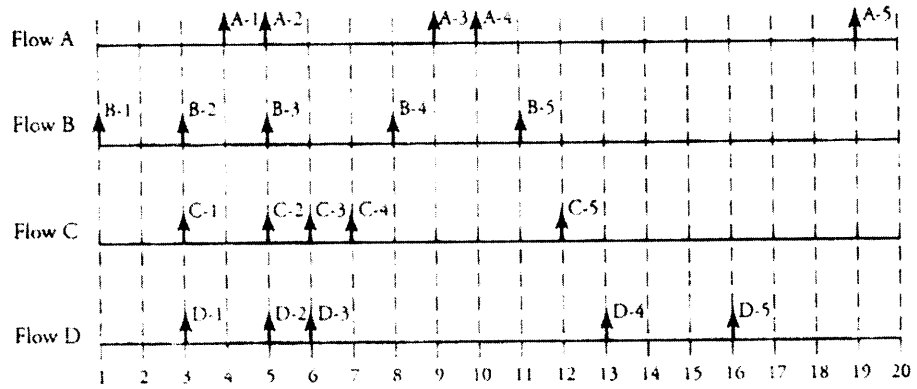

Figure 12.17 Exercise 15 comparison of priority queueing, fair queueing, and weighted fair queueing on processing of four flows.

in the figure. Specify the order of packets transmitted at this output, using the following three schedulers. All schedulers scan the flows, starting from flow A.

(a) *Priority queueing.* Priorities decrease from line A to line D.

(b) *Fair queueing.* If the arrival times of two packets are the same, the smaller flow number is selected.

(c) *Weighted fair queueing*, where flows A, B, C, and D are given 20 per cent, 10 per cent, 40 per cent, and 30 per cent, respectively, of the output capacity.

CHAPTER 13

Networks in Switch Fabrics

A *switch fabric* is the core switching engine in switching devices as routers. Chapter 3 presented an overview of switching devices. This chapter analyzes this core segment of such devices and introduces several topologies for switching networks.

This chapter explores the following topics:

- *Characteristics and features of switch fabrics*
- *Crossbar switch fabrics*
- *Blocking switch fabrics*
- *Nonblocking switch fabrics*
- *Concentration and expansion switches*
- *Shared-memory switch fabrics*
- *Techniques for improving performance*
- *Case study multipath buffered crossbar*

We begin by classifying characteristics of switching networks and presenting features and definitions of switch fabrics. *Crossbar switches* are the building blocks of switching fabrics, so all aspects of this switch are covered. The case study at the end of this chapter combines several buffered crosspoints to form a buffered crossbar. Blocking switches are constructed with crossbar switches, including *Omega networks*, *Banyan networks*, *Delta networks* and *Beneš networks*. Nonblocking switches constructed with

crossbars include *Clos networks*. *Concentration-based* and *expansion-based* switching networks—two special-purpose switch fabrics—are investigated as well.

A quite different approach for switching in *time domain* uses shared memory, without using any switch elements. Switch fabric techniques that offer better performance include the use of buffering, combined networks, and parallel-plane switching networks.

13.1 Characteristics and Features of Switch Fabrics

The switching function takes place in the switching fabric. The switching structure depends on the need of network operation, available technology, and the required capacity. A switch fabric is an interconnected network of smaller switching units. Several factors can be used to characterize switching systems: buffering, complexity, capability of multipoint connections, speed, performance, cost, reliability, fault tolerance, and scalability. Here, we focus on the topology of the interconnection network as the heart of modern switching systems. The key factors for classifying switching networks are

- Single path versus multipath
- Fixed interstages versus variable interstages
- Deflection routing versus packet discard
- Blocking versus nonblocking

Switching networks are either *single-path* or *multipath* networks. A single-path network has exactly one route between each input port and output port. However, this property can be a source of traffic congestion and traffic blocking. In a multipath network, any connection can be established through more than one path. Each of these two groups of networks can be further classified as either *single-stage* or *multistage* networks. In a single-stage network, a connection is established through one stage of switching; in a multistage network, a packet must pass through several switching stages. In a multistage network, the number of stages often grows with increasing network size.

13.1.1 Blocking and Nonblocking Networks

A switching network is said to be *blocking* if an input port cannot be connected to an unused output port. Otherwise, the switching network is *nonblocking*. A nonblocking switching network can be either of two types. A network is *wide-sense nonblocking* if any input port can be connected to any unused output port without requiring a path to be rerouted. A network is *rearrangeably nonblocking* if, for connecting an input

port to an unused output port, rearranging other paths may be required. A wide-sense nonblocking network is *strictly nonblocking* if there is an idle path from any idle input to idle output for all states of the network. Rearrangeably nonblocking architectures have a more complex control algorithm to make connections, since fewer switches are used. The complexity of the routing algorithm is a trade-off with cost.

13.1.2 Features of Switch Fabrics

Switching networks have several features and options.

- Switching networks may be either *buffered* or *unbuffered*. Buffers may be used to reduce traffic congestion.
- To regulate the flow of packets and prevent buffer overflow, *flow control* can be provided between stages in a multistage network.
- *Discard versus deflection.* At each stage of a switching network, a conflict may arise if several packets request the same output. In networks without flow control, arriving packets that cannot be buffered can be either *discarded* or *deflected*. Or, these two methods can be combined in a switch.
- Modern switching networks are expected to have the capability of *multicasting*—copying to any subset of the outputs—along with *broadcasting*.

13.1.3 Complexity of Switching Networks

A useful measure for approximating the cost of a network is to consider the number of crosspoints and links. In practice, the number of crosspoints has greater impact on the cost of the switching network. Thus, *complexity* refers to the total number of crosspoints used in a switching network. It is important to note that the integrated circuit pin constraints also influence implementation of large-scale networks, especially those with parallel data paths. Practically, a switching network or portions of a network can be placed entirely on a single integrated circuit package. In such cases, the area occupied by the circuits becomes an important complexity measure.

13.1.4 Definitions and Symbols

The topic of switch networking entails a number of definitions and symbols. Consider a network, N, with n inputs and m outputs. This network is referred to as an (n, m) network. An (n, n) network is referred to as an n network. A connection for a given network is identified by (x, y), where x is an input port and y is an output port. Inputs and outputs to a network are numbered in decimal or binary, starting from 0. Switch

elements are labeled by (i, j), where i is the stage number of the switch, and j is the location of the switch in that stage. We use $(i, j)_h$ to identify input or output port h on switch (i, j).

A network is *uniform* if all switch elements have the same dimensions and is *square* if it has the same number of inputs and outputs. The *mirror* of a network N is denoted N^m and is obtained by exchanging inputs and outputs and reversing the directions of all links. Consider network N_1 with n_1 outputs and network N_2 with n_2 inputs. The *concatenation* of two networks N_1 and N_2 is represented by $N_1 + N_2$ and is realized by using output i of N_1 through input i of N_2. If i is a positive integer and N is an (n, m) network, $i.N$ realizes the network obtained by taking i copies of N, without interconnecting them. The product term $N_1 \times N_2$ is the *series* connection of N_1 and N_2 and is defined by

$$N_1 \times N_2 = (n_2.N_1) + \phi_{n_1,n_2} + (n_1.N_2), \tag{13.1}$$

where ϕ_{n_1,n_2} is the *permutation* among n_1 and n_2 points. Similarly, for network N_1 with n_1 outputs, network N_2 with n_2 inputs and n_3 outputs, and network N_3 with n_1 inputs, the product term $N_1N_2N_3$ is defined as

$$N_1N_2N_3 = (n_2.N_1) + \phi_{n_1,n_2} + (n_1.N_2) + \phi_{n_3,n_1} + (n_3.N_3) \tag{13.2}$$

and is called the *parallel* connection of N_1 and N_2.

13.2 Crossbar Switch Fabrics

Crossbar switches are the building blocks of switching fabrics. A crossbar switch with n inputs and n outputs is shown by $X_{n,n}$. Figure 13.1 shows a $X_{4,4}$, or a 4×4 crossbar switching fabric. In a crossbar, every input can be uniquely connected to every output through a *crosspoint*. A crosspoint is the smallest unit of the switching function and can be built using a variety of electronic or digital elements, such as photodiodes, transistors, and AND gates.

Crossbar switches are considered strictly (wide-sense) nonblocking, as a dedicated crosspoint exists for every connection. This is clearly an attractive feature of a crossbar. The blocking, however, may occur when multiple packets are sent to the same output simultaneously. If a particular output port of the crossbar is idle, the desired connection can always be established by selecting the particular crosspoint dedicated to the particular input/output pair.

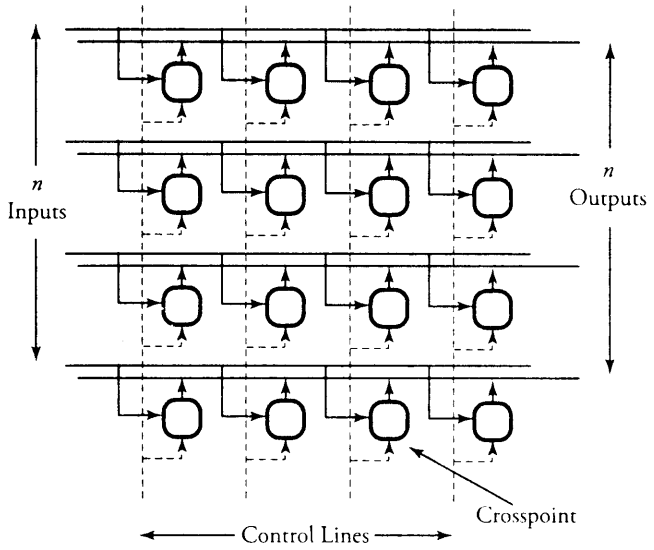

Figure 13.1 A crossbar switching fabric with $n = 4$ inputs/outputs

Crossbar switches are conceptually simple. Their only potential problem is output port contention. The complexity of a switch lies in the complexity of its crosspoints. For an $n \times n$ crossbar, the complexity rises in a quadratic fashion and is equal to n^2 crosspoints. By some clever design work, the complexity can be reduced at the price of bringing certain blocking into the switching.

13.3 Blocking Switch Fabrics

Although, a crossbar can be deployed solely as a switch fabric, it may also be used to construct other multistage switch fabrics, with the trade-off of cost versus higher blocking. This section looks at types of blocking switches: *Omega networks*, *Banyan networks*, *Delta networks*, and *Beneš networks*.

13.3.1 Omega Network

The *Omega network*, $\Omega_{n,d}$, is shown in Figure 13.2 (a). Using $\Omega_{d,d} = X_{d,d}$, we define the Omega network by

$$\Omega_{n,d} = \phi_{n/d,d} + (n/d.X_{d,d}) + \phi_{d,n/d} + (d.\Omega_{n/d,d}). \tag{13.3}$$

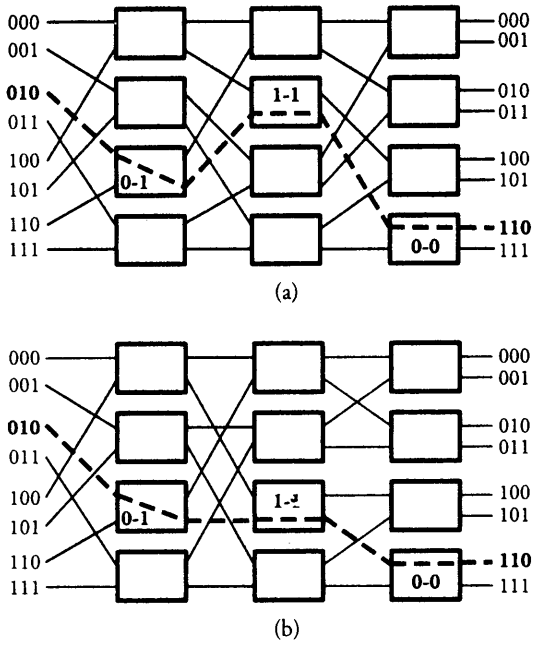

Figure 13.2 (a) An Omega network with an example of its routing; (b) a Banyan network with an example of its routing

In an *Omega network*, only one route exists between each input i_x and output o_x. This fact explains the potential for blocking. The path uniqueness in this network can be shown by the induction on the number of stages, s. The number of stages is exactly $s = \log_d n$.

The Omega network has some useful unique properties. One is that the interconnection segment between each two consecutive stages is identical to that of all other segments. Each of these identical interconnection segments is configured with a *shuffling* property. The shuffling property is easy to understand: An interconnection link starting at the output number of a given crossbar stage ends at the input of the next crossbar stage with the same number but one step rotated to left. For example, consider the second interconnection segment between stages 1 and 2. In this segment, suppose that we are interested in finding the address of the crossbar input this link is connecting to. According to the rotation property, this address is obtained by one left rotation of 010, which results in 100.

The Omega network is also self-routing. The routing in this network is very simple and is achieved by comparing bit by bit between a source address and a destination

address. At each stage, we look at the corresponding bits, starting at the most-significant bit (MSB) of the source and the destination addresses. If bits are the same, a message passes through; otherwise, it is crossed over. For example, consider a case with the source address 010 and the destination address 110, as shown in Figure 13.2. In the first stage, the message is crossed over, as the MSBs of the two addresses are different (0-1) . In the second stage, the next corresponding bits are the same (1-1), so the message is sent straight along the crossbar. The case for the third stage is also the same as for the second stage but with corresponding bits of 0-0.

Using the method of blocking probability estimation presented in Section 7.6.4, we can easily obtain an estimation of blocking for the Omega network. Finding all the paths that connect any given input to a given output and assigning p to each and every link of paths as the blocking probability of that link, estimate the blocking probability for an Omega network, $\Omega_{n,d}$, to be $1 - (1 - p)^{s-1}$, where $s = \log_d n$.

13.3.2 Banyan Network

The *Banyan network*, $Y_{n,d}$, as shown in Figure 13.2 (b), is similar to the Omega network. Using $Y_{d,d} = X_{d,d}$, we define the Banyan network by

$$Y_{n,d} = \phi_{n/d,\,d} + (n/d.X_{d,d}) + \phi_{d,\,n/d} + (d.Y_{n/d,d}). \tag{13.4}$$

The Banyan network has some useful unique properties. Similar to the Omega network, the Banyan network also has the property of self-routing. Identical to Omega networks, the blocking probability for a Banyan networks, $B_{n,d}$, is estimated to be $1 - (1 - p)^{s-1}$, where $s = \log_d n$.

13.3.3 Delta Networks

The *Delta network* consists of a network of crossbar switch elements with shared crosspoints, thus raising the possibility of blocking. The Delta network $D_{n,d}$, using $d \times d$ crossbars, $X_{d,d}$, and with a input/output ports, is defined by

$$D_{n,d} = X_{d,d} \times D_{n/d,d}. \tag{13.5}$$

In a Delta network, only one route between each input i_x and output o_x exists. This fact explains the fundamental reason for blocking. The path uniqueness in this network can be shown by the induction on the number of stages, s. The number of stages is exactly $s = \log_d n$. As with Omega networks, the blocking probability for a Delta network, $D_{n,d}$, is estimated to be $1 - (1 - p)^{s-1}$, where $s = \log_d n$.

If $s = 1$, the network becomes $D_{d,d}$, or simply a crossbar. For $s > 1$, the output o_x belongs to one of d subnetworks denoted by $D_{n/d,d}$. Let s_1 be the first-stage switch,

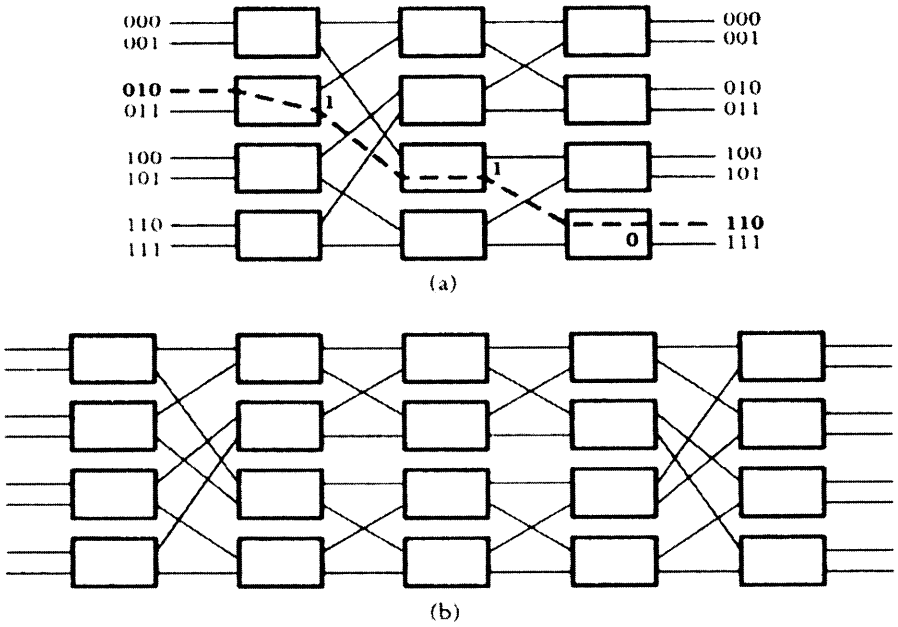

Figure 13.3 (a) A $D_{8,2}$ Delta network with the realization of its routing; (b) extension of the Delta network to a $B_{8,2}$ Beneš

in which i_x is an input. Since there is only one link l connecting s_1 to the subnetwork containing o_x, the only route connecting i_x to o_x is the route consisting of i_x, together with the route connecting l to o_x, which is unique. As seen in Figure 13.3, the Delta network is self-routing. A path from any input i_x to output o_y is performed by selecting link determined by successive digits of $o'_y s$ label. This process is also reversible, so we can route from output o_y back to i_x by following the links determined by successive digits of $i'_x s$ label.

13.3.4 Beneš Networks

The *Beneš network* consists of a combination of two networks: a Delta network and its mirrored overlapped in one stage, as shown in Figure 13.3. This network is capable of realizing all possible input/output permutations and is defined by $B_{d,\,d} = X_{d,\,d}$ and expanded by

$$B_{n,d} = X_{d,d} B_{n/d,\,d} X_{d,d}. \tag{13.6}$$

The recursive structure of the network leads us to decomposition of the routing problem into a set of smaller problems corresponding to each of the stages in the recursion. The top-level problem consists of selecting, for each connection, one of the d subnetworks $B_{n/d,d}$ to route through. The routing problems for the d subnetworks can be solved independently. The following algorithm can be used to route a packet to a specific output port:

Define:

d Crossbar switch element dimension
k $\log_d n$
r Number of stages, $r = 2 \log_d n - 1$
j Stage number, $j \in \{1, 2, \ldots, r\}$
A Output address of packet
a_j Address in stage j
v_j Bit vector specifying outputs that receive copies

Begin Beneš Network Routing Algorithm

For $1 \leq j \leq k - 1 \Longrightarrow$
a_j = random number $\in [0, d - 1]$

For $k \leq j \leq r \Longrightarrow$
$a_j = \left\lfloor \frac{A}{d^{r-j}} \right\rfloor \pmod d$
$v_j = 2^{a_j}$ ■

With this algorithm, packets in the routing network are guided to the requested outputs. As a result of its structure using Delta networks, we can derive the complexity of the Beneš network by

$$X_c = \frac{n}{d}(2log_d n - 1)d^2 = nd(2log_d n - 1) \tag{13.7}$$

knowing that $\frac{n}{d}$ is the number of crossbar switches per stage and that d^2 is the complexity of $d \times d$ crossbars.

13.4 Nonblocking Switch Fabrics: Clos Networks

Figure 13.4 shows a *Clos network* with $n = 8$ inputs, $d = 2$, and $k = 5$. If the number of middle-stage crossbar switch elements k is greater than or equal to $2d - 1$, $C^3_{n,d,k}$ is strictly nonblocking. Let d and k be integers with $2 \leq d \leq k$. The three-stage Clos network $C^3_{n,d,k}$ is defined by

$$C^3_{n,d,k} = X_{d,k} X_{n/d,n/d} X_{k,d}. \tag{13.8}$$

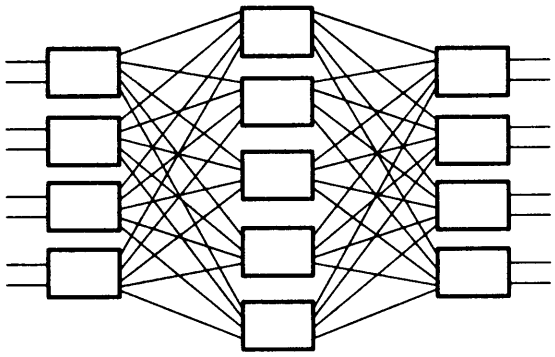

Figure 13.4 A Clos network with $n = 8$ inputs, $d = 2$, and $k = 5$

The proof of this claim can be derived by first observing that a connection through the three-stage switch requires a middle-stage switch element having an idle link from the first stage and an idle link to the third stage, as shown in Figure 13.5. In $C^3_{n,d,k}$, we search for a route realizing a connection request from input x to output y. Thus, the number of outputs of the stage 1 switch elements containing input x that are busy is at most $d - 1$. This means that there are at most $d - 1$ middle-stage switch elements that are not accessible form x. Similarly, there are at most $d - 1$ middle-stage switch elements that are not accessible from y.

Clearly, there are at most $2d - 2$ middle-stage switches that are either not accessible from x or not accessible from y. Hence, the worst-case situation for blocking is that these two sets of middle-stage switch elements are unavailable for connection request (x, y). However, if one additional free middle-stage switch element exists as shaded in the figure, that element can be used to set up the connection (x, y). Hence if $k = (d - 1) + (d - 1) + 1 = 2d - 1$, the switch is strictly nonblocking. More generally, at least one middle-stage switch that is accessible from both sides or a state that blocks connection request (x, y) can be found if $k \geq 2d - 1$. The complexity of the three-stage Clos network is

$$\begin{aligned} X_c &= dk\left(\frac{n}{d}\right) + k\left(\frac{n}{d}\right)^2 + dk\left(\frac{n}{d}\right) \\ &= 2kn + k\left(\frac{n}{d}\right)^2. \end{aligned} \tag{13.9}$$

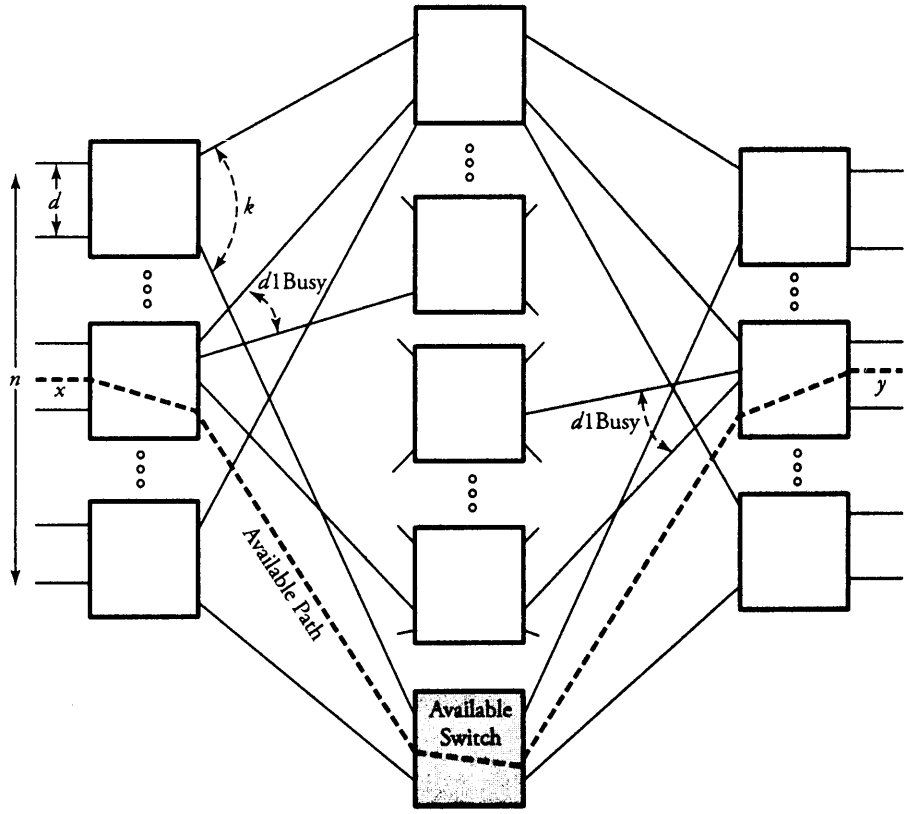

Figure 13.5 Analysis of blocking in the three-stage Clos network

In order to find the complexity of the Clos network under nonblocking conditions, we can substitute $k = 2d - 1$ in Equation 13.9. Thus, the complexity of a nonblocking network, $X_c^{n.b.}$, becomes

$$X_c^{n.b.} = (2d - 1)\left[2n + \left(\frac{n}{d}\right)^2\right]. \tag{13.10}$$

It is always useful to optimize the complexity of switching networks, especially for the nonblocking Clos networks, in which finding the best d would be beneficial. To achieve this goal, we can optimize the network by

$$\frac{\partial X_c^{n.b.}}{\partial d} = \frac{\partial}{\partial d}\left((2d - 1)\left[2n + \left(\frac{n}{d}\right)^2\right]\right) = 0. \tag{13.11}$$

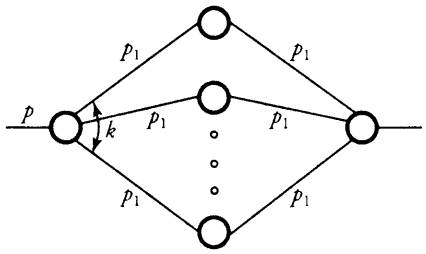

Figure 13.6 A blocking model for Clos switching network

This procedure releases $d = \sqrt{n/2}$, which is the absolute minimized value of d. The optimized crosspoint count under this optimization, therefore, is

$$X_{c,opt}^{n.b.} = 4n(\sqrt{2n} - 1). \tag{13.12}$$

However, the number of crosspoints for large three-stage switches is still quite prohibitive. Large switching systems typically use more than three stages to provide greater reductions in crosspoints. Three-stage Clos networks can be enhanced by substituting another three-stage Clos network for the middle-stage crossbars, resulting in a five-stage Clos network. We can continue in this fashion to construct networks with many stages.

13.4.1 Estimation of Blocking Probabilities

To estimate the internal blocking probability of a Clos network, consider $d \times k$ switch elements in the first stage. Figure 13.6 shows a probability graph of a three-stage network. All possible internal paths for an input/output connection pair are shown. For the middle-stage crossbars, the blocking-probability estimation for n parallel links, each with probability p, is generally $B = p^n$; for a series of n links, the blocking probability is generally $B = 1 - (1 - p)^n$. To apply this rule, let p_1 be the probability that an internal link is busy, as shown in the figure, and thus $1 - p_1$ is the probability that a link is idle. Then, B, the internal blocking probability of the switching network, or the probability that all paths are busy, is

$$B = [1 - (1 - p_1)^2]^k. \tag{13.13}$$

We want the traffic load to balance at both sides of each node. Specifically, at the first node, we want the external and internal traffic to be equal, such that $kp_1 = dp$ or

$p_1 = p(d/k)$. Using this and Equation (13.13) leads to B in terms of p:

$$B = \left[1 - (1 - p(d/k))^2\right]^k. \tag{13.14}$$

Example. For a Clos network with $k = 8$, $d = 8$, and $p = 0.8$, find the internal blocking probability.

Solution. Obviously, $k/d = 1$, and therefore the internal blocking probability $B = (1 - (1 - 0.8)^2)^8 = 0.72$.

Example. For the Clos network presented in the previous example, if we add up to 100 percent more crossbar switches into the middle stage of the network, how much reduction in internal blocking probability do we obtain?

Solution. Since $k = 16$, $d = 8$, and $p = 0.8$, then $k/d = 2$, and thus $B = ((1 - (1 - 0.8 \times 1/2)^2)^{16} = 0.0008$. The impact is remarkable, as the internal blocking probability is reduced 900 times.

13.4.2 Five-Stage Clos Networks

Figure 13.7 shows a five-stage Clos network obtained by replacing every middle-stage switch element in a three-stage Clos network. This structure provides less complexity, as the complexity of each middle three-stage network has been reduced as was explained for a three-stage network. Note that this statement can be true only if a five-stage switch is strictly nonblocking, where $k_1 \geq 2d_1 - 1$ and also $k_2 \geq 2d_2 - 1$. This type of design is especially useful for large-scale switching systems, in which a substantial reduction in the overall complexity of a network is targeted. The blocking probability of a five-stage network is modeled in Figure 13.8 and is determined as follows:

$$B = \{1 - (1 - p_1)^2[1 - (1 - (1 - p_2)^2)^{k_2}]\}^{k_1}. \tag{13.15}$$

13.5 Concentration and Expansion Switches

This section introduces switching networks that either expand or concentrate incoming traffic. In a concentration-based switching network, the number of output ports is less than the number of input ports. In an expansion-based switching network, the number of output ports is more than the number of input ports.

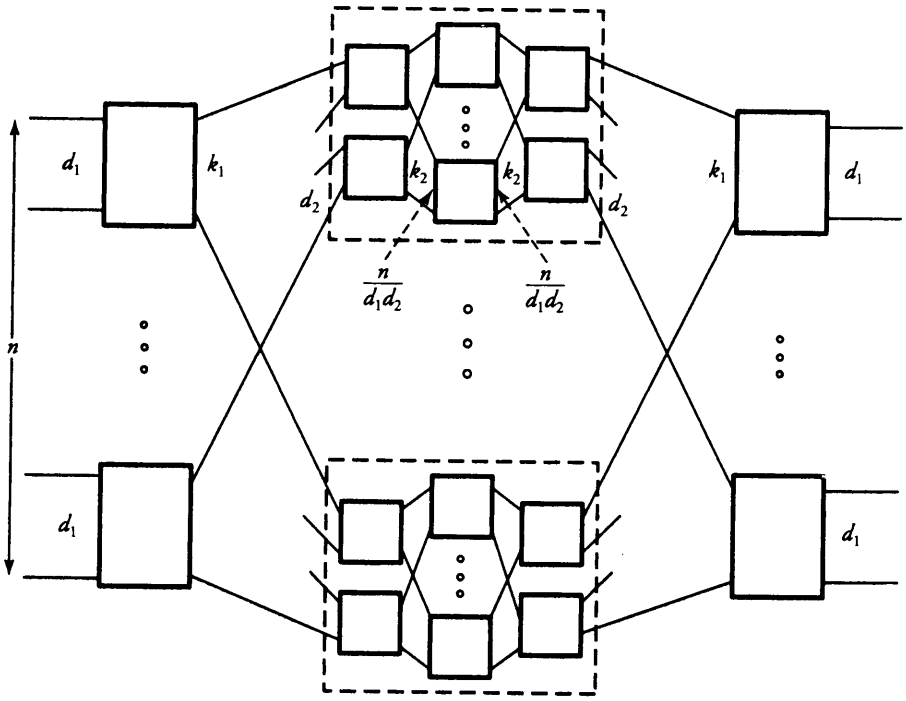

Figure 13.7 Constructing a five-stage Clos network by using three-stage networks to reduce blocking probability.

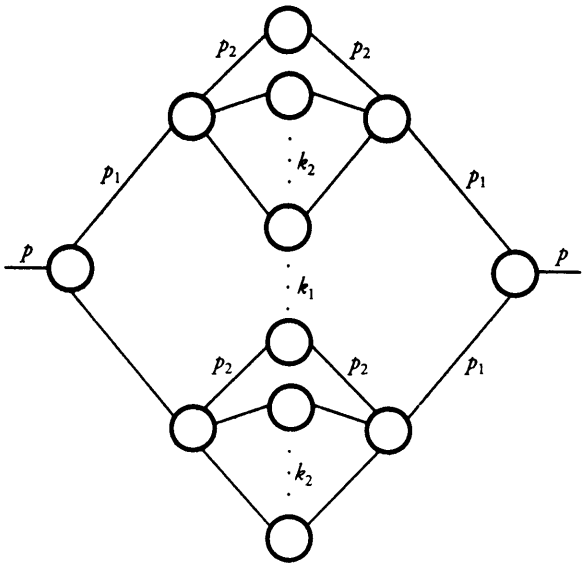

Figure 13.8 A blocking model for the five-stage Clos network

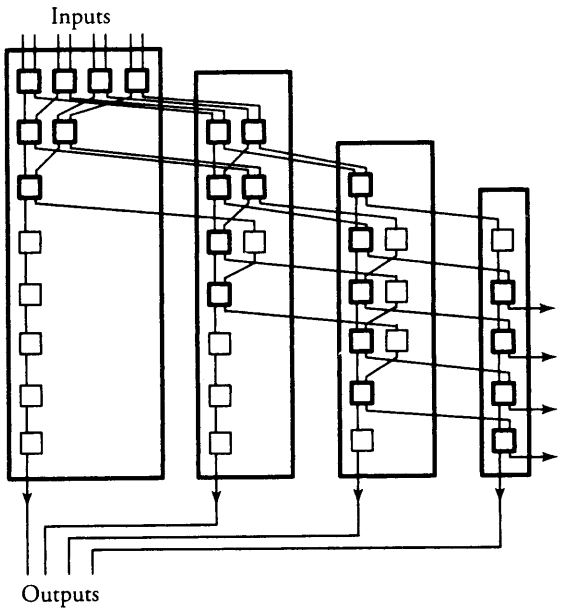

Figure 13.9 The knockout switching network

13.5.1 Knockout Switching Network

One of the simplest *concentration switching networks* is called a *knockout switch*, shown in Figure 13.9. The knockout switch is a blocking network and is constructed with interconnected crossbars with less complexity than a crossbar switch of the same size. The knockout switch is a good substitution for a crossbar switch when the likelihood is small that many inputs will need to send packets to the same output simultaneously.

The idea of this switch is based on the concept of knocking a possible k packets out of n active inputs if the number of switch outputs is m, where $m < n$. To implement this mechanism, consider the example illustrated in Figure 13.9, where $n = 8$ and $m = 4$. Assume that all the eight inputs have packets to transmit to outputs; obviously, the system would need to discard a minimum of four packets. The switching network consists of a number of either 2×2 switch elements and delay elements.

The switch elements act as concentrators, ensuring fairness to all entering packets. The eight incoming packets from eight switch inputs compete in the first four switch elements. A packet that wins the competition, based on a random selection in a switch element, stays in the same column of switches and continues to reach its desired output. Otherwise, a losing packet is knocked out to the next column of switch elements for

the next round of competition. Thus, the first-round losers in this process go straight to the second column and play off against one another and at later stages, face the second- and the third-round losing packets from the first column.

Note that the system needs to ensure that no input port is singled out for bad treatment every time the output is overloaded. This fairness task is implemented by playing packets against one another in a form of a knockout column-by-column scenario to select m winners. In the second column, two packets, including those losing packets, are competing in a similar fashion to reach their desired outputs, and so on. Finally, all the losing packets are dropped, and eventually, we select up to $m = 4$ packets, dropping all the rest. Note that this process requires timing synchronization if all the incoming packets are to experience the same switching delay. To ensure packet time synchronization, the delay elements present units of delay to packets, as shown in the figure by additional delay units.

13.5.2 Expansion Network

An *expansion network* with n_1 inputs and n_2 outputs, where $n_1 < n_2$, can scale up the number of receivers from n_1 to n_2. A three-stage *expansion network* is shown in Figure 13.10. This network is constructed with three types of crossbars of sizes $d_1 \times m$, $\frac{n_1}{d_1} \times \frac{n_2}{d_2}$, and $m \times d_2$, respectively, and is defined by

$$M(n_1, d_1, n_2, d_2, m) = X_{d_1,m} X_{n_1/d_1, n_2/d_2} X_{m,d_2}. \tag{13.16}$$

An expansion network is a generalization of the three-stage Clos network and is useful in applications in which an incoming signal is to be sent to multiple outputs. Such networks are also called *distribution networks*. Considering the previous network dimensions, the complexity of a three-stage expansion network, X_e, is derived by

$$X_e = d_1 m \frac{n}{d_1} + \frac{n_1 n_2}{d_1 d_2} m + d_2 m \frac{n_2}{d_2}. \tag{13.17}$$

An expansion network is nonblocking if $m \geq (d_1 - 1)n_2/d_2 + d_2$. We can prove this claim by assuming that if a given connection goes to two or more outputs of the same third-stage switch, the connection branches in the third-stage switch supply all the outputs. Suppose that we want to add a new end point to an existing connection from an input x to an idle output y. If an output is in the same third-stage switch as y, we can add the necessary branch at that point. Note that $d_2 - 1$ middle-stage switches are inaccessible from y, and at most $(d_1 - 1)n_2/d_2$ are inaccessible from x. Since m is greater than the sum of these two values, at least one middle-stage switch

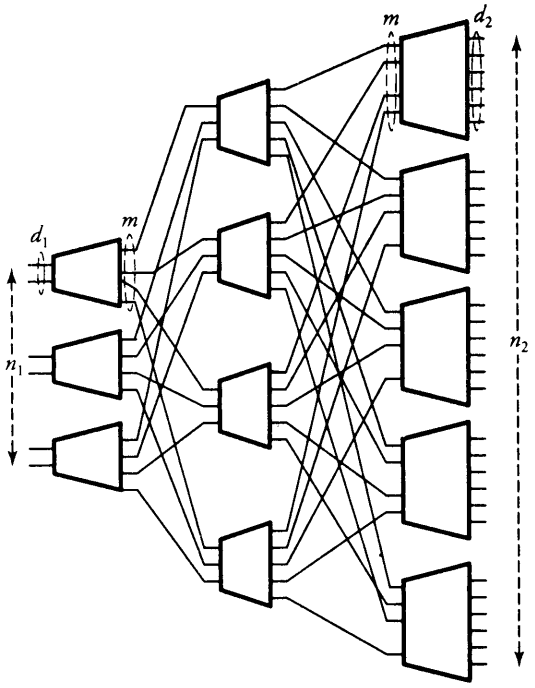

Figure 13.10 A three-stage expansion switching network

must be accessible to both x and y. A similar argument applies if we wish to create a new connection from an idle input x to an idle input y.

Example. Suppose that $n_1 = n_2 = 256$, $d_1 = 16$, and $d_2 = 64$. From the preceding theorem, if $m \geq 1024 + 64 = 1{,}088$, the network becomes nonblocking.

13.6 Shared-Memory Switch Fabrics

A totally different approach for switching can be achieved in time domain and without using any switch elements: the *shared-memory switch fabric*. As shown in Figure 13.11, the distinctive feature of this switch fabric is the use of a high-speed $n : 1$, time-division multiplexer (TDM) with a bit rate n times as large as the rate on each individual input/output line. Assume packet a at input i needs to be switched on output j and that packet b at input j is supposed to be switched on output i. First, the multiplexer creates an n-channel-long frame from the n inputs. Each frame created at the output

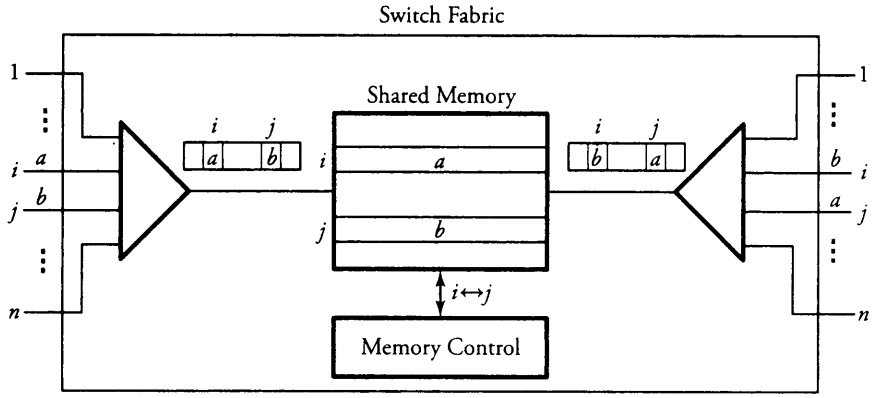

Figure 13.11 Shared-memory switch fabric

of the multiplexer is stored in *shared memory* in the same order as the frame's channels. Thus, a is written at location i, and b is written at location j of memory.

To implement the switching action between the two channels i and j, the *memory control* exchanges the reading order of the two memory locations i and j. This way, the new frame made out of memory contains packet a placed in channel j and packet b placed in channel i. This frame is scanned by a demultiplexer for output ports. If the memory partition for an output is empty, the corresponding minislot remains unfilled. It is a matter of implementation whether frame size in shared memory should be rigid or flexible. Clearly, variable-size frames require more sophisticated hardware to manage, but the packet-loss rate improves, since memory does not suffer from overflow until all memory slots are occupied. See Chapter 3 for more on multiplexers handling variable-size frames.

13.7 Techniques for Improving Performance

Several practical methods can be used to improve the performance of switching systems. The enhancement of a switch performance is typically related to speed, throughput, and therefore the possibility of blocking. Some possible methods for enhancing performance in switching systems are as follows:

- *Buffering*. The use of buffers in switching networks reduces traffic congestion and thus increases system throughput. With the use of buffers in a switch, packets are

not discarded when they request the same switch output but instead are kept in buffers for a later contention resolution.

- *Combined networks*. Combined networks consists of several cascaded switch fabrics, providing extra stages and producing extra paths, although a combined network can also be designed for other purposes, such as multicasting.
- *Randomizing traffic*. This technique is used to distribute traffic evenly across the switching network to prevent local congestion.
- *Recirculation of traffic*. A packet that is not successfully delivered to a given port can be recirculated. Such packets contend for delivery—possibly with higher priorities—in the next cycle.
- *Increase speed-up factor*. This *speed advantage* in a switching system refers to the ratio of the internal link speed to the external link speed.
- *Parallel-plane switching networks*. This technique forms parallel planes of switch fabrics to process slices of a packet in parallel.

The use of buffering, combined networks, and parallel-plane switching networks results in increased system complexity and cost but also better performance. Let's take a look at one such technique.

13.7.1 Parallel-Plane Switching Networks

One possible way to improve the performance and speed of switching networks, especially to reduce the blocking probability, is to arrange parallel planes, or slices, of the switching network, as shown in Figure 13.12. The *Cantor network*, $K_{n,d,k}$, is a widely used example of parallel-plane networks and is constructed by m planes of Beneš switching networks, $B_{n,d}$. The Cantor network is defined by

$$K_{n,d,m} = X_{1,m} B_{n,d} X_{m,1}. \tag{13.18}$$

The Cantor network is strictly nonblocking whenever

$$m \geq \frac{2}{d}(1 + (d - 1)\log_d(n/d)). \tag{13.19}$$

For $d = 2$, this reduces to $m \geq \log_2 n$, which gives us a strictly nonblocking network with approximately $4n(\log_2 n)^2$ crosspoints.

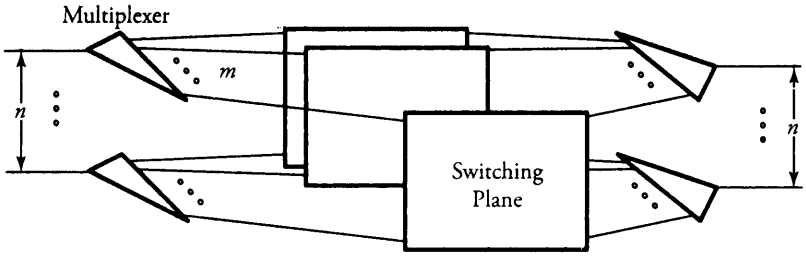

Figure 13.12 A Cantor network with m parallel switching planes for better reliability

Example. For $n = 2^{12}$, compare crossbar, Clos network, and Cantor network complexities.

Solution. If $n = 2^{12}$, the complexity for a crossbar is $n^2 = 2^{24} \approx 17 \times 10^6$. For a Clos network, the complexity is $4\sqrt{2}n^{3/2} = 1.5 \times 10^6$. For a Cantor network, the complexity is $4n(\log n)^2 = 2^{24} = 2.4 \times 10^6$.

In parallel-plane switching networks, the chip pin constraint limits the number of signals that can enter or leave a physical component. The integrated circuit pin constraints also influence implementation so that such systems are best designed with each data path passing through physically separate components. Such *bit-sliced organization* causes complexity to grow in proportion to the data path width.

13.8 Case Study: Multipath Buffered Crossbar

For our case study on a switching fabric, we choose an expandable switch fabric constructed with identical building-block modules called the *multipath buffered crossbar* (MBC). Conventional $n \times n$ crossbars are vulnerable to faults. Unlike a conventional crossbar, MBC is a crossbar with n rows and k columns. Each row contains an input bus and an output bus. A packet being transferred from input i to output j is sent by input port processor i on input bus i to one of the k columns. The packet then passes along this column to the output bus in row j. The crosspoints that make up the MBC switch fabric include mechanisms that allow the inputs to contend for access to the various columns. The crosspoint buffers in one row act as a distributed output buffer for the corresponding output port. As shown in Figure 13.13, every crosspoint has two different *switches* and a buffer.

Consider a case of point-to-point connection from input i_1 to output a_1. The packet is first randomly sent to a crosspoint at one of the k *shared data buses*, say, column

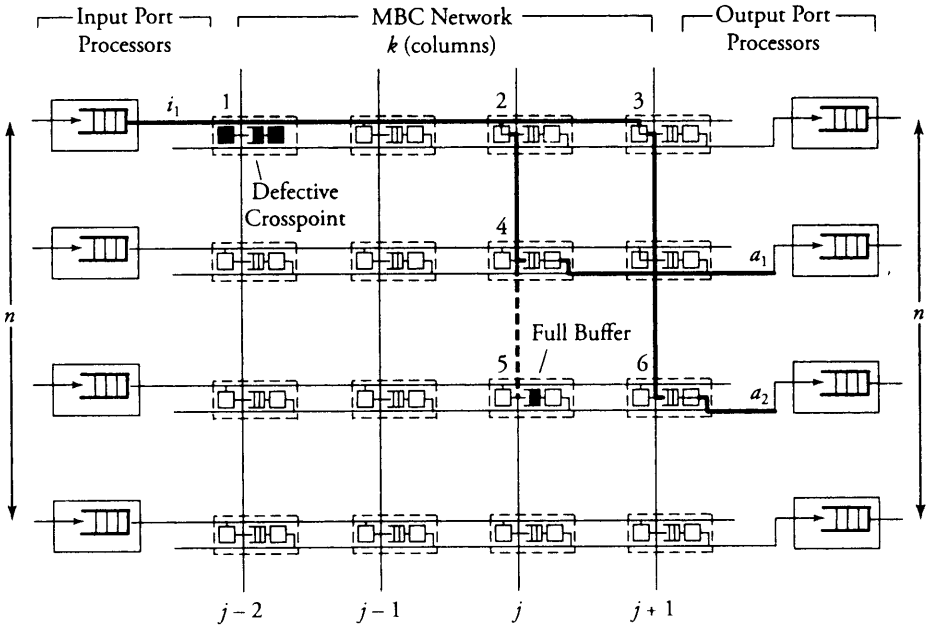

Figure 13.13 Routing and multicasting in the multipath buffered crossbar ($n = k = 4$)

$(j - 2)$. In case the crosspoint on this bus is not available for any reason, another crosspoint selected at random is examined, and so on. The reason for randomizing the initial location is to distribute the traffic as evenly as possible. When a functioning crosspoint (crosspoint 3) at column j is selected, the packet can be sent to a second crosspoint (crosspoint 4) at this column unless the selected crosspoint is found to be faulty or its buffer is full. The packet is buffered in crosspoint 4 and can be sent out after a contention-resolution process in its row.

The availability of each crosspoint at a given node (i, j) is dynamically reported to the input ports by a flag $g_{i,j}$. If more than one input port requests a given column, based on a contention-resolution process, only one of them can use the column. Packets in the other input ports receive higher priorities to examine other columns. Obviously, the more the packet receives priority increments, the sooner it is granted one free column, which it selects randomly. Once it gets accepted by one of the β buffers of a second crosspoint, the packet contends to get access to its corresponding output. The losers of the contention resolution get higher priorities for the next contention cycle until they are successfully transmitted. In Figure 13.13, the connections $i_1 \rightarrow a_1$ and $i_1 \rightarrow a_2$ are made possible through nodes {2, 4} and nodes {3, 6}, respectively.

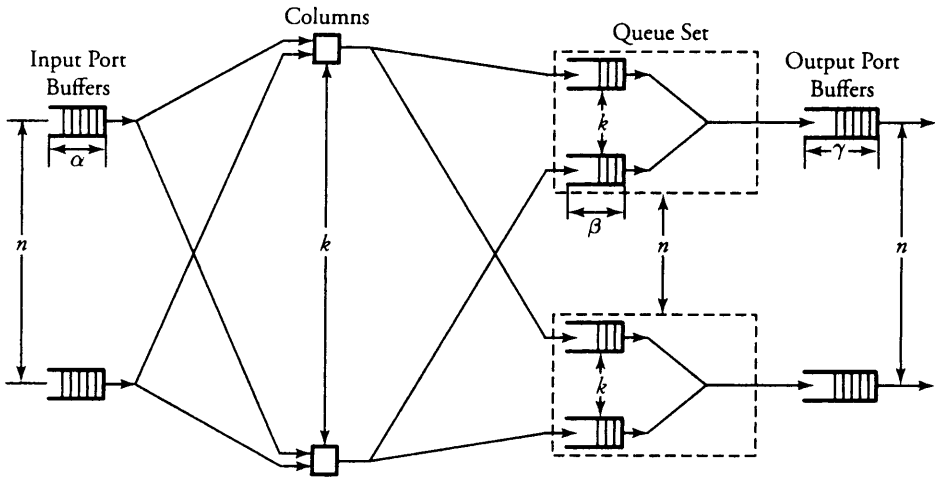

Figure 13.14 Queueing model for multipath buffered crossbar

13.8.1 Queueing Model

The queueing model for this system is shown in Figure 13.14. Packets enter the network from n *input port buffers* and pass through any of the k crosspoints belonging to the same row as the input. These crosspoints do not contribute in any queueing operation and act only as ordinary connectors. Each packet exiting these nodes can confront n crosspoint buffers of length β that are situated on one of the k-specified columns. A packet selects a buffer on the basis of the desired address. In this system architecture, each crossbar row contains k buffers, constructing a related group of buffers called a *queue set*, as seen in the figure. The winning packet of each set is then permitted to enter the corresponding *output port buffer*.

Note that the state of a buffer within each queue set is dependent on the state of all other buffers in that queue set. There are no *grant flows* between the output port buffers and the crosspoint buffers, and the admission of a packet by a queue set is made possible by granting the *flag* acknowledgments to the input port buffers. It should also be noted that the analysis that follows assumes that all crosspoints are functional. Let $z_k^{\beta}(s)$ be the number of ways to distribute s packets among k distinct buffers within a queue set, under the restriction that each buffer may contain at most β packets. Note that here, we are not concerned about the number of combinations of different packets in the queue. Therefore, $z_k^{\beta}(s)$ can be recursively computed by

$$z_k^\beta(s) = \begin{cases} 1 & \text{if } (k = 1 \wedge s \leq \beta) \vee (s = 0) \\ 0 & \text{if } s > k\beta \\ \sum_{0 \leq i \leq \min\{\beta, s\}} z_{k-1}^\beta(s - i) & \text{if } 0 < s \leq k\beta. \end{cases} \tag{13.20}$$

Let $X_k^\beta(r, s)$ be the probability that a given crosspoint buffer has r packets when the entire queue set contains s packets. Then

$$X_k^\beta(r, s) = \frac{z_{k-1}^\beta(s - r)}{z_k^\beta(s)}. \tag{13.21}$$

To further extend this analysis, an expression for the flow control between a crosspoint queue set and the input buffers is required. A second crosspoint generates a final flag acknowledgment, Ψ, after having completed a three-step process. The probability that a packet is available to enter the network, a, is the same probability that the input port buffer is not empty and is given by

$$a = 1 - \pi_i(0). \tag{13.22}$$

Let ψ be the probability that a packet contending for access to a column wins the contention. Note that the probability that any given input attempts to contend for any given column is a/k. Hence,

$$\psi = \sum_{0 \leq c \leq n-1} \frac{1}{c+1} \binom{n-1}{c} (a/k)^c (1 - a/k)^{(n-1)-c}, \tag{13.23}$$

where $\binom{n-1}{c}$ is a *binomial* coefficient indicating the number of different combinations of $n - 1$ possible existing contenders on one of the k columns, taken c contenders at a time. Now, the probability that a packet forwarded by input port buffer i is admitted, Ψ, is given by:

$$\Psi = \psi \sum_{0 \leq s \leq b} \pi_x(s) \left[1 - X_k^\beta(\beta, s)\right]. \tag{13.24}$$

We use Ψ later to derive the probability that an input port buffer contains exactly a certain number of packets.

13.8.2 Markov Chain Model

The expression for Ψ facilitates the determination of the state of the input port buffers. The input port queue can be modeled as a $(2\alpha + 1)$-state *Markov chain* (see Figure 13.15). The transition rates are determined by the offered load ρ and

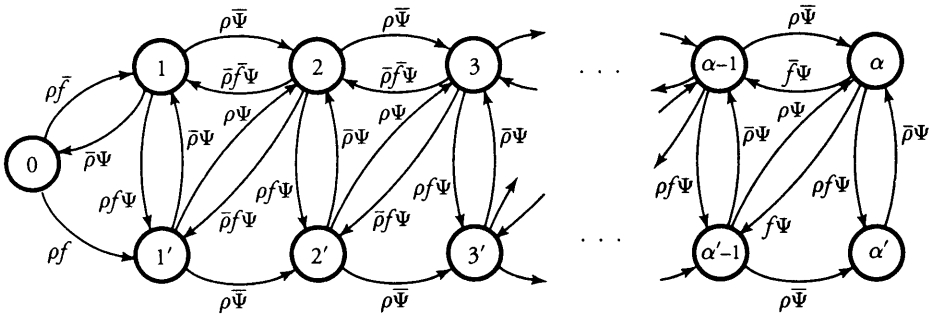

Figure 13.15 Markov chain model of the input port queue.

the crosspoint queue-set grant probability Ψ. Let f be the probability that a packet has fanout 2 (a *bipoint connection*). In this model, a bipoint connection, which is a two-copying packet-recirculation task, is treated as two independent *point-to-point* connections. The upper row of the Markov chain depicts the state of the queue when a packet needs no additional copy. However, if the packet has fanout 2, it proceeds to the second copy of the packet on the upper row once the operation for its first copy in the lower row of the chain is completed.

The state of the queue can be changed from 0 to 1 with transition probability $\rho\overline{f}$ if the packet has fanout 1, or it can enter state 1′ with transition probability ρf if the packet has fanout 2. On the arrival of a fanout-2 packet at the head of queue, a state in the upper chain is changed to a lower one with probability of $\rho f\Psi$ if a new packet arrives at the queue, or $\overline{\rho} f\Psi$ if no new packet arrives. Similarly, any of the lower chain states can be changed with probabilities $\rho\Psi$ or $\overline{\rho}\Psi$ to an upper one as soon as the first copy of a fanout-2 packet is completed. These transition probabilities are free of f, since only the first copy has been completed at this point. The second copy is processed in the upper chain independently. The probability that an input port buffer contains exactly s packets is computed recursively by

$$\begin{aligned}\pi_i(s) = {} & \pi_i(s-1)\rho\overline{\Psi} + \pi_i(s'-1)\rho\Psi + \pi_i(s)(\overline{\rho}\,\overline{\Psi} + \rho\overline{f}\Psi) \\ & + \pi_i(s')\overline{\rho}\Psi + \pi_i(s+1)\overline{\rho}\,\overline{f}\Psi \end{aligned} \tag{13.25}$$

and

$$\pi_i(s') = \pi_i(s'-1)\rho\overline{\Psi} + \pi_i(s)\rho f\Psi + \pi_i(s')\overline{\rho}\,\overline{\Psi} + \pi_i(s+1)\overline{\rho} f\Psi. \tag{13.26}$$

Next, let $w_k^{\beta}(r, s)$ be the number of distributions that leave exactly r buffers of capacity β "not full" while a queue set contains s packets, and let $W_k^{\beta}(r, s)$ be the probability that exactly r buffers are not full when a queue set contains s packets. Therefore:

$$w_k^{\beta}(r, s) = \binom{k}{r} z_r^{\beta-1}(s - (k - r)\beta), \tag{13.27}$$

and then,

$$W_k^{\beta}(r, s) = \frac{w_k^{\beta}(r, s)}{z_k^{\beta}(s)}. \tag{13.28}$$

Now, we need to compute the probability that exactly j packets enter a queue set when the queue is in state s at the beginning of the current packet cycle. The probability that a packet is available to enter a particular buffer of a queue set, a_x, is calculated based on a that was introduced earlier and is given by

$$a_x = \frac{1}{n}\left[1 - \left(1 - \frac{a}{k}\right)^n\right]. \tag{13.29}$$

Then, $p(j, s)$ is

$$p(j, s) \approx \sum_{j \le r \le k} W_k^{\beta}(r, s)\binom{r}{j}(a_x)^j(1 - a_x)^{r-j}. \tag{13.30}$$

We also need to compute the probability that exactly j packets leave a queue set when the queue is in state s at the beginning of the current packet cycle. It should be remembered that for $q(j, s)$, there is only one output for every queue set, owing to the queueing model, so the probability is introduced by a simple form of

$$q(j, s) = \begin{cases} 0 & \text{if } (s = 0 \wedge j = 1) \vee (s = 1 \wedge j = 0) \vee (j > 1) \\ 1 & \text{if } (s \ge 1 \wedge j = 1) \vee (s = 0 \wedge j = 0). \end{cases} \tag{13.31}$$

Each queue set is modeled as a $(b + 1)$-state *Markov chain* $(b = k\beta)$. The transition probability, $\lambda(\hat{s}, s)$, and the next-state probability, $\pi_x(s)$, defined earlier, can be determined as follows:

$$\lambda(\hat{s}, s) = \sum_{h=\max\{0, s-\hat{s}\}}^{\min\{k, b-s\}} p(h, \hat{s})q(h - (s - \hat{s}), \hat{s}), \tag{13.32}$$

where $0 \leq (s \text{ and } \hat{s}) \leq b$, and

$$\pi_x(s) = \sum_{\hat{s}=\max\{0,s-k\}}^{\min\{b,s+k\}} \pi_x(\hat{s})\lambda(\hat{s}, s)\cdot \tag{13.33}$$

By having $\pi_x(s)$, we can estimate the system delay and throughput.

13.8.3 Throughput and Delay

By putting together all the preceding parameters, we can compute the system *throughput* (T) and *delay* (D). The throughput refers to the number of packets leaving a queue set per output link per packet cycle. Since the entire network is modeled as one middle-stage queue, the throughput, on the other hand, is the probability of having at least one packet in a queue set and is simply

$$T = 1 - \pi_x(0)\cdot \tag{13.34}$$

Clearly, from the definition given by Equation (13.31), $q(j, s) \in \{0, 1\}$. Thus, so long as at least one packet is in the queue set, $q(j, s) = 1$, and T can be expressed free of $q(j, s)$. The *delay* is defined as the number of packet cycles that a given packet spends in the crossbar from network entry until network exit. This is another interpretation of the ratio of *queue length* and the *arrival rate*. Let the queue length of the network and the input port processors be denoted by Q_x and Q_i, respectively. They are expressed by

$$Q_x = \sum_{0\leq s\leq b} s\pi_x(s) \tag{13.35}$$

and

$$Q_i = \sum_{0\leq j\leq \alpha} j\pi_i(j)\cdot \tag{13.36}$$

The arrival rate at the input port buffers is the offered load, ρ, and the arrival rate for each queue set can be defined by A:

$$A = \sum_{0\leq j\leq k} \sum_{0\leq s\leq b} \pi_x(s)jp(j, s)\cdot \tag{13.37}$$

Thus, the total delay is the summation of the delays:

$$D = \frac{Q_x}{A} + \frac{Q_i}{\rho}\cdot \tag{13.38}$$

The delay (D) measurement is based on three main variables: ρ, n, and k. Figure 13.16 shows a set of results. This figure shows variations of incoming traffic load

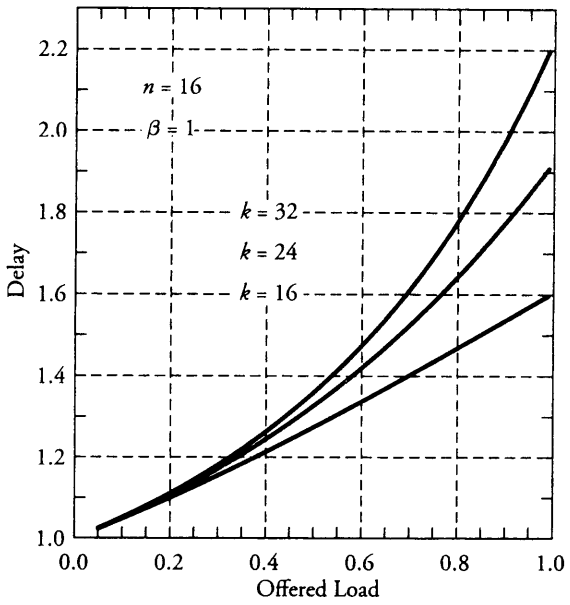

Figure 13.16 Delay (D) of a 16-port network when offered load (ρ) and number of columns (k) vary

on a 16-port MBC network. In these examples, $\beta = 1$, and the fraction of multipoint packets is assumed to be $f = 0.75$. As expected, the throughput is improved when k increases as 16, 24, 32, mainly because of the availability of a larger number of columns for packets waiting in the input buffers. Delay curves show that the average time that packets spend in the network cannot exceed 2.2 packet times. This amount of time was predictable, since packets pass through only two stages of switching from an input port to their destinations. This behavior explains the better performance capability of an MBC switch fabric compared to a typical multistage switch fabric.

13.9 Summary

Switch fabrics are the core segments of switching devices as routers. A *crossbar* is a nonblocking switch and can be used as the building block of other switch fabrics. A crossbar requires n^2 crosspoints. Blocking switches constructed with crossbar switches are *Omega*, *Banyan*, and *Delta* networks, each requiring $\log_d n$ stages; the fourth, the *Beneš* network, requires $2 \log_d n - 1$ stages. The Beneš network consists of a combined Delta network and its mirrored Delta network. *Clos networks* are nonblocking switches constructed with crossbars. A $C^3_{n,d,k}$ Clos network is strictly nonblocking if $k \geq 2d - 1$.

Concentration-based and *expansion-based* switching networks are two special-purpose switch fabrics.

In a different approach for switching, *time domain*, a memory controlled by another memory, without using any crossbar, performs the task of switching. Finally, various techniques such as deploying buffers in the structure of switches, offer better performance. The case study presented a switch fabric constructed with a crossbar and buffers in its structure. We estimated the delay and throughput for this switch.

Having finished networking issues in switch fabrics, we are ready to study the basics of optical switching networks and other optical networking topics. An optical network is the backbone of high-speed networking and is presented in the next chapter.

13.10 Exercises

1. Compare the complexity of crossbars and Delta networks in one plotting chart, in which the network size varies over the range $n = 2^2$, 2^4, and 2^5. For the Delta networks, assume that $d = 2$.
2. For the following switching networks $D_{16,2}$ and $D_{16,4}$.
   (a) Sketch the networks.
   (b) Compare $D_{16,2}$ and $D_{16,4}$ in terms of complexity and communication delay.
3. For the following switching networks $\Omega_{16,2}$ and $\Omega_{16,4}$:
   (a) Sketch the networks with the minimum number of stages that provide all possible input/output connections.
   (b) Compare the two networks in terms of complexity and communication delay.
4. Using the blocking-probability estimation method, derive an expression for the internal blocking of the two networks $\Omega_{16,2}$ and $\Omega_{16,4}$.
5. For the following switching networks, $B_{16,2}$ and $B_{16,4}$:
   (a) Sketch the networks.
   (b) Compare $B_{16,2}$ and $B_{16,4}$ in terms of complexity and communication delay.
6. Using Lee's method (discussed in Chapter 7), derive an expression for the internal blocking of $B_{16,2}$ and $B_{16,4}$ networks.
7. For the following switching networks $Y_{16,2}$ and $Y_{16,4}$:
   (a) Sketch the networks.
   (b) Is there any self-routing rule existing for Banyan networks?
8. For the following switching networks $B_{9,3}$ and $\Omega_{9,3}$:
   (a) Sketch the networks.

(b) Compare $B_{9,3}$ and $\Omega_{9,3}$ in terms of complexity, internal blocking, and communication delay.

9. Delta networks can be improved by extending stages of switching. For d, s, and h with $h < s$ and $n = d^s$, we define the extended Delta network $D^E_{n,d,h}$ as $D^E_{n,d,h} = D_{d^h,d} D_{d^{k-h},d} D_{d^h,d}$.
   (a) Sketch a picture of an extended Delta network with eight inputs/outputs, using switch elements of size 2.
   (b) Derive an equation for the complexity of this network.
   (c) Derive an equation for the blocking probability.

10. Design an $n = 8$ port three-stage Clos switching network.
   (a) Find d and k that meet the minimum nonblocking condition, and sketch the network.
   (b) To design a nonblocking Clos network, the network is nonblocking as long as $k = 2d - 1$. Why would anyone want to design a network with $k > 2d - 1$?

11. Compare the complexity of large-scale three-stage and five-stage Clos switching systems, where the number of ports varies from 1,000 ports to 2,000 ports.

12. For any nonblocking switching network, Lee's method can still estimate some blocking. Justify this contradiction.

13. There are two ways to design a three-stage Clos network with six inputs and six outputs yielding the "minimum" nonblocking operation.
   (a) Select dimensions for both types of crossbars used in either of the two types of networks, and sketch the networks.
   (b) Show a Lee's model for both networks, and find the total blocking probability for any of the networks.
   (c) Which choice is better? (Compare them from all possible aspects.)

14. To design a five-stage Clos network with $n = 8$ and $d = 2$ yielding "minimum" nonblocking operation:
   (a) Select dimensions for all three types of crossbars—no complexity optimizations needed—and sketch the network.
   (b) Show a Lee's model.
   (c) Assuming the probability that any link in the networks is busy is $p = 0.2$, find the total blocking probability for the network.

15. To increase the speed and reliability of the five-stage Clos network $C^5_{8,2,3}$ using $C^3_{4,2,3}$, we form three multiplexed parallel planes, each of which contains this network.

(a) Sketch the overview of the network.
(b) Show a Lee's model.
(c) Assuming the probability that any link in the networks including multiplexing links is busy is $p = 0.2$, find the total blocking probability for the network.

16. To design a five-stage Clos switching network with n inputs and n outputs:
(a) Give dimensions for all five types of crossbars to yield nonblocking operations.
(b) Select dimensions for all five types of crossbars to yield nonblocking operation and minimum crosspoint count. (No optimizations is needed.)

17. Design a shared-memory switching system that supports up to 16 links and uses a multiplexer, RAM, and a demultiplexer. The multiplexer output carries 16 channels, each as wide as a segment (packet fragment). Each segment is as large as 512 bytes, including the segment header for local routing. A total of 0.4 μs form a frame at each multiplexer. RAM is organized in 32-bit words with 2 ns write-in time, 2 ns read-out time, and 1 ns access time (controller process) per word.
(a) Find the minimum required size of RAM.
(b) Find the size of a RAM address.
(c) Find the maximum bit rate that can be supported at the output of RAM.
(d) Find the speed of this switch, which is the segment-processing rate per second per switch port.

18. *Computer simulation project*. Write a computer program to simulate a 2×2 crossbar switch.
(a) Assign each packet a switch-output destination at random.
(b) Construct and simulate a single crosspoint. Clearly demonstrate how it switches on and off, using a central crossbar controller.
(c) Extend your program to construct a four-crosspoint crossbar.

19. *Computer simulation project*. Carry the program you developed for a single buffer in Chapter 11 (computer simulation project) and extend the preceding 2×2 crossbar switch to a switch with a simple buffered input port. Each of the two inputs of the switch has a buffer with $K = 64$ buffer slots, and each slot can fit only in a packet of size 1,000 bytes. Dynamically assign packets of different size every 1 ms, and send out a packet to the switch every t seconds from the buffer.
(a) Assign each packet a switch-output destination at random.
(b) Construct, simulate, and test each of the two buffers individually.
(c) Integrate these two buffers with the switch, and measure the performance metrics of both buffers together, given different values of t.
(d) Measure average delay for a packet.

CHAPTER 14

Optical Networks and WDM Systems

Optical communication systems have been developed to meet two main objectives: reaching higher data transmission bandwidths and reducing the overall cost of communication networks. Optical communication technology uses principles of light emission in a glass medium, which can carry more information over longer distances than electrical signals can carry in a copper or coaxial medium. This chapter focuses on principles of optical communications, optical multiplexing, and switching used in optical computer networks. Following are the major topics covered:

- *Overview of optical networks*
- *Basic optical networking devices*
- *Large-scale optical switches*
- *Optical routers*
- *Wavelength allocation in networks*
- *Case study of SSN, an all-optical switch*

Some basic optical devices are *optical filters*, *wavelength-division multiplexers* (WDM), *optical switches*, and *optical buffers* and *delay lines*. Optical switches are given special attention, as they are core engine of all-optical networks. Contention-resolution methods for optical switches are explained and types of switch elements identified.

In optical networks, a number of optical routing devices are interconnected by optical links. The routing devices have optical multiplexing, switching, and routing

components, and links are optical fibers. An important issue in optical networks is *wavelength reuse and allocation*. To keep lightpaths separated on a link, they should be allocated different wavelengths. The case study at the end of the chapter describes an optical switching network with a *spherical switching network* (SSN) topology.

14.1 Overview of Optical Networks

Optical fibers can transmit digitized light signals over long distances because of the purity of glass fiber combined with improved electronics technology. With some acceptable transmission loss, low interference, and high-bandwidth potential, an optical fiber is almost an ideal transmission medium. Optical communication systems may sometimes be combined with electrical components. In such systems, electrical data bits are converted into light, using a certain *wavelength*; when the transmission is completed, the optical signal is converted back to an electrical signal, which is then passed to higher layers that manage switching, restoration, and grooming.

Optical networks can provide more bandwidth than regular electronic networks can. Major advantages of partial optical networks are gained by incorporating some electronic switching functions. The optical network shown in Figure 14.1 consists of *optical nodes* interconnected with physical *optical links*. A point-to-point physical optical link may consist of several logical paths called *lightpaths*, each carrying a different wavelength. A lightpath is set up by assigning a dedicated wavelength to it. The data can be sent over the lightpath once it is set up. A lightpath can create virtual neighbors out of nodes that may be geographically far apart in the network. A lightpath uses the same wavelength on all fiber links through which it passes.

14.1.1 Protocol Models and Standards

Optical networks provide routing, grooming, and restoration of data at the wavelength level and developed from the emergence of the optical layer in the transport and network layers. The challenges of optical networks are network management in the presence of different wavelengths and keeping the cost down to be able to add new services. An optical network's agility gives it the speed and intelligence required for efficient routing. Thereby improving network management and faster connections and network restorations. For an optical system with many channels on a single fiber, a small fiber cut can potentially initiate multiple failures, causing many independent systems to fail.

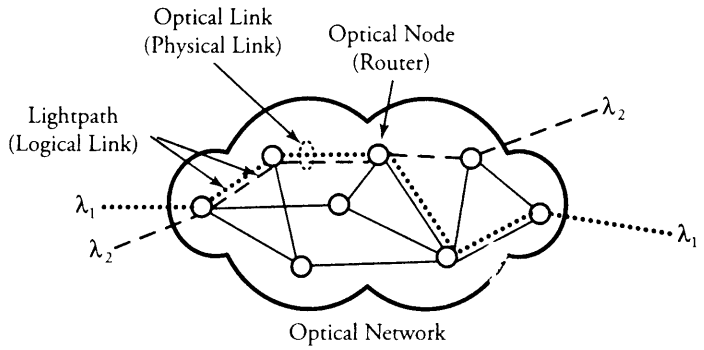

Figure 14.1 Overview of an optical network

Several standards apply in the optical communication environment. The widely used *synchronous optical network* (SONET) standard provides the transport infrastructure for worldwide telecommunications and defines interface standards at the physical layer of the protocol stack. SONET also defines a hierarchy of interface rates that allow data streams at different rates to be wavelength multiplexed. SONET establishes *optical carrier* (OC) levels ranging from level 1 at 51.8 Mb/s (OC-1) to level-192 at 9.95 Gb/s (OC-192). Communication carriers throughout the world use this technology to interconnect their existing digital carriers and fiber optic systems.

In optical networks, client layers and optical layers are managed separately, with no direct interaction between the clients and optical-layer equipment. A centralized network-management system provides optical-layer connections. However, distributed control protocols are used to handle network protection and restoration. Because of the centralized management scheme for connection provisioning, a network may not be reliable. From a control and management perspective, there is a great deal of interest in obtaining a closer interaction between the network layer and the optical layer.

Two popular models are the *overlay model* and the *peer model*. In the overlay model, the optical layer and the network (IP) layer have their own independent control planes. Here, the client layer interacts with the optical layer through a *user-to-network interface* (UNI), and optical-layer elements talk to each other through a *network-to-network interface* (NNI). In an overlay model, the optical network topology is hidden from the client layer through UNI. In the peer model, the network layer and the optical layer run the same control-plane software. Routers have full topology awareness of the optical layer. But the peer model is complicated because the optical layer imposes

significantly different constraints from those on the IP layer. This elegant model has a closer coupling between the IP layer and the optical layer.

14.2 Basic Optical Networking Devices

An optical network comprises optical devices interconnected by optical transmission links. Other basic elements on an optical networks are: tunable lasers, optical buffers or delay elements, optical amplifiers, optical filters, wavelength-division multiplexers, and optical switches.

14.2.1 Tunable Lasers

Tunable lasers can continuously change their emission wavelengths, or colors, in a given spectral range. These changes work in collaboration with optical switches to select a particular wavelength for connection establishment. A *tunable dispersion compensator* is used to compensate for fiber-dispersion losses over the length of the fiber.

14.2.2 Optical Buffers or Delay Elements

Buffering in optical nodes also faces serious limits forced by optical-technology shortcomings. *Optical buffers*, or *optical delay elements*, can be implemented by using a certain length of fiber to delay signals. No practical optical memory is available with the current technology.

14.2.3 Optical Amplifiers

Often in non-all-optical networks, a signal may not be able to remain in optical form and may have to be regenerated, requiring the *amplification* of the signal or the conversion of the signal from optical to electronic. The use of *optical amplifiers* allows achieving large-distance communication without the need of a regenerator. A major milestone in the evolution of optical fiber transmission systems was *Erbium-doped fiber amplifiers* (EDFAS), or simply optical amplifiers. An optical amplifier can amplify signals at many wavelengths simultaneously. The amplification of a signal involves the extraction of the clock from the signal and reclocking the signal, resulting in the creation of "speed bumps" in a network.

14.2.4 Optical Filters

Signal filtering is often needed in optical networks. An *optical filter* equalizes the gain of transmission systems and filters the noise or any unwanted wavelength. In

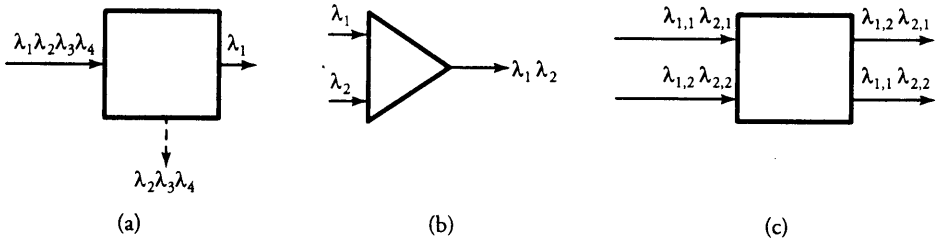

Figure 14.2 Basic communication devices for optical networks: (a) optical filter; (b) wavelength-division multiplexer; (c) optical switch

Figure 14.2 (a), an optical cable carries four wavelengths, λ_1, λ_2, λ_3, and λ_4, and is connected to a filter. This particular filter is designed to allow only λ_1 to pass and to filter λ_2, λ_3, and λ_4.

The design of an optical filter has a number of challenging factors. *Insertion loss* is one and is the loss of power at a filter. A low insertion loss is one of the specifications for a good optical filter. The state of polarization of the input signals should not affect the loss.

Keeping the temperature at the desired degree in optical systems, especially on filters, is another factor. Temperature variation should not affect the passband of a filter. As a result, wavelength spacing between adjacent channels on transmission systems should be large enough so that the wavelength shift does not affect the entire operation. The passband of filters in an optical system can be narrower if a number of filters are cascaded. The objective is that at the end of cascading the broad passband, each filter causes a small change in operating wavelengths.

14.2.5 Wavelength-Division Multiplexer (WDM)

Wavelength-division multiplexing (WDM) transmission systems divide the optical-fiber bandwidth into many nonoverlapping optical wavelengths: so-called WDM channels. As shown in Figure 14.2 (b), a WDM mixes all incoming signals with different wavelengths and sends them to a common output port. A demultiplexer does the opposite operation, separating the wavelengths and dispatching them onto output ports. On the common link, each channel carries information at light speed with minimal loss. This multiplexing mechanism provides much higher available transmission capacity in communication networks.

14.2.6 Optical Switches

An *optical switch* is the heart of an optical network. The objective in using optical switches rather than semiconductor switches is to increase the speed and volume of traffic switching in a core node of a computer communication network. The optical switch acts as an *optical cross-connect* (OXC) and can accept various wavelengths on network input ports and route them to appropriate output ports. The optical switch performs three main functions:

1. Routes all wavelengths of an incoming fiber to a different outgoing fiber
2. Switches specific wavelengths from an incoming fiber to multiple outgoing fibers
3. Takes incoming wavelengths and converts them to another wavelength on the outgoing port

Figure 14.2 (c) shows a simple 2 × 2 optical switch. A signal with wavelength i arriving at input j of the switch denoted by $\lambda_{i,j}$ can be switched on any of the four output ports. In this figure, $\lambda_{1,1}$ and $\lambda_{2,1}$ arrive on the first input port of the switch; $\lambda_{1,1}$ can be switched on output port 2, and $\lambda_{2,1}$ can be forwarded on output port 1. This basic optical switch is a switch element in larger-scale switch architectures. The basic switch can be made using various technologies. Such switch elements are broadly classified as either *non-electro-optical* or *electro-optical*.

Classification of Switch Elements

Non-electro-optical switches have simple structures. For example, a *mechanical optical switch* uses mirrors at a switch's input and output ports. A switch can be controlled by moving the mirrors and directing a light beam to a desired direction, and eventually to the desired output port. The advantages of mechanical switches are low insertion, low cross talk, and low cost. However, they have low speed. Another example is the *thermo-optic switch*, which is built on a waveguide. In this type of switch, a change of temperature in the thermo-optic structure of the waveguide can change the refractive index of the material, thereby allowing a switching function to be formed. Although similar to mechanical switches, thermo-optical switches operate at low speeds, but their cross talk is lower.

Figure 14.3 shows a typical 2 × 2 *electro-optic switch*, which uses a *directional coupler*. A 2 × 2 directional coupler is an integrated waveguide that can combine or split signals at its output ports. A coupler is the building block of filters, multiplexers, and switches and is known as a *star coupler*. A star coupler operates by changing the refractive index of the material placed in the coupling region. The switching function can be achieved by applying an appropriate voltage across the two electrodes.

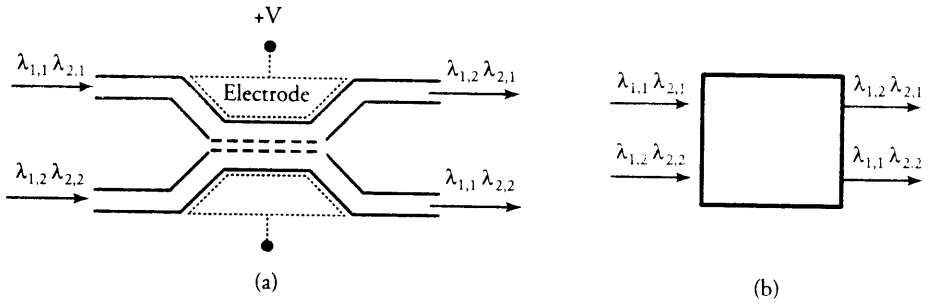

Figure 14.3 Smallest optical switch constructed with directional coupler: (a) architecture; (b) symbol

The switching function is achieved by taking a fraction, Φ, of the power from input port 1 and placing it on output port 1, with the remaining fraction $1 - \Phi$ of the power on output port 2. Similarly, a fraction $1 - \Phi$ of the power from input port 2 is placed on output port 1, with the remaining power fraction Φ on output port 2. The advantages of this switch are its speed and a modest level of integration to larger modules. However, the high cost of its production is a disadvantage of this switch over the other switch types.

Contention Resolution

Designing large-scale switching devices is one of the main challenges in optical networks. Ideally all the functions inside an optical node should be performed in the optical domain. However, packet contention in switches can be processed only in electronic devices. Several technological factors bring restrictions to the design of an optical switch.

The expansion of a fixed-size switch to higher scales is also a noticeable challenge. Electronic switches have much better flexibility of integration than do optical devices. Some performance factors other than switching speed should be considered in order to prove the suitability of a switch for optical networks. As do regular switches, optical switches face the challenge of *contention resolution* within their structures. Contention resolution is of the following three types:

1. *Optical buffering*. Although buffering is a challenge in an optical environment, optical buffers are implemented using fixed-length delay fibers.
2. *Deflection routing*. If two or more packets need to use the same output link, only one is routed along the desired output link; the others are deflected onto undesired paths directed to the destination through a longer route but with higher priorities.
3. *Wavelength conversion*. By changing wavelengths, signals can be shuffled and forwarded onto other channels.

Another important factor in optical switches is *insertion loss*, which is the fraction of power lost in the forward direction. This factor should be kept as small as possible in optical switches. Sometimes, the dynamic range of the signals must be increased in switches just because the level of loss may not be acceptable for the corresponding particular connection.

The factor of *cross talk* in switches must be minimized. Cross talk is the ratio of output power from a desired input to the output power from all other inputs. At issue is the amount of energy passing through from adjacent channels. This energy can corrupt system performance. The cost of optical devices, especially WDMs, is high compared to that of electronic modules. The production of all-fiber devices can also be a way to reduce the cost.

14.3 Large-Scale Optical Switches

Large-scale optical switches can be achieved either by implementing large-scale star couplers or by cascading 2×2 or even 1×2 multiplexers. However, the implementation of star couplers larger than 2×2 is expensive, owing to difficulties in the manufacturing process. Cascading small switches is a practical method to expand a switch. This method of switch expansion can make a desired-size switch quickly and at lower expense. However, some factors affect the overall performance and cost of an integrated switch, as follows:

- *Path loss*. Large-scale switches have different combinations of switch elements; therefore, a signal may experience different amounts of losses on different paths. Hence, the number of switch elements cascaded on a certain path might affect the overall performance of the switch.
- *Number of crossovers*. Large optical switches can be manufactured on a single substrate by integrating multiple switch elements. In an integrated optical system, a connection between two switch elements is achieved by one layer of a waveguide. When paths of two waveguides cross each other, the level of cross talk increases on both paths. As a result, the number of crossovers in an optical switch must be minimized.
- *Blocking*. As discussed in Section 13.1, a switch is *wide-sense nonblocking* if any input port can be connected to any unused output port without requiring a path to be rerouted; is *rearrangably nonblocking* if, for connecting any input port to an unused output port, a rearrangement of other paths is required.

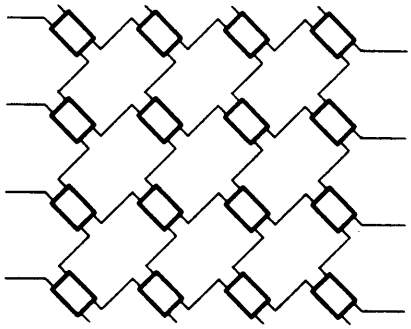

Figure 14.4 A 4 x 4 wide-sense nonblocking optical crossbar without waveguide crossovers, using basic 2 x 2 switch elements

There are several proposed topologies for large-scale switching networks. Two of the more practical ones are *crossbar* and the *Spanke-Beneš network* architectures.

14.3.1 Crossbar Switching Network

The architecture of an optical *crossbar* is not quite the same as the one in Chapter 13, as indicated in Figure 14.4. An optical crossbar is wide-sense nonblocking. The fundamental difference in the architecture of the optical crossbar and traditional crossbars is the existence of different and variable-length paths from any input to any output. This feature was created in the structure of the optical crossbar to minimize the crossover of interconnections. An optical crossbar is made up by 2×2 switches. Denoting the number of switches a signal faces to reach an output port from an input port by ℓ, the bounds for ℓ can be investigated from the figure as

$$n \leq \ell \leq 2n - 1, \tag{14.1}$$

where n is the number of inputs or outputs of the crossbar. In an $n \times n$ crossbar, the number of required 2×2 switches is n^2. The 16 2×2 switches in the Figure 14.4 crossbar are structured in an appropriate way. The main issue with the crossbar structure is its cost, which is a function of n^2. But a great advantage of such an architecture is that the switch can be manufactured without any crossovers.

14.3.2 Spanke-Beneš Switching Network

Figure 14.5 shows a rearrangably nonblocking switch called a *Spanke-Beneš switching network*. This network is designed with identical 2×2 switch elements, and there are

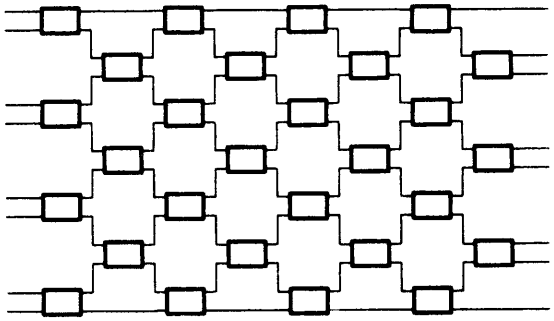

Figure 14.5 An 8 x 8 Spanke-Beneš switching network with rearrangeably nonblocking structure without requiring any interconnection crossover

no crossover paths. This switching network is also known as n-stage planar architecture. Denoting the number of switches a signal faces to reach an output port from an input port by ℓ, we can derive the bounds for ℓ by

$$\frac{n}{2} \leq \ell \leq n. \tag{14.2}$$

This inequity can be verified from Figure 14.5, which shows multiple paths between each input/output pair. Thus, the longest possible path in the network is n, and the shortest possible path is $n/2$. This arrangement is made to achieve a moderately low blocking. Thus, there are n stages (columns) and $\frac{n(n-1)}{2}$ switch elements in an $n \times n$ switching network.

14.4 Optical Routers

An *optical router*, known as the *wavelength router*, or simply a network *node*, is a device that directs an input wavelength to a specified output port. Each router in a network is thus considered a node. Optical nodes are classified as either *broadcast nodes* or *wavelength routing nodes*.

In a broadcast node, a wavelength is broadcast by a passive device to all nodes. The passive device can be an *optical star coupler*. A coupler combines all incoming signals and delivers a fraction of the power from each signal on to each output port. A *tunable optical filter* can then select the desired wavelength for reception. Broadcast nodes are simple and suitable for use in access networks, mainly LANs. The number of optical links connected to a broadcast node is limited, since the wavelengths cannot be reused in the network.

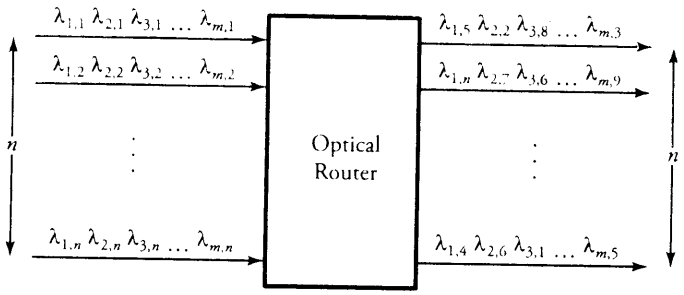

Figure 14.6 Overview of a wavelength routing node

Wavelength routing nodes are more practical. Figure 14.6 shows an overview of an $n \times n$ wavelength routing node in which full connectivity is ensured between n inputs and n outputs, using m wavelengths. In general, a signal with wavelength i arriving at input j of the router denoted by $\lambda_{i,j}$ can be switched on any of the n output ports. For example, a light with wavelength $\lambda_{3,2}$ among all wavelengths $\{\lambda_{1,2}, \lambda_{2,2}, \lambda_{3,2}, \cdots, \lambda_{m,2}\}$ arrives on the second input port of the router. This wavelength can be switched on any of the outputs and, possibly, with a different wavelength, as $\lambda_{4,7}$.

14.4.1 Structure of Wavelength Routing Nodes

Wavelength routers are of two types. The first type can switch an input only to an output using the same wavelength. Figure 14.7 gives an overview of this router. Consider this router with n inputs and n outputs, where each input i can bring up to m wavelengths as $\{\lambda_{1,i}, \lambda_{2,i}, \lambda_{3,i}, \ldots, \lambda_{m,i}\}$. In this case, any wavelength $\lambda_{k,i}$ can be switched on output j with wavelength $\lambda_{k,j}$. This way, in each round of switching, all wavelengths at the input ports with the same wavelengths can be switched to desired output ports taking place in the same corresponding channels. As shown in Figure 14.7, the router consists of n input WDMs, m optical switches of size $n \times n$, and n output WDMs.

The second type of wavelength router requires *wavelength-conversion* capabilities, as illustrated in Figure 14.8. Consider this router with n inputs and n outputs with up to m wavelengths per input. With this router, a wavelength $\lambda_{k,i}$ at input i can be switched on output port j; in addition, the kth wavelength can be converted to the gth, to be shown by $\lambda_{g,j}$. This way, in each round of switching, any wavelength at the input port with the same wavelengths can be switched on any output port, using any

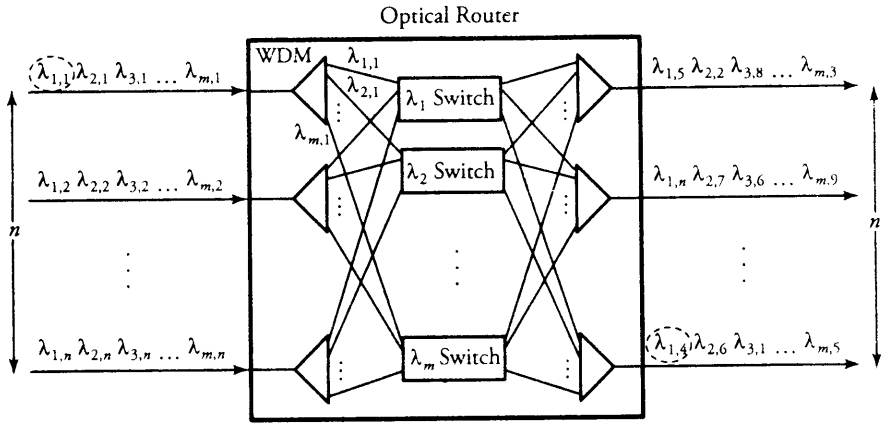

Figure 14.7 An all-optical router that replaces wavelength channels

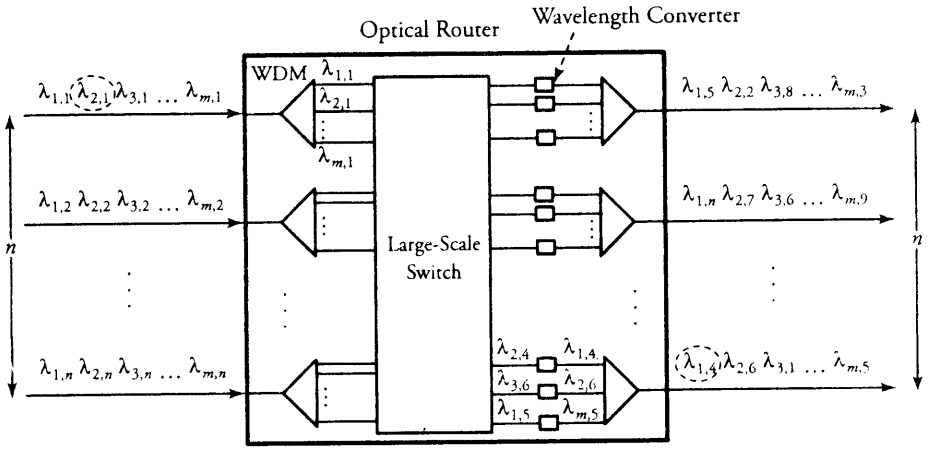

Figure 14.8 An all-optical router, which can replace wavelength channels and convert wavelengths

available wavelength. In Figure 14.8, $\lambda_{2,1}$ from input 1 is switched to $\lambda_{1,4}$ on the nth output.

In general, wavelengths may be required to be converted in a network for the following reasons:

- The wavelength of a signal entering the network may not be appropriate at a node.
- A node may realize the availability of different and better wavelengths on the network links.

Depending on whether a wavelength converter converts a fixed- or variable-input wavelength to a fixed- or variable-output wavelength, there are different categories for wavelength converters, such as fixed-input fixed-output, fixed-input variable-output, variable-input fixed-output, and variable-input variable-output.

Semi-Optical Routers

Depending on the type of a switch fabric used in the structure of an optical router, we can further classify optical routers into two main groups:

1. Routers with *all-optical switches*, in which signals traverse the network entirely in the optical domain.
2. Routers with *optically opaque optical switches*, in which some forms of optoelectronic processing take place within multiplexing elements.

All-optical, or *photonic*, *switches*, are the ones we have studied so far. In such switches, the optical signal need not be converted to electrical, and switching is handled in the optical domain. All-optical domains have an advantage of large-capacity switching capabilities. In the optically opaque, or so-called semioptical switches, an optical signal is converted to electrical, and then the appropriate switching takes place. It is obviously a case with an electrical switch core followed up with optical interfaces.

In an all-optical switch, switching and wavelength conversion take place entirely in the optical domain. An optically opaque switch uses an optoelectronic conversion mechanism. This architecture requires other interfaces, such as optical-to-electrical converters or electrical-to-optical converters. The electronic segments are used around the optical switch matrix.

14.5 Wavelength Allocation in Networks

Similar to nonoptical networks, optical networks consist of a number of routing nodes connected by communication links. In optical networks, nodes may have optical multiplexing, switching, and routing components, and links are optical fibers. In the early generations of optical fibers, transmission of signals was degraded on short distances mainly because of lack of amplification. In later versions of optical-fiber systems, the loss of signals was significantly reduced through carefully designed amplifiers. In the latest generations of optical communication systems, the overall cost has been lowered, the effect of dispersion has been eliminated, the data bit rate has been enhanced up to beyond tens of gigabits per second, optical amplifiers has replaced regenerators, and WDM technology has been used for multiplexing purposes.

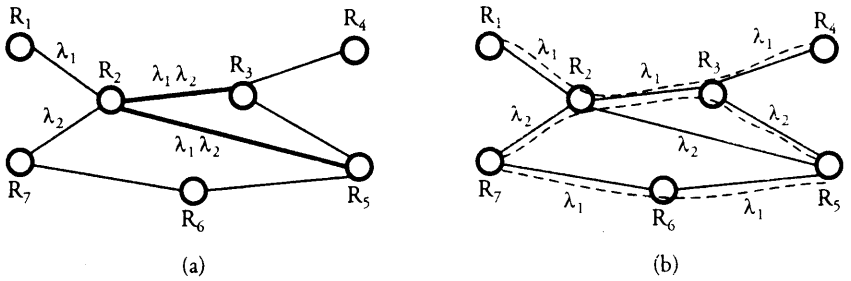

Figure 14.9 An optical network with (a) broadcast nodes and (b) wavelength routing nodes

All-optical networks can be designed to carry data using any wavelength regardless of the protocol and framing structure. All-optical networks were encouraged by the development of ultralong-haul fibers mounted on a single physical fiber. By transmitting each signal at a different frequency, network providers could send many signals on one fiber, just as though each signal were traveling on its own fiber. All-optical networks handle the intermediate data at the optical level. This efficient functioning of the network provides a number of advantages in handling data: reduced overall cost and increased system bandwidth.

A lightpath carries not only the direct traffic between the nodes it interconnects but also traffic from nodes upstream of the source to downstream of the destination. Thus, a lightpath can reduce the number of allocated wavelengths and improve the network throughput. In practice, a large number of lightpaths may be set up on the network in order to embed a virtual topology.

14.5.1 Classification of Optical Networks

Optical networks can be classified according to the type of nodes being used in the network. Figure 14.9 (a) shows an optical network of broadcast nodes. A broadcast node combines all incoming signals and delivers a fraction of the power from each signal on to each output port. A *tunable optical filter* can then select the desired wavelength for reception.

Figure 14.9 (b) illustrates a network that uses wavelength routing nodes. These types of nodes are capable of reusing wavelengths and handling many simultaneous lightpaths with the same wavelength in the network. Two lightpaths—R_1-R_2-R_3-R_4 and R_7-R_6-R_5—do not use any shared link and can therefore be assigned the same

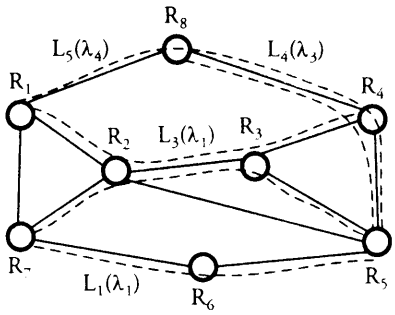

Figure 14.10 Allocation of wavelengths in an optical network

wavelength λ_1. Because lightpaths R_1-R_2-R_3-R_4 and R_7-R_2-R_3-R_5 share part of their common path (R_2-R_3), they must therefore use a different wavelength.

14.5.2 Wavelength Allocation

Consider an all-optical network in which each link in the network can carry a maximum of λ_n wavelengths. Because of the maximum wavelength capacity, a network may not be able to handle all lightpath requests, so some requests may be blocked. To keep lightpaths separated on the same link, they should be allocated different wavelengths. For example, consider the all-optical network shown in Figure 14.10, with five lightpaths: L_1 through L_5. On link R_2-R_3, the same wavelength cannot be allocated.

A lightpath with wavelength λ_i at any node's input can be converted to any available wavelength $\lambda_j \in \{\lambda_1, \cdots, \lambda_n\}$ on the node's output link. If no wavelength is available, a lightpath uses the same wavelength on all links of its path. Based on our earlier discussion, wavelength allocations for this case can be arranged as follows: L_1 is assigned λ_1, L_2 is assigned λ_2, L_3 is assigned λ_1, L_4 is assigned λ_3, and L_5 is assigned λ_4.

The concept of wavelength allocation can be analyzed in two ways. One way is to assume that the probability of a wavelength being used on a link is independent of the use of the same wavelength on all other links of the lightpath. Although this method is not practical, the analysis provides a quick approximation on how effective the assignment of wavelengths is. The second method removes the assumption of independence.

Wavelength Allocation Without Dependency

The wavelength-allocation algorithm assigns an arbitrary but identical wavelength on every link of a lightpath when one such wavelength is free on every piece of its path.

In this section, we assume that the probability of a wavelength being used on a link is independent of the use of the same wavelength on all other links of the lightpath. Consider a single link in which λ_n wavelengths are available. For each lightpath request, the first available wavelength is assigned. The wavelength-request arrival is assumed to be Poisson with the rate that leads to a utilization ρ. Then, the blocking probability on this link follows the Erlang-B formula expressed by Equation (11.46):

$$P(\lambda_n) = \frac{\rho^{\lambda_n}}{\lambda_n! \left(\sum_{i=0}^{\lambda_n} \frac{\rho^i}{i!}\right)}. \tag{14.3}$$

This formula calculates the probability that an arriving request finds no available wavelength while there is no waiting line for request. For gaining the highest efficiency on assigning wavelengths to requests, wavelengths must be reused effectively. If lightpaths overlap, the *wavelength-conversion gain* would be low.

Not only the overlap between lightpaths impacts the wavelength-conversion gain; so too does the number of nodes. Assume that for each lightpath, the route through the network is specified. Let the probability that a wavelength is used on a link be p. If the network has λ_n wavelengths on every link and a lightpath request chooses a route with r links, the probability that a given wavelength is not free on at least one of existing r links of the route can be derived by

$$P_b = 1 - \left(1 - p^r\right)^{\lambda_n}. \tag{14.4}$$

Note that for P_b, we use the rules developed in Section 7.6.4. The probability that a given wavelength is free on any given link is $(1 - p)$. Consequently, the probability that a wavelength is not free on all the r links of the path is $1 - (1 - p)^{\lambda_n}$. Using the parallel rule explained in Section 7.6.4, P_b expresses the probability that a given wavelength is not available on at least one of existing r links of the route. Similarly, the probability that a lightpath request is blocked is

$$P_r = 1 - \left(1 - p^{\lambda_n}\right)^r. \tag{14.5}$$

The wavelength allocation with dependency is much more complicated. Next, we try to present a simplified analysis for this case.

Wavelength Allocation with Dependency

In practice, the allocation of a free wavelength to a lightpath on its every link is dependent on the use of other wavelengths on the same link. Let $P(\ell_i|\hat{\ell}_{i-1})$ be the probability that a wavelength is used on link i, given that the wavelength is not used

on link $i-1$. Also, let $P(\ell_i|\ell_{i-1})$ be the probability that a wavelength is used on link i, given that the wavelength is used on link $i-1$. Then:

$$P(\ell_i|\ell_{i-1}) = P(\ell_i|\hat{\ell}_{i-1}). \tag{14.6}$$

If we substitute $P(\ell_i|\hat{\ell}_{i-1})$ for p in Equation (14.4), P_b can be reexpressed for the dependency condition as

$$P_b = 1 - \left(1 - P(\ell_i|\hat{\ell}_{i-1})^r\right)^{\lambda_n}. \tag{14.7}$$

We can now estimate the packet delay over a lightpath on link i, j from node i to node j. Assume that packet-transmission times are exponentially distributed with mean time $1/\mu$, and Poisson packet arrival distribution with mean arrival rate Λ. Let $s_{i,j}$ be the loading value on link i, j, or the number of source/destination pairs whose traffic is routed across link i, j. Then, we can obtain the average delay on link i, j by using Equation (11.21):

$$E[T] = \frac{1}{\mu - s_{i,j}\Lambda}. \tag{14.8}$$

If a network has a total of n nodes, the average queueing delay incurred for a packet through all nodes over all source/destination pairs is expressed by

$$E[T_n] = \frac{1}{n(n-1)} \sum_{i,j} \frac{s_{ij}}{\mu - s_{i,j}\Lambda}. \tag{14.9}$$

14.6 Case Study: An All-Optical Switch

As a case study on optical switching networks, we consider the *spherical switching network* (SSN), a switch fabric first introduced by the author of this book in the *Journal of Computer Networks* (issue 40, 2002). The spherical network is a regular mesh network that can be realized as a collection of horizontal, vertical, and diagonal rings, as shown in Figure 14.11. Rings appear in bidirectional pairs that can form cyclical entities and create full regularity in the structure of the network. The spherical network uses a simple self-routing scheme. Furthermore, the availability of a large number of interconnections among switch elements and the special arrangement of interconnections potentially reduce the number of deflections compared to the existing deflection-routing networks.

The network is constructed with *fixed-size* switch elements, regardless of the size of the network. Each switch element consists of a 9×9 crossbar and a local controller. Although not indicated in the figure, an incoming link and an outgoing link connect the

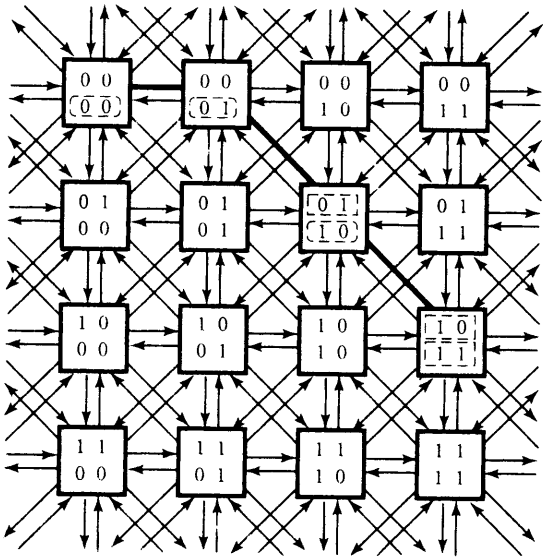

Figure 14.11 A 16-port proposed spherical switching network. Each square represents a 9 × 9 switch element with eight pairs of internal links (shown) and one pair of external links (not shown). An example for self-routing is from switch element 0000 to 1011.

network at each switch element to external links and then to local processors, which deliver and receive data sent through the network. One of the nine pairs of links is external and carries the external traffic; the other eight pairs are internal.

The spherical network doesn't use any buffering within the network, and thus this network gains from the lower complexity when compared later with other switching networks. Contention resolution in each switch element is based on the deflection of losing packets on undesired internal links and with increments in their priority fields. With the *torus*-shaped topology embedded in the network, when more than one packet requests the same outgoing link at each switch element, only one of them is forwarded on the preferred link; the others are deflected onto other links. By maintaining this rule in the system, once a packet is admitted to a network, it is not discarded when congestion occurs. Instead, the packet receives an increment in its priority field and is misrouted temporarily but will reach its destination. One main advantage of this network over many others is the existence of possible multiple equally desired outputs at each switch element. In other words, it might be possible at each point in the network to find *more than one shortest path* to a destination.

14.6.1 Self-Routing in SSN

The routing in the spherical network is fairly simple and is self-routing. Any switch element is given an index number. For example, the indices in a 16-port network are [00 – 00] through [11 – 11], as in Figure 14.11. Depending on whether the flowing traffic is directed toward decreasing index or increasing index, the self-routing is performed by decremental or incremental addressing, respectively. Clearly, the network has four routing cases:

1. Straight horizontal
2. Straight vertical
3. Straight diagonal
4. A combination of diagonal/horizontal routing or diagonal/vertical

In each case, there are a certain number of *shortest paths* through which a packet is channeled to its destination by taking a corresponding direction. In the first case, depending on whether the index number of the source is greater or smaller than the index number of the destination, the value of the second two index bits of the source address is decreased or increased hop by hop, respectively, until the address of a source and a destination becomes identical where the routing is completed.

In the second case, the process is identical to the horizontal case but in a vertical direction, and index decrement or increment is done on the value of the first two index bits of the source address. In case 3, the routing process is accomplished when the values of both the first and second two index bits of the source address simultaneously receive increment or decrement, as described in cases 1 and 2. Case 4 can be a combination of either the first and third cases or the second and third cases. In case 4, there might be more than one preferred path.

The preceding routing rules use the preferred directions to route each packet to its destination along a shortest path. Contention resolution in each switch element is based on deflection of a losing packet onto an undesired output if all preferred outputs are not available and giving an increment to the priority field of the packet.

14.6.2 Transmission in SSN

SSN can be used to construct an all-optical switch core. Suppose that any optical signal is assigned a unique wavelength. To use the fiber bandwidth efficiently, it is divided into 72 nonoverlapping optical wavelengths corresponding to 8 + 1 pairs of links at

each switch element. Each node is connected into eight other nodes through eight pairs of internal optical links plus a pair of external links. There are nine groups of wavelengths: $\Lambda(1)$, $\Lambda(2)$, $\Lambda(3)$, $\Lambda(4)$, $\Lambda(5)$, $\Lambda(6)$, $\Lambda(7)$, $\Lambda(8)$, and $\Lambda(9)$. Each group of wavelengths includes eight nonoverlapping optical wavelengths:

$$\Lambda(1) = (\lambda_{1,1}, \lambda_{1,2}, \lambda_{1,3}, \lambda_{1,4}, \lambda_{1,5}, \lambda_{1,6}, \lambda_{1,7}, \lambda_{1,8})$$
$$\Lambda(2) = (\lambda_{2,1}, \lambda_{2,2}, \lambda_{2,3}, \lambda_{2,4}, \lambda_{2,5}, \lambda_{2,6}, \lambda_{2,7}, \lambda_{2,8})$$
$$\Lambda(3) = (\lambda_{3,1}, \lambda_{3,2}, \lambda_{3,3}, \lambda_{3,4}, \lambda_{3,5}, \lambda_{3,6}, \lambda_{3,7}, \lambda_{3,8})$$
$$\Lambda(4) = (\lambda_{4,1}, \lambda_{4,2}, \lambda_{4,3}, \lambda_{4,4}, \lambda_{4,5}, \lambda_{4,6}, \lambda_{4,7}, \lambda_{4,8})$$
$$\Lambda(5) = (\lambda_{5,1}, \lambda_{5,2}, \lambda_{5,3}, \lambda_{5,4}, \lambda_{5,5}, \lambda_{5,6}, \lambda_{5,7}, \lambda_{5,8})$$
$$\Lambda(6) = (\lambda_{6,1}, \lambda_{6,2}, \lambda_{6,3}, \lambda_{6,4}, \lambda_{6,5}, \lambda_{6,6}, \lambda_{6,7}, \lambda_{6,8})$$
$$\Lambda(7) = (\lambda_{7,1}, \lambda_{7,2}, \lambda_{7,3}, \lambda_{7,4}, \lambda_{7,5}, \lambda_{7,6}, \lambda_{7,7}, \lambda_{7,8})$$
$$\Lambda(8) = (\lambda_{8,1}, \lambda_{8,2}, \lambda_{8,3}, \lambda_{8,4}, \lambda_{8,5}, \lambda_{8,6}, \lambda_{8,7}, \lambda_{8,8})$$
$$\Lambda(9) = (\lambda_{9,1}, \lambda_{9,2}, \lambda_{9,3}, \lambda_{9,4}, \lambda_{9,5}, \lambda_{9,6}, \lambda_{9,7}, \lambda_{9,8})$$

Wavelength groups $\Lambda(1)$ through $\Lambda(8)$ are assigned to the eight pairs of internal links, and $\Lambda(9)$ is assigned to the external links. At each incoming fiber link, there are at most eight packets multiplexed into one fiber. All packets are destined to different addresses. When packets arrive at a node, they are demultiplexed, and the node makes the routing decision for each packet. After a next hop for each incoming packet is decided, the wavelength of that packet is converted to one of the available wavelengths by wavelength converter and is then transmitted to the next switch element.

14.7 Summary

This chapter focused on optical communication networks, beginning with basic definitions and a review of such optical devices as *optical filters*, *wavelength-division multiplexers*, *optical switches*, and *optical buffers* and *optical delay lines*.

Optical switches can be classified as *non-electro-optical* switches using non-electro-optical devices such as *mechanical optical switches* or *thermo-optic switches*, and *electro-optic* switches using *directional couplers*. The cost of large-scale optical switches constructed by a one-segment optical switch is high. Therefore, several topologies of large-scale switching networks use simple switch elements, such as *crossbar* and *Spanke-Beneš network* architectures.

Optical networks constructed with optical devices provide routing, grooming, and restoration of data at the wavelength level. Two popular models for managing the net-

work (IP) layer and the optical layer are the *overlay model* and the *peer model*. In the overlay model, the optical layer and IP layer each have their own independent control planes.

Wavelength reuse and allocation in optical networks is a key topic. Because of the maximum capacity on the number of wavelengths that can be carried on a link, a network may not be able to handle all lightpath requests; consequently, some requests may be blocked. To keep lightpaths separated on a same link, they should be allocated different wavelengths. Wavelength allocation can be analyzed in two ways. One way is to assume that the probability of a wavelength being used on a link is independent of the use of the same wavelength on all other links of the lightpath. The other method does not make any assumption of independence.

An all-optical switching network called *spherical switching network* (SSN) was presented as a case study. Routing in that network is fairly simple and self-routing. Any switch element is given an index number. Depending on whether the flowing traffic is directed toward decreasing index or toward increasing index, the self-routing is performed by decremental or incremental addressing respectively.

The next chapter discusses multicast protocols and algorithms. In addition, switching techniques at the node level are used to show how copies of packets are made in switch fabrics.

14.8 Exercises

1. Consider a crossbar switching network similar to the one shown in Figure 14.4 but of size 8×8. Suppose that the crosstalk suppression of each 2×2 switch element is 40 dB.
   (a) Calculate the maximum and minimum overall cross-talk suppressions for the crossbar switch.
   (b) Calculate the average of overall cross-talk suppressions, using all possible existing paths.
2. Consider the Spanke-Beneš network shown in Figure 14.5. Suppose that the cross-talk suppression of each 2×2 switch element is 40 dB. Calculate the overall cross-talk suppressions, using all possible existing paths.
3. To design an 8×8 optical router, we are comparing three different structures, each using, respectively, a crossbar switching network, a Spanke-Beneš network, and an 8×8 directional coupler.
   (a) Which structure offers the best overall cross-talk suppression?
   (b) Which structure has the lowest overall average switching delay?

4. Consider a ring optical backbone network consisting of eight 2×2 switching nodes labeled 1 through 8. Nodes are interconnected with two fiber-optic rings carrying traffic in opposite directions. Three wavelengths, λ_1, λ_2, and λ_3, are available on each link.
   (a) A SONET network consisting of duplex lightpaths 2-4, 4-7, and 3-6 is constructed over this backbone. For convenience, assume that both halves of the duplex SONET network are identical in terms of wavelengths. Find the minimum number of wavelengths to construct the SONET network.
   (b) Now, assume that another SONET network, consisting of duplex lightpaths 2-5, 5-6, and 2-8, is constructed over this backbone. Using the same assumptions as in part (a), find the minimum number of wavelengths to construct this SONET network.
   (c) Both SONET networks are supposed to be constructed over this backbone network simultaneously. Find the minimum number of wavelengths to construct the two SONET networks.
5. Suppose that four wavelengths exist on each single link of a three-link path. For each lightpath request, the first available wavelength is assigned. The wavelength request arrival is assumed to be Poisson, with the rate that leads to 80 percent utilization. For each lightpath, the probability that a wavelength is used on a link is 20 percent. Assume that a lightpath request chooses a route with two links.
   (a) Find the blocking probability on this link.
   (b) Find the probability that a given wavelength is not free on at least one of existing two links of a route.
   (c) Find the probability that a lightpath request is blocked.
6. Consider an optical network with n nodes. Let $L_{i,j}$ be an arriving Poisson traffic rate on link i, j in packets per second, and let $1/\mu_{i,j}$ be the mean time of exponentially distributed packet transmission on link i, j.
   (a) Find the average queueing delay on link i, j.
   (b) Let $s_{i,j}$ be the number of source/destination pairs whose traffic is routed over link i, j. In this case, find the average queueing delay on link i, j.
   (c) Find the average queueing delay for a packet, given all source/destination pairs.

CHAPTER 15

Multicasting Techniques and Protocols

The capability of *multicasting* traffic has become a fundamental factor in the design of computer communications. Multicasting is the transmission of data from one source to a group of destinations. Data networks must be able to support such multimedia applications by multicasting data, voice, and video. Multicasting has led to the development of such applications as teleconferencing, multipoint data dissemination, educational distance learning, and Internet TV. In such applications, audio and video streaming technologies are needed to implement multicasting, owing to the real-time component of multimedia applications. In this chapter, we focus on several advanced multicast protocols used in the Internet. The following topics are covered:

- *Basic definitions and techniques*
- *Intradomain multicast protocols*
- *Interdomain multicast protocols*
- *Node-level multicast algorithms*

The chapter begins with some basic definitions and techniques: multicast group, multicast addresses, and muticast tree algorithms. Next, tree algorithms must be explored, as they are the next set of foundations for understanding Internet packet multicasting Two main classes of protocols are considered: *intra*domain multicast routing protocols, by which packets are multicast within a domain, and *inter*domain routing protocols,

by which packets multicast among domains. Techniques and algorithms used within router hardware are introduced at the end of the chapter.

15.1 Basic Definitions and Techniques

A simple operation of data multicasting can be viewed through a transmitting host and several receiving hosts. Instead of forcing the source host to send a separate packet to each destination host or user, the source needs to be able to forward a single packet to multiple addresses, and the network must be able to deliver a copy of that packet to each *group* of receiving hosts or users. Hosts can then choose to join or leave this group without synchronizing with other members.

A host may also belong to more than one group at a time. Figure 15.1 shows a typical packet multicasting in different layers. As shown, multicasting a packet could involve almost all layers of a communication network. A multicast task can be performed at the application layer, where two hosts can directly connect through a copy of the source. Meanwhile, a message can be multicast systematically through the physical, link, and network layers.

With the use of a robust multicasting protocol, a network reduces traffic load by simultaneously delivering a single stream of information to potentially thousands of recipients. An example is the distribution of real-time stock quotes and news. In such cases, the application source traffic is delivered to multiple receivers without burdening the source or the receivers and using a minimum amount of network bandwidth. Among the challenges of implementing data multicasting are the following two, which are especially relevant in large networks:

1. *Manageability*. As a data network becomes larger, constructing a central management system for distributed multicast tasks can become increasingly challenging.
2. *Scalability*. In practice, large networks that lack mechanisms for extending or adding equipment can experience substantial scalability problems, especially in constructing multicast trees.

Tight real-time components require a constant flow of data and have a very low jitter tolerance. To this end, corrupted data frames are often dropped rather than retransmitted. In this context, one would like to see a dynamic multicast system that adapts to deletions and additions of ongoing data over time. In the multicast service, we want to make copies of data at or near a source with only one access to the transport media. But having copies of packets floating around raises issues of

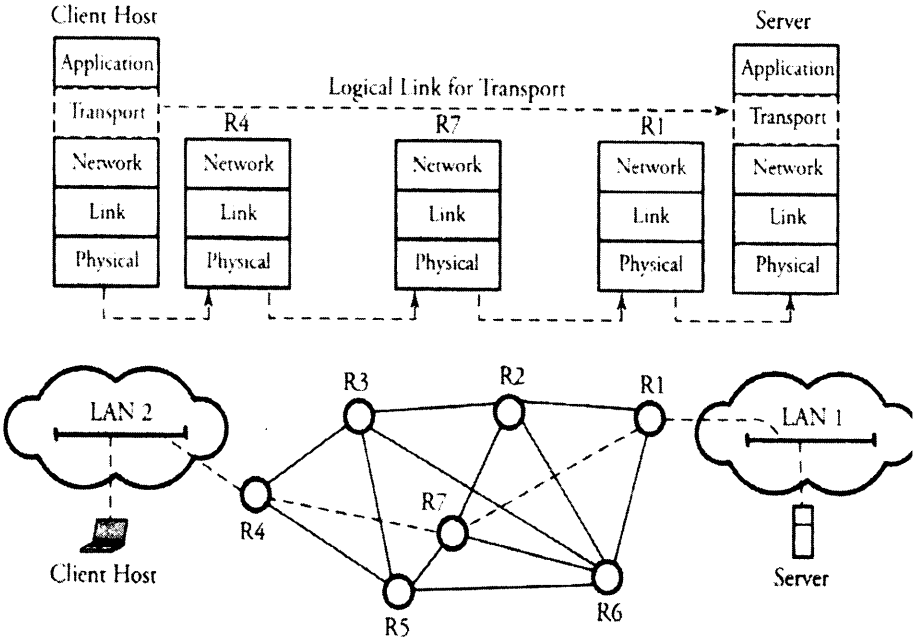

Figure 15.1 An IP multicast model

congestion. However, copying packets near destinations risks losing all copies of packets if the original packet is lost at that location.

15.1.1 IP Multicast Address

Chapter 2 discussed IP addressing, noting that multicast packets are of class D. The first four bits a class D IP address identify the class of the address, where 1110 represents a multicast address. The remaining 28 bits identify a particular multicast group. Thus, multicast addresses can range from 224.0.0.1 through 239.255.255.255, and 2^{28} is the maximum number of multicast groups that can be formed simultaneously.

Multicast addressing is an open issue at both the network and the link layers. As shown in Figure 15.2, a multicast address from the network layer needs to be mapped to multicast addresses of the data-link layer when mapping between IP multicast and an Ethernet multicast address is performed. Note that the space for the Ethernet multicast address is smaller than that for an IP multicast address. Therefore, the first 23 bits of the IP multicast address are placed into the first 23 bits of the Ethernet multicast address field. The 25 bits left in the Ethernet multicast address are assigned a fixed value. Note

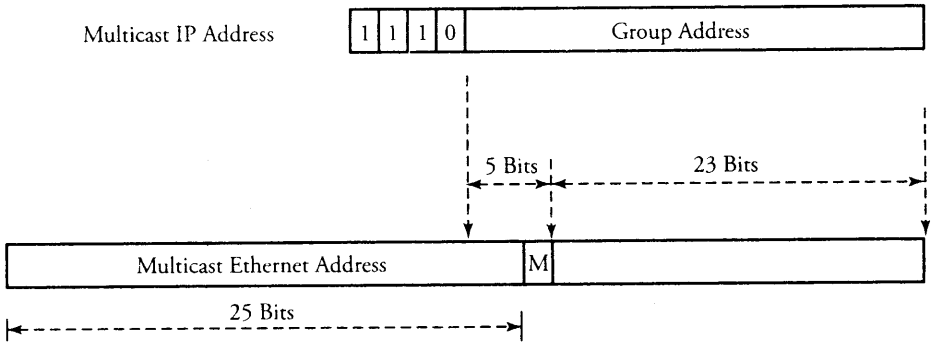

Figure 15.2 Mapping between IP multicast and Ethernet multicast addresses

also that a 1-bit field, *M*, represents the two cases: the Internet multicast with 0 and any other applications with 1.

15.1.2 Basic Multicast Tree Algorithms

Multicast group membership is a dynamic function, so any host can join or leave an IP multicast group at any time. Multicast packets are also subject to loss, delay, duplication, and out-of-order delivery. In WANs, data multicast transmission requires routers capable of managing multicast trees. A main difference between a regular unicast communication and a multicast communication is the nature of their routing. Unicast routing uses the shortest path, whereby two nodes communicate by means of a minimum-weight path.

The challenge in multicast routing is to identify the multicast group of nodes and to then find in the multicast group the minimum-weight tree that spans all members of the group. The multicast tree must also have low end-to-end delay, be scalable and survivable, and support dynamic membership. In a multicast transmission, a single copy of the packet is sent to a router with multicast capability, and the router makes copies and forwards them to all receivers. This is an efficient use of the bandwidth, as there is only one copy of a message over a link. The standard multicast model for IP networks expresses the way end systems send and receive multicast packets. This model is summarized as follows.

- A source typically uses the User Datagram Protocol (UDP), not TCP. As a result, multicast packets are delivered using a best-effort algorithm anytime, with no need to register or to schedule transmission.

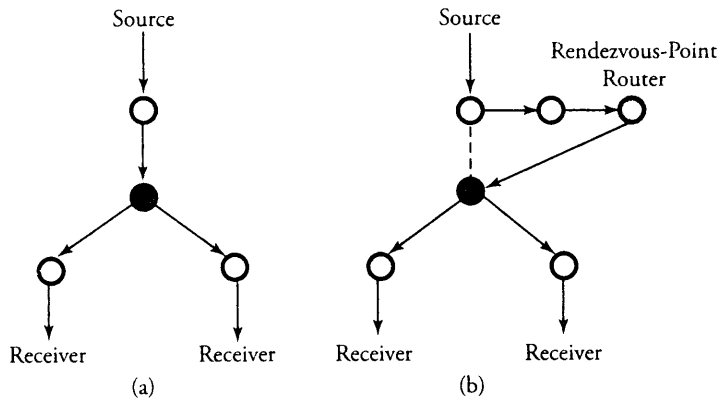

Figure 15.3 Two methods of constructing multicast algorithms: (a) dense mode, using a source-based tree; (b) sparse-mode, using a shared tree.

- Sources need to consider only the multicast address; they need not be a member of the multicast group to which they are sending.
- Multicast group members do not need to negotiate within the central group manager. Thus, they can join or leave a group at any time.

A *multicast tree algorithm* is the heart of multicast protocols and implements the fundamental task of packet copying and distribution in networks. Note that unicast routing uses the destination address to make its forwarding decision, whereas multicast routing normally uses the source address to make its forwarding decision. Tree algorithms form a triangle, or a tree, whereby the top vertex is represented by the source, and all recipients of the packet reside on the bottom line. During construction of a distribution tree, a *group* membership has to be formed, with copying nodes as members of the group. Updated information has to be maintained in a router if it is a member of the group. Two types of methods are used to construct a multicast tree: *dense-mode* trees and *sparse-mode* trees (see Figure 15.3).

Dense-Mode Algorithm

With *dense-mode*, or *broadcast-and-prune*, algorithm, a multicast tree technique called *source-based tree* is used. This technique requires the determination of a shortest-path tree to all destinations and uses a reverse shortest-path tree rooted at a source. The tree starts at the source; for every source, there is always a corresponding shortest-path tree. As a result, the spanning tree guarantees the lowest cost from a source to all leaves of

the tree. Packets are forwarded on the shortest-path tree according to: (1) the source address they originated from, and (2) the group address they are addressed to. The source-based tree in a network provides the shortest distance and the least delay from a source to all receivers.

Sparse-Mode Algorithm

The *sparse-mode algorithm* uses a *shared-tree* technique. Relatively short distances are selected, and no effort is made to find the best paths to destinations. First, some shared trees are formed. These trees consider multiple senders and receivers and use some nodes acting as *rendezvous points* to connect sources and receivers. A rendezvous point—known as the *core*, or *root*—is considered a point in the network where roots of distribution subtrees are shared and the multicast data flows down to reach the receivers in the network. The sparse-mode algorithm uses a rendezvous router to coordinate forwarding packets from source to receivers and to prevent the initial flooding of datagrams. However, this may introduce extra delay from the source to all receivers but requires less hardware complexity for finding the shortest path. All routers in the network must discover the information about network routers that become rendezvous routers and the mappings of multicast groups to rendezvous routers.

The sparse mode has a couple of drawbacks. One is that a rendezvous router can be a hot-spot for multicast traffic and a point of failure in routing. Also, forwarding traffic to the rendezvous router and then to receivers causes delay and longer paths.

15.1.3 Classification of Multicast Protocols

The fundamental algorithms described in Section 15.1.2 are the basic procedures for constructing networking multicast protocols. These algorithms are used as foundations of protocols for multicasting packets in the Internet. Two main classes of multicast protocols are described in Section 15.2 and 15.3, respectively: *intra*domain multicast routing protocols, by which packets are multicast within a domain, and *inter*domain routing protocol, by which packets are multicast among domains.

15.2 Intradomain Multicast Protocols

*Intra*domain routing protocols carry out the multicast function within domains. The implementation of multicast routing faces the following particular challenges:

- Dynamic change in the group membership
- Minimizing network load and avoiding routing loops
- Finding concentration points of traffic

In practice, several protocols play major roles in establishing multicast connections. The *Distance Vector Multicast Routing Protocol* (DVMRP) and the *Internet Group Management Protocol* (IGMP) are the two original protocols forming the early version of the *multicast backbone* (MBone). Other protocols, such as *Multicast Open Shortest Path First* (MOSPF), *core-based trees* (CBT), and *protocol-independent multicast* (PIM) enhance MBone performance.

15.2.1 Distance Vector Multicast Routing Protocol (DVMRP)

The Distance Vector Multicast Routing Protocol (DVMRP) is one of the oldest multicast protocols. It is based on a concept of exchanging routing table information among directly connected neighboring routers. The MBone topology can enable multiple tunnels to run over a common physical link. Each participating router maintains information about all the destinations within the system. DVMRP creates multicast trees, using the dense-mode algorithm. A multicast router typically implements several other independent routing protocols besides DVMRP for multicast routing, such as RIP or OSPF for unicast routing.

This protocol is not designed for WANs with a large number of nodes, since it can support only a limited number of hops for a packet. This is clearly considered a drawback of this protocol, as it causes packet discarding if a packet does not reach its destination within the maximum number of hops set. Another constraint in this protocol is the periodic expansion of a multicast tree, leading to periodic broadcasting of multicast data. This constraint in turn causes the issue of periodic broadcast of routing tables, which consumes a large available bandwidth. DVMRP supports only the source-based multicast tree. Thus, this protocol is appropriate for multicasting data among a limited number of distributed receivers located close to the source.

15.2.2 Internet Group Management Protocol (IGMP)

The *Internet Group Management Protocol* (IGMP) is used for TCP/IP between a receiver and its immediate multicast-enabled routers reporting multicast group information. This protocol has several versions and is required on all machines that receive IP multicast. As the name suggests, IGMP is a group-oriented management protocol that provides a dynamic service to registered individual hosts in a multicast group on a particular network.

When it sends an IGMP message to its local multicast router, a network host it in fact identifies the group membership. Routers are usually sensitive to these types of messages and periodically send out queries to discover which subnet groups are active. When a host wants to join a group, the group's multicast address receives an IGMP

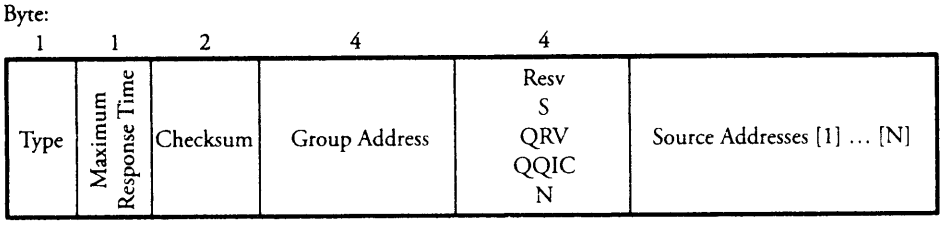

Figure 15.4 IGPM packet format

message stating the group membership. The local multicast router receives this message and constructs all routes by propagating the group membership information to other multicast routers throughout the network.

The IGMP packet format has several versions; Figure 15.4 shows version 3. The first 8 bits indicate the message *type*: which may be one of *membership query*, *membership report* v_1, *membership report* v_2, *leave group*, and *membership report* v_3. Hosts send IGMP membership reports corresponding to a particular multicast group, expressing an interest in joining that group.

IGMP is compatible with TCP/IP, so the TCP/IP stack running on a host forwards the IGMP membership report when an application opens a multicast socket. A router periodically transmits an IGMP membership query to verify that at least one host on the subnet is still interested in receiving traffic directed to that group. In this case, if no response to three consecutive IGMP membership queries is received, the router times out the group and stops forwarding traffic directed toward that group. (Note that v_i refers to the version of the protocol the membership report belongs to.)

IGMP version 3 supports *include* and *exclude* modes. In *include* mode, a receiver announces the membership to a host group and provides a list of source addresses from which it wants to receive traffic. With *exclude mode*, a receiver expresses the membership to a multicast group and provides a list of source addresses from which it does not want to receive traffic. With the *leave group* message, hosts can report to the local multicast router that they intend to leave the group. If any remaining hosts are interested in receiving the traffic, the router transmits a group-specific query. In this case, if the router receives no response, the router times out the group and stops forwarding the traffic.

The next 8 bits, *max response time*, are used to indicate the time before sending a response report (default, 10 seconds). The *Resv* field is set to 0 and is reserved for future development. The next field is *S* flag, to suppress router-side processing. *QRV*

indicates the querier's robustness variable. *QQIC* is the querier's query interval code. N shows the number of sources, and *Source Address* $[i]$ provides a vector of N individual IP addresses.

15.2.3 Multicast OSPF (MOSPF) Protocol

The *Multicast Open Shortest Path First* (MOSPF) protocol, an extension to the unicast model of OSPF discussed in Chapter 7, constructs a link-state database with an advertisement mechanism. Let's explore what new features a link-state router requires to become capable of multicast functions.

Link-State Multicast

As explained in Chapter 7, *link-state routing* occurs when a node in a network has to obtain the state of its connected links and then send an update to all the other routers once the state changes. On receipt of the routing information, each router reconfigures the entire topology of the network. The link-state routing algorithm uses Dijkstra's algorithm to compute the least-cost path.

The multicast feature can be added to a router using the link-state routing algorithm by placing the spanning tree root at the router. The router uses the tree to identify the best next node. To include the capability of multicast support, the link-state router needs the set of groups that have members on a particular link added to the *state* for that link. Therefore, each LAN attached to the network must have its host periodically announce all groups it belongs to. This way, a router simply detects such announcements from LANs and updates its routing table.

Details of MOSPF

With OSPF, a router uses the flooding technique to send a packet to all routers within the same hierarchical area and to all other network routers. This simply allows all MOSPF routers in an area to have the same view of group membership. Once a link-state table is created, the router computes the shortest path to each multicast member by using Dijkstra's algorithm. This protocol for a domain with n LANs is summarized as follows.

Begin MOSPF Protocol

1. **Define:** N_i = LAN i in the domain, $i \in \{1, 2, \cdots, n\}$
   $G_j(i)$ = Multicast group j constructed with connections to N_i
   R_i = Router attached to N_i
2. **Multicast groups:** R_i maintains all N_j group memberships.

3. **Update:** Use the link-state multicast whereby each router R_i floods all its multicast group numbers, $G_j(i)$, to all its directly attached routers.
4. Using the shortest-path tree algorithm, each router constructs a least-cost tree for all destination groups.
5. When it arrives at a router, a multicast packet finds the right tree, makes the necessary number of copies of the packet, and routes the copies. ■

MOSPF adds a link-state field, mainly membership information about the group of LANs needing to receive the multicast packet. MOSPF also uses Dijkstra's algorithm and calculates a shortest-path multicast tree. MOSPF does not flood multicast traffic everywhere to create state. Dijkstra's algorithm must be rerun when group membership changes. MOSPF does not support sparse-mode tree (shared-tree) algorithm. Each OSPF router builds the unicast routing topology, and each MOSPF router can build the shortest-path tree for each source and group. Group-membership reports are broadcast throughout the OSPF area. MOSPF is a dense-mode protocol, and the membership information is broadcast to all MOSPF routers. Note that frequent broadcasting of membership information degrades network performance.

Example. In Figure 15.5, each of the five LANs in a certain domain has an associated router. Show an example of MOSPF for multicasting from router R_1 to seven servers located in LANs 1, 2, and 3.

Solution. Multicast groups 1 and 2 are formed. For group 1, the best tree is implemented using copying root R_4. For group 2, the best tree is implemented using copying root R_7.

15.2.4 Protocol-Independent Multicast (PIM)

Protocol-independent multicast (PIM) is an excellent multicast protocol for networks, regardless of size and membership density. PIM is "independent" because it implements multicasting independently of any routing protocol entering into the multicast routing information database. PIM can operate in both *dense mode* and *sparse mode*. Dense-mode is a flood-and-prune protocol and is best suited for networks densely populated by receivers and with enough bandwidth. This version of PIM is comparable to DVMRP.

Sparse-mode PIM typically offers better stability because of the way it establishes trees. This version assumes that systems can be located far away from one another and that group members are "sparsely" distributed across the system. This protocol for a domain with n LANs is summarized as follows.

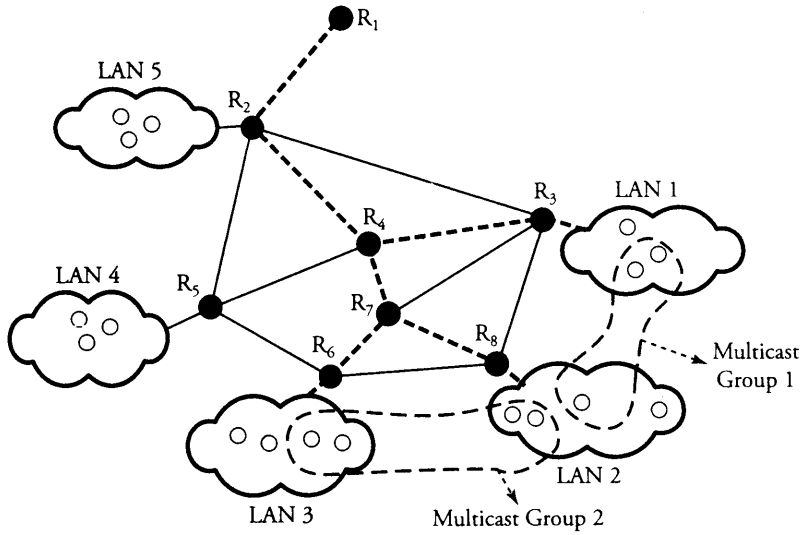

Figure 15.5 Use of MOSPF to multicast from router R_1 to seven servers located in two different multicast groups spread over three LANs

Begin PIM Algorithm

1. **Define:** N_i = LAN i in the domain $i \in \{1, 2, \cdots, n\}$
   $G_j(i)$ = Multicast group j constructed with connections to N_i
   R_i = The router attached to N_i
2. **Multicast groups:** R_i maintains all N_j group memberships.
3. **Rendezvous router:** Using the sparse-mode tree algorithm, each group $G_j(i)$ chooses a rendezvous router.
4. **Update:** Each Router R_i updates its rendezvous router whenever there is a change in the group.
5. When it arrives at a rendezvous router, a multicast packet finds the right tree and routes the packet. ■

Sparse-mode PIM first creates a shared tree, followed by one or more source-specific trees if there is enough traffic demand. Routers join and leave the multicast group *join* and *prune* by protocol messages. These messages are sent to a rendezvous router allocated to each multicast group. The rendezvous point is chosen by all the routers in a group and is treated as a meeting place for sources and receivers. Receivers join the

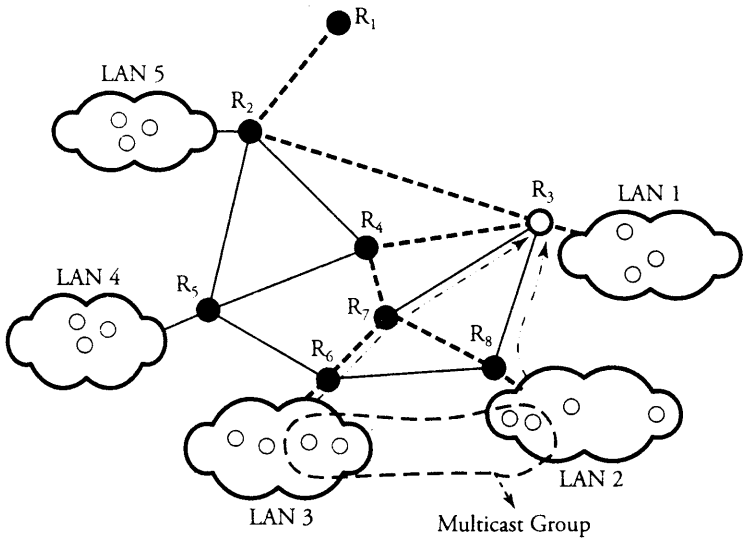

Figure 15.6 Sparse-mode PIM for multicasting from router R_1 to four servers in a multicast group spread over LANs 2 and 3

shared tree, and sources register with that rendezvous router. Note that the shortest-path algorithm made by the unicast routing is used in this protocol for the construction of trees.

Each source registers with its rendezvous router and sends a single copy of multicast data through it to all registered receivers. Group members register to receive data through the rendezvous router. This registration process triggers the shortest-path tree between the source and the rendezvous router. Each source transmits multicast data packets encapsulated in unicast packets to the rendezvous router. If receivers have joined the group in the rendezvous router, the encapsulation is stripped off the packet, and it is sent on the shared tree. This kind of multicast forwarding state between the source and the rendezvous router enables the rendezvous router to receive the multicast traffic from the source and to avoid the overhead of encapsulation.

Example. In Figure 15.6, the five LANs in a certain domain each have an associated router. Show an example of sparse-mode PIM for multicasting from router R_1 to four servers located in a multicast group spread over LANs 2 and 3.

Solution. Multicasting is formed with R_3 the associated rendezvous router for this group. Thus, the group uses a reverse unicast path as shown to update the rendezvous

router of any joins and leaves. For this group, the multicast tree is implemented using copying root R_7.

15.2.5 Core-Based Trees (CBT) Protocol

In sparse mode, forwarding traffic to the rendezvous router and then to receivers causes delay and longer paths. This issue can be partially solved in the *core-based tree* (CBT) protocol, which uses bidirectional trees. Sparse-mode PIM is comparable to CBT but with two differences. First, CBT uses bidirectional shared trees, whereas sparse-mode PIM uses unidirectional shared trees. Clearly, bidirectional shared trees are more efficient when packets move from a source to the root of the multicast tree; as a result, packets can be sent up and down in the tree. Second, CBT uses only a shared tree and does not use shortest-path trees.

CBT is comparable to DVMRP in WANs and uses the basic sparse-mode paradigm to create a single shared tree used by all sources. As with the shared-trees algorithm, the tree must be rooted at a *core* point. All sources send their data to the core point, and all receivers send explicit join messages to the core. In contrast, DVMRP is costly, since it broadcasts packets, causing each participating router to become overwhelmed and thereby requiring keeping track of every source-group pair. CBT's bidirectional routing takes into account the current group membership when it is being established.

When broadcasting occurs, it results in traffic concentration on a single link, since CBT has a single delivery tree for each group. Although it is designed for intradomain multicast routing, this protocol is appropriate for interdomain applications as well. However, it can suffer from latency issues, as in most cases, the traffic must flow through the core router. This type of router is a bottleneck if its location is not carefully selected, especially when transmitters and receivers are far apart.

15.2.6 Multicast Backbone (MBone)

The first milestone in the creation of a practical multicast platform was the development of the *multicast backbone* (MBone), which carried its first worldwide event when several sites received audio simultaneously. The multicast routing function was implemented using unicast-encapsulated multicast packets. The connectivity among certain receivers was provided using point-to-point IP-encapsulated *tunnels*. Figure 15.7 shows an example of tunneling among routers in the early version of MBone. Each tunnel connects two end points via one logical link and crosses several routers. In this scenario, once a packet is received, it can be sent to other tunnel end points or broadcast to local members. The routing in earlier version of MBone was based on DVMRP and IGMP.

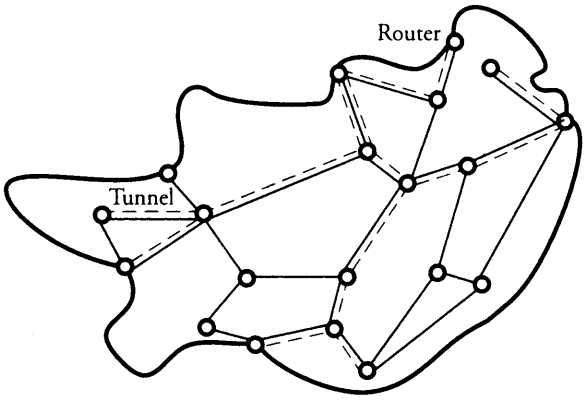

Figure 15.7 Tunneling in the multicast backbone

15.3 Interdomain Multicast Protocols

Interdomain multicast protocols are designed for hierarchical and Internet-wide multicast purposes. Within a domain, a network manager can implement any routing protocol desired. The challenge in interdomain multicast administration is choosing the best external link to route to hosts in an external domain. Among the protocols in this category are *multiprotocol Border Gateway Protocol* (MBGP), *Multicast Source Discovery Protocol* (MSDP), and *Border Gateway Multicast Protocol* (BGMP).

15.3.1 Multiprotocol BGP (MBGP)

Multiprotocol BGP (MBGP) is an extension to its unicast version, Border Gateway Protocol (BGP). In Figure 15.8, three domains are connected through two types of paths: a unicast path handled by BGP and a multicast path handled by MBGP. With BGP, the multicast routing hierarchy operates the same way as multiple unicast routing does. Between each two domains, two corresponding border routers compute the set of domain parameters that should be traversed to reach any network. Typically, the parameters in one domain are not known or trusted by the others.

Each domain-management system advertises the set of routes that can reach a particular destination. Packets are routed on a hop-by-hop basis. Thus, MBGP can determine the next hop to a host but cannot provide multicast tree-construction functions. Sources within a domain register with a rendezvous router, and receivers send joins to this router. This join message is required to find the best reverse path to the source.

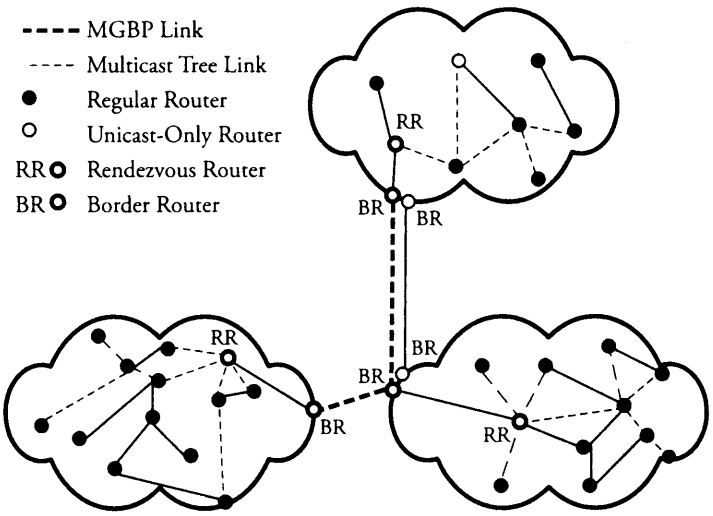

Figure 15.8 MBGP: interdomain multicast routing

This protocol can handle multiprotocol paths, and each participating router needs to know only the topology of its own domain and the paths to reach each of the other domains. Since MBGP is an interdomain protocol, a class D address is not carried in an MBGP message, and it does not carry information about multicast groups. Because the time and frequency of join messages transmission are not determined in this protocol, and multicast trees in various domains are not connected, other multicast protocol choices are more attractive.

15.3.2 Multicast Source Discovery Protocol (MSDP)

One of the main issues in interdomain multicasting is how to inform a rendezvous router in one domain while there are sources in other domains. In other words, rendezvous routers of two adjacent domains are not able to communicate with each other when one receives a source register message. In practice, only one rendezvous point is in each domain. This issue becomes more critical when group members should be spread over multiple domains. In this case, multicast trees among domains are not connected. As a result, the traffic can reach all receivers of a domain, but any sources outside the domain stay disjoint.

The *Multicast Source Discovery Protocol* (MSDP) has potential solutions to these issues. Figure 15.9 shows how this protocol operates. A unique feature of this protocol is

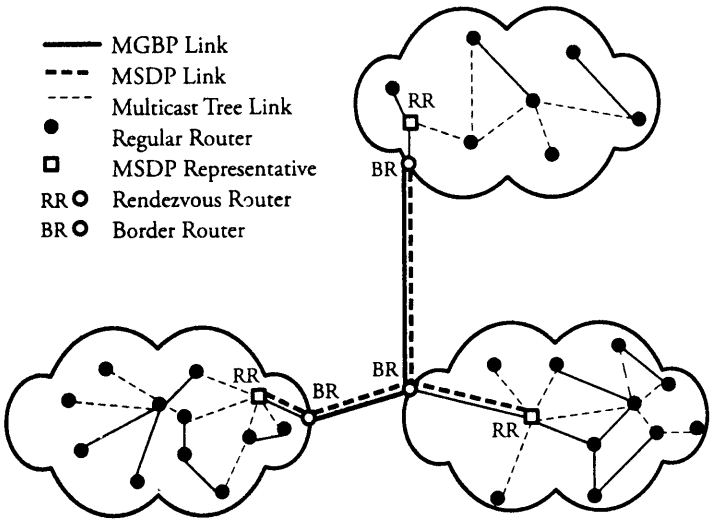

Figure 15.9 MSDP: interdomain multicast routing

that it assigns representatives in each domain. A representative reports to other domains the existence of active sources. A new source for a group must first register with the domain's rendezvous router.

Each MSDP representative in the domain detects the existence of the new source and sends a message to all MSDP representatives, as shown. The representatives check whether the broadcasting representative has sent the message through the correct path to prevent possible message looping. Once this process is complete and the message is on the correct router interface, the message is forwarded to all remaining associated representatives. An MSDP representative and the rendezvous router within a domain can be the same point; this is the checkpoint where the state of group membership is checked. If it has the correct state, the router sends a join message to the source address mentioned in the message. Here, we assume that the intradomain multicast is performed using PIM, whereby a join message is generated and processed. The message is then forwarded on the multicast tree by the rendezvous router; once all group members receive data attached to the message, they may use a shortest-path tree, using sparse-mode PIM. This process ends when all the representatives finish the process.

This multicast procedure uses a combination of three protocols: sparse-mode PIM, MBGP, and MSDP. Although this multicast procedure has been well accepted as a

practical multicast method, its complexity results in a timing issue. One aspect is scalability. Also, when sources are bursty or group member join and leave events frequently, the overhead of managing the group can be noticeable. This fact in turn creates the timeout issue, whereby the period of silence between packet bursts becomes greater than the forwarding-state timeout. MSDP solves this issue by selecting and processing every n packets in burst. However, this trick is not quite a practical solution when a fairly large network is under multicasting.

15.3.3 Border Gateway Multicast Protocol (BGMP)

The *Border Gateway Multicast Protocol* (BGMP) is based on the construction of bidirectional shared trees among domains using a single root. Finding the best domain to place the root of such shared trees is a challenge, but several solutions are available. One of the methods of address resolution is the *Multicast Address-Set Claim* (MASC) protocol, which guarantees the immediate resolution of address collisions.

Another method of resolving this issue is to use *root-addressed multicast architecture* (RAMA) by which a source is selected as the root of the tree. This way, the complexity of root placement in other multicast routing protocols can be eliminated. RAMA is of two types. The first type is *express multicast*: The root of the tree is placed at the source, and group members send join messages on the reverse path back to the source. This protocol is aimed at systems that use logical channels, such as single-source multimedia applications, TV broadcast, and file distribution. The second type is *simple multicast*: Multiple sources per group are allowed. In this case, one source must be selected, and the root of the tree is put at this node as a primary and first-hop router. Then, all receivers send join messages to the source. The next step is to construct a bidirectional tree through which additional sources send packets to the root. Since a bidirectional tree is constructed, packets arrive at a router in the tree and are forwarded both downstream to receivers and upstream to the root.

15.4 Node-Level Multicast Algorithms

Multicasting techniques can also be used at the router level. To implement a multicast connection, a binary tree is normally constructed with the source switch port at the root and the destination switch ports at the leaves. Internal nodes act as relay points that receive packets and make copies. A number of such multicast methods are used. One is a *tree-based multicast algorithm* using a separate copy network. The *Boolean*

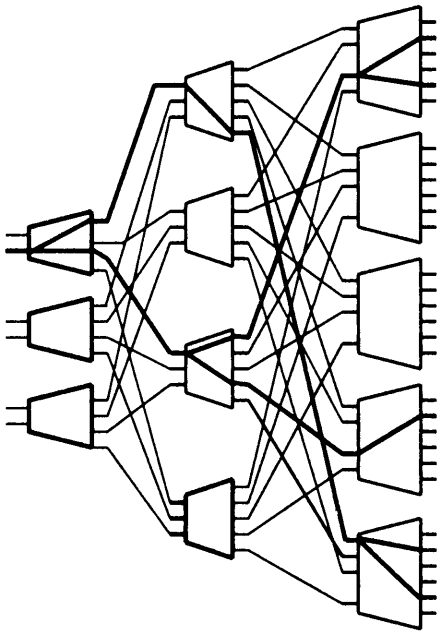

Figure 15.10 Tree-based multicasting in an expansion switch

splitting multicast algorithm is used for multistage switches. The third technique the *packet recirculation multicast algorithm*. The fourth is multicasting in *three-dimensional switches*.

15.4.1 Tree-Based Multicast Algorithm

Imagine that a multicast tree algorithm must be applied on a multistage switch fabric, as shown in the expansion switch in Figure 15.10. Multicasting is implemented as a tree structure. A source generates a packet and sends it out to the first crossbar switch. The packet may have a field that specifies how many copies of this packet are to be made. All copies may be destined for other crossbars, in which more copies of the packet are made, and some packets may move to their destinations.

For implementing a *tree-based multicast algorithm*, the copying tasks in multistage switch fabrics typically involve a copy switch and a routing switch. An example of such systems consists of two Beneš networks for constructing the copying and routing networks. The following algorithm presents the multicast steps within a copy network with n inputs or outputs constructed with $d \times d$ switch element:

Define:

$k = \log_d n$

r = Number of stages

j = Stage number, $j \in \{1, 2, ..., r\}$

F = Global number of copies, $F \in \{1, 2, \ldots, n\}$

F_j =Number of copies remaining before the packet arrives at stage j

f_j = Local number of copies made at stage j

Begin Tree-Based Multicast Algorithm

For Stage $1 \le j \le r - k \Rightarrow$

$F_j = F$

$f_j = 1$

For Stages $(r-k)+1 \le j \le r \Rightarrow$

$F_j = \left[\frac{F_{j-1}}{f_{j-1}}\right]$

$f_j = \left[\frac{F_j}{d^{r-j}}\right]$

■

In the copy network shown in Figure 15.11, packets are replicated as designated by the initial *global number of copies*, F, given in the routing-control field of the packet header. The global number of copies refers to the total number of packet copies requested. Let F_j be the remaining number of copies of a packet when it arrives at stage j, and let f_j be the number of copies made locally at stage j. First, the algorithm initialize F_j to F and f_j to 1. The copying method is such that the replication of the packets takes place stage by stage within the network in order to distribute the traffic as evenly as possible. The routing is either point to point or multipoint connection.

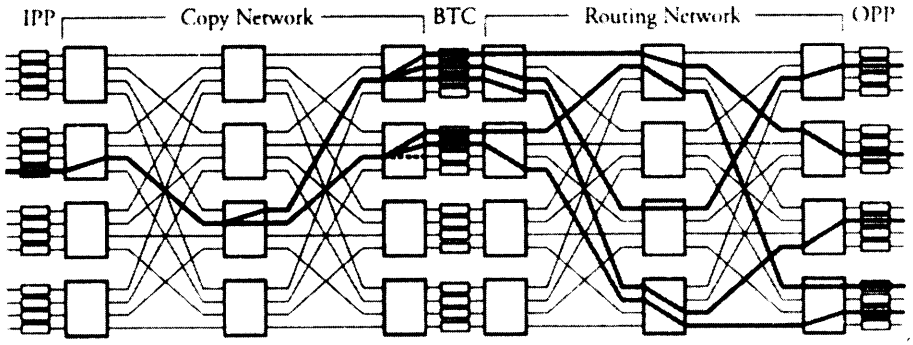

Figure 15.11 Tree-based multicasting through a three-stage copy network followed by a routing network

Consider a copy network with k stages. When multipoint connections are requested, a packet is routed randomly in the first $k - 1$ stages. In the last k stages, the packet is copied. The operation $f_{j-1} = \left\lceil \frac{F_j}{f_{k-j}} \right\rceil$ computers a new number of copies for a packet a stage j, with the local number of copies $F_{j-1} = \left\lfloor \frac{F_j}{d^{r-j}} \right\rfloor$. The algorithm is set up to generate the final number of copies that appear on the output to be the smallest multiple of 2 greater than or equal to F.

This technique reduces the hardware complexity in the controller. If the final-number of copies that appear on the outputs is more than requested, the unneccessary packet copies can be easily thrown away. This task is completed by broadcast translated circuit (BTC), shown in Figure 15.11. BTCs receive all the copies of packet. A central monitoring mechanism for all BTCs compares the number of requested copies read from the packet's header and the number of copies made in the copy network. This mechanism then decides to remove unused copies, based on this comparision. For the routing network, routing is similar to that in a point-to-point copy network.

Example. Find all numbers of copies in every stage of a $B_{16,4}$ switching network in which the global number of copies of an incoming packet is $F = 5$ and copies of the packet are to be routed to output port numbers 1, 6, 9, 12, and 14.

Solution. The copy network has $r = \log_d n - 1 = 3$ stages and $k = 2 \log_d n$. For stages $1 \le j \le r - k = 1$ or at stage $j = 1$, we have $F_1 = 5$ and the local number of copies $f_1 = 1$, so the packet gets distributed. For stages $(r - k) + 1 = 2 \le j \le r = 3$, starting at stage $j = 2$, $F_2 = [F_1/f_1] = 5$, and $f_2 = [F_2/d^{r-j}] = 2$; thus, two copies are made at this stage and guided to two switches at the last stage ($j = 3$). At the third stage, $F_3 = [F_2/f_2] = 3$, and the local number of copies $f_3 = [F_3/d^{r-j}] = 3$.Therefore, for each of these switches, three copies of the packet are made. Note that the sum of these copies (6) has exceeded the requested global number of copies (5), so the one additional copy is thrown away by the corresponding BTC (see Figure15.11).

15.4.2 Boolean Splitting Multicast Algorithm

The *Boolean splitting multicast algorithm* is a method of copying packets in any space-division switch fabric with n inputs or outputs constructed with 2×2 switch elements and therefore with $\log_2 n$ stages. Consider a Banyan network in which switching nodes replicate packets based on 2-bit header information. The following algorithm presents multicast steps within a copy network.

Define:

$k = \log_d n$

r = Number of stages

j = Stage number , $j \in \{1, 2, \ldots, r\}$

f_j = Local number of copies made at stage j

$a(j)$: = Lowest requested output address at stage j: $a_k a_{k-1} \ldots a_1$

$A(j)$ = Highest requested output address at stage $_j$: $A_k A_{k-1} \ldots A_1$

Begin Boolean Splitting Multicast Algorithm

For stages $1 \le j \le r - k \Rightarrow$

$F_j = F$

$f_j = 1$

For stages $(r - k) + 1 \le j \le r \Rightarrow$

Sort the output addresses

Compare a_j with A_j

If $a_j = A_j = 0 \Rightarrow$

Send the packet on crossbar output 0 (link 0).

For the packet forwarded to link 0 $\Rightarrow$

$a_{j+1} = a_j$

$A_{j+1} = A_k \ldots A_{k-(j-1)} 01 \ldots 1$

If $a_j = A_j = 1 \Rightarrow$

Send the packet on crossbar output 1 (link 1),

For the packet forwarded to link 1 $\Rightarrow$

$a_{j+1} = a_k \ldots a_{k-(j-1)}\, 10 \ldots 0$

$A_{j+1} = A_j$

If $a_j = 0$ and $A_j = 1$, or $a_j = 1$ and $A_j = 0$, $\Rightarrow$

Replicate the packet.

Send one copy of the packet carrying upper half of addresses on link 0.

Send one copy of the packet carrying lower half of addresses on link 1.

■

The switch fabric replicates packets according to a Boolean interval-splitting algorithm, based on the address interval in the new header. An output port address of the switch fabric that receives a copy of the packet is represented by two k-bit binary numbers. Suppose that a packet arrives at an arbitrary crossbar at stage $j - 1$ from a previous crossbar j. If we represent the minimum number by a k-bit address $a_k \ldots a_1$ and the maximum number by a k-bit address $A_k \ldots A_1$, we can make the following observations: If $a_j = A_j = 0$ or $a_j = A_j = 1$, the packet is sent out on crossbar output

0 or 1, respectively. If $a_j = 0$ and $A_j = 1$, the network replicates the packet and sends both packets out on both links 0 and 1.

If replication takes place, the packet header needs to be modified by splitting the original address interval into two subintervals, which can be expressed by the two following main recursions: $a_{j-1} = a_j$ and $A_{j-1} = A_k \ldots A_{(j-1)}01\ldots1$ for the packet sent out on Link 0, and $a_{j+1} = a_k \ldots a_{k-1}10 \ldots 0$, and $A_{j+1} = A_j$ for the packet sent out on link 1. This modification is a routine operation, meaning that it is independent of the header contents and can be considered built-in hardware.

Example. Show the detail of the Boolean splitting multicast algorithm applied to copy a packet in a Banyan network $Y_{16,2}$. Find all numbers of copies in every stage of a switcing fabric, where the global number of copies of an incoming packet is $F = 6$, and copies of the packet are routed to output port numbers 5, 6, 7, 8, 9, and 10.

Solution. The switch fabric has $k = [\log_2 16] = 4$ stages. At stage $j = 1$, we have $F_1 = 6$ $a(1) = 0101$ and $A(1) = 1010$; consequently, $a_1 = 0$ is compared with $A_1 = 1$. Since these two bits are different, 0101 to 0111 are assigned to link 0, and 1000 to 1010 are assigned to link 1. If we continue the multicast process, the results of this example will be as shown in Figure 15.12.

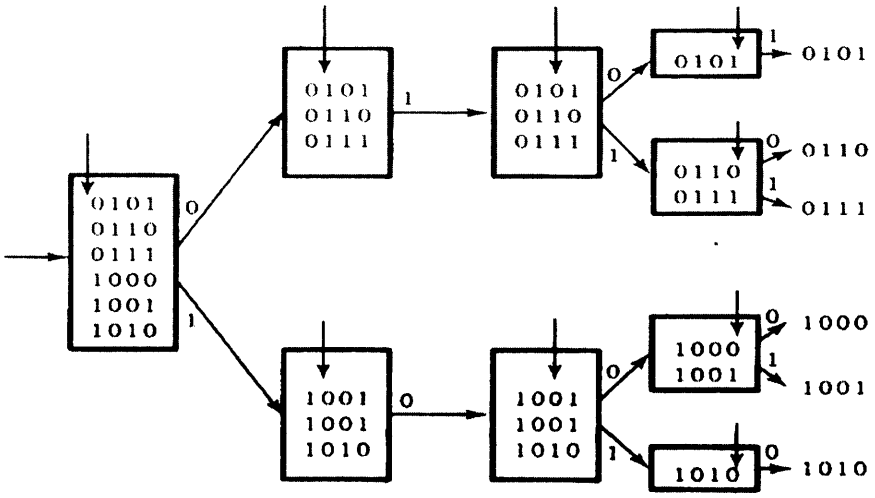

Figure 15.12 The Boolean splitting multicast algorithm in a multistage switch

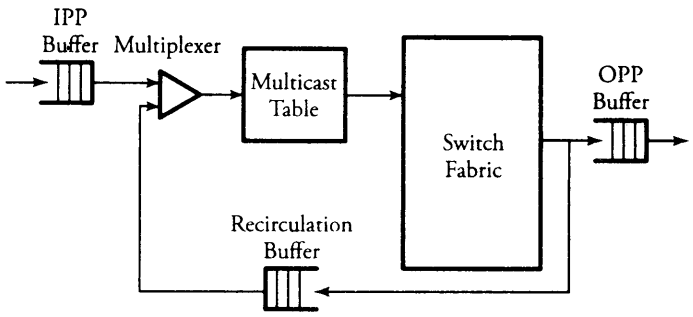

Figure 15.13 Recirculation multicasting

15.4.3 Packet Recirculation Multicast Algorithm

Packet recirculation (*recycling*) is another feasible method for constructing large switching fabrics for broadband switching applications. To implement a multicast connection, a binary tree is constructed with the source port at its root and the destination switch port at its leaves. This technique can be used for almost all kinds of space-division switch fabrics.

If a switch fabric is constructed by a multistage interconnected network, internal crossbars can act as relay points to accept packets and recycle them back into the switch fabric after packet relabeling. New labels carry the information about the destination pair, identifying the next two switch ports to which they are to be sent. Figure 15.13 illustrates the recirculation technique. A *multicast table* provides an output port/IP address pair. This pair is added to the packet header; thus, two additional bits indicate whether the pair is to be recirculated again. An *IPP buffer* holds packets received from the input link and forwards them to the switching network. An *OPP buffer* holds packets waiting to be transmitted on outgoing links. A *recirculation* buffer provides buffering to the packets that need to be recycled.

Packet recirculation and copying within the switch fabric is straightforward. Figure 15.14 illustrates packet recirculation and copying. Suppose that a is a source that generates packets to be delivered to output ports b, c, d, and e and that x and y are relay points. Suppose that a packet entering at input port a must have several copies. For such a packet, there may be entries of the type (x, j) and (e, k) in the multicast table. This can be interpreted as making two copies of the packet: one to be *recirculated* back to port x on address j; the other one, to port e with address k. The multicast table entry at x may now have entries (b, n) and (y, m), as seen. Typically, each packet header has 2 bits to indicate whether a packet needs to be recirculated.

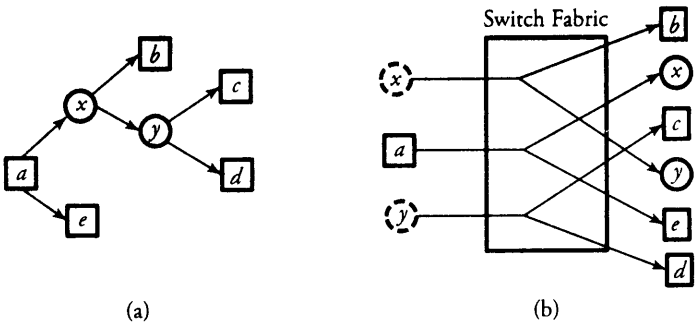

Figure 15.14 Within the switch fabric of routers: (a) packet recirculation and (b) copying

In summary, an end point that is added needs to have a parent. A good candidate is the one that does not have heavy recirculation traffic. Like the parent of the new end point, the intermediate point also is new, so it has to be put into the multicast table entry of another node, which becomes the grandparent of the new end point, whose child will now become a sibling of the new end point.

To remove a connection, we can consider two cases. We should examine whether the output to be removed has no grandparent but its sibling has children; in that case, we replace the parent's multicast table entry with the sibling's children. If the output to be removed has no grandparent and its sibling has no children, we simply drop the output to be removed from its parent's multicast table entry, and the connection reverts to a simple point-to-point connection. Note that since packets must be resequenced when they exit from the system, the resequencing buffer at the output port processor (OPP) of a router must be dimensioned to delay packets long enough so that slow packets have a chance to catch up with fast packets.

15.4.4 Multicasting in Three-Dimensional Switches

This switch uses a Clos network as a building block module. An n-port Clos network is a nonblocking network if $k \geq 2d - 1$, where d and k are the sizes of the first-stage crossbar switch elements. The three-dimensional structure of the Clos network consists of m parallel planes of the same type and same size Clos network in Figure 15.15. An incoming packet can be demultiplexed among m planes. The demultiplexer does the work of distributing the input packets. In other words, the three-dimensional switching system accepts packets coming from different planes and time multiplexes them onto the same output port. The multicast algorithm for each plane is described as follows.

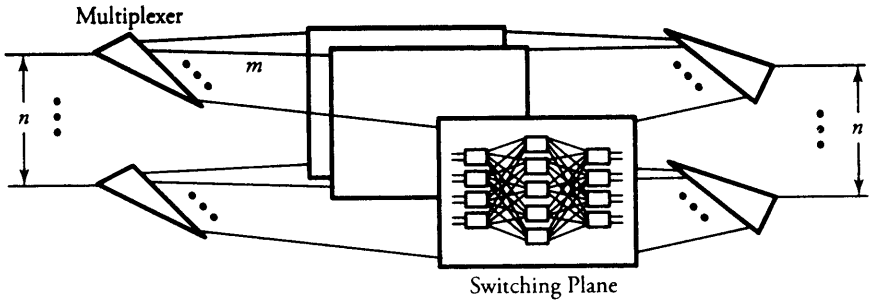

Figure 15.15 Three-dimensional multiplexed Clos network

At the first stage, no routing decision needs to be made. The first-stage switch elements maintain a table that indicates utilization of the center-stage arrays. This table is created on the basis of back-pressure information obtained from the center-stage arrays. A center-stage element least used at any particular time gets the highest priority. In fact, this table can be shared by all the first-stage arrays. Once a multicast packet arrives at the center-stage array, the switch looks at the first g bits of every address, where $g = \log_2 n/d$. The packet is then split according to the number of distinct k bits found. The maximum number of new packets that can be created is n/d. All addresses with the same first g bits are sent to the corresponding switch element in the last stage. Also, since four distinct addresses are identified by the first 2 bits in our example, the packet is split four ways.

The final stage (stage 3) looks at the remaining l bits, where $l = \log_2 n$, in each address of the multicast packet received by the last-stage array. The packet is again split according to the number of different sets of l bits found. The maximum number of packets at this stage is l. The packets are then sent to the appropriate output port corresponding to the last l bits. If a packet is split, the header has to be modified. The new header contains the addresses with the same initial g bits.

A multicast packet on a particular input port requesting F copies of a packet can distribute the request equally between the parallel planes. This reduces the load on any particular plane, leading to a reduction in the use of buffers. An analysis of a three-dimensional Clos network is based on the blocking probability $P_b(i)$ for input number i out of n:

$$P_b(i) = \sum_{j=\lceil nm/F_{max} \rceil + 1}^{i} \binom{i}{j} \rho^j (1-\rho)^{i-j}, \tag{15.1}$$

where ρ is the offered load per input, and F_{max} is the maximum number of copies requested by any packet. For better performance, a priority is assigned to each plane on the basis of usage. Therefore, planes that are being used less, as determined by the back-pressure information, are given higher priorities. The distribution of multicast packets among planes is done by using the following algorithm.

Begin Multicast Algorithm in 3-D Switches

1. If $F \leq m$: Send a packet to the first F available planes.
2. Each plane makes one copy of the packet and routes it to the appropriate output port, depending on the header information. Thus, we get the required F copies.
3. If $F > m$: Find $\frac{F}{m}$, the number of copies equally distributed among the planes, Q, and the copies left over, R.
4. Choose the highest-priority plane. For the first R planes, add 1 to the number of copies desired to be made by that plane, which is simply the value of Q. That is, the remaining R copies are further divided among the least-busy planes (or the highest-priority planes). The remaining $m - R$ planes now have to make Q copies.
5. The first Q (or $Q + 1$ for those planes that take care of the extra requests) addresses form the address space and add the new header to it, which has information about the destination plane, number of copies, and so on. Repeat this for all addresses.

■

A distribution circuit at the input has the processing ability to take the packet header and modify it according to the algorithm for distribution of packets to the planes. Some fields are added on top of the original packet. In fact, an original packet is encapsulated into this new packet with the new packet header.

15.5 Summary

Communication systems must be able to multicast a message to many users and even route many messages to one user. In a multipoint environment, traditional networks are subject to severe blocking, since they are not typically designed for high-bandwidth communications. Under both multirate and multipoint conditions, the performance of switching networks is closely related to the network architecture.

This chapter focused on multicast techniques, protocols, and algorithms in all levels: from the network layer to the physical layer. After defining some basic terms and techniques—including dense-mode and sparse-mode multicast tree algorithms—we discussed two main classes of protocols. *Intra*domain multicast routing protocols multicasts packets within a domain, and the *inter*domain routing protocol multicasts

packets among domains. Among the intradomain protocols, MOSPF is an extension to the unicast OSPF and based on the dense-mode tree algorithm. PIM protocols use sparse-mode tree algorithms.

The inclusion of multicasting capability into modern networks and routers requires considerable sophistication in the architecture and requires a great deal of performance evaluation in both analysis and simulation. Simplistic approaches at the node level can fail to meet the required network performance. A set of multicast algorithms were presented for the router level. The tree-based technique is appropriate for multistage switch fabrics. Multicasting by packet recirculation is a method that reduces switching-network complexity, resulting in less memory required for routing packets in multicast virtual circuits than with tree-based network architecture. However, extra hardware is needed to implement recirculation.

The next chapter looks at some other special-purpose routing situations. One major topic is how two branches of a geographically separate organization can create a secure route by tunneling and establishing a virtual private network (VPN). Other, related topics covered are multiprotocol label switching, point-to-point communications, and overlay networks.

15.6 Exercises

1. For sending a message from a point to multiple points, compare the trade-offs of using multiple unicast connections, and using any multicast approach, such as DVMRP, explained in this chapter.
2. Discuss the efficiency of MOSPF if the sparse-mode algorithm is used.
3. Consider Figure 15.5, and assume that we want to implement MOSPF from a server in LAN 1 to a multicast group of five members, three group members in LAN 3 and the two others in LAN 4. Also assume that the cost of each link is indicated by the sum of two ending routers, subscripts. For example, the cost of link R_3 to R_8 is 11.If a router ends a LAN, the cost of its associated link is assumed to be 1.
   (a) Show the cost of all links on the network.
   (b) From the least-cost tree for this multicast action, and find the root of copying routers.
4. To optimize the routing delay in a sparse-mode PIM:
   (a) Find the optimal place for the rendezvous point.
   (b) Give an example of a spanning tree, and justify your answer.

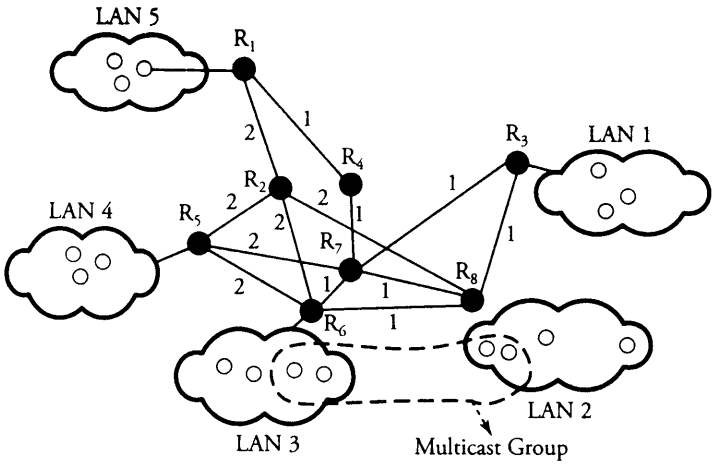

Figure 15.16 Exercise 7 example for multicast protocols

5. Consider Figure 15.6, and assume that we want to implement sparse-mode PIM from a server in LAN 1 to a multicast group of five members, with three group members in LAN 3 and the two others in LAN 4. Also assume that the cost of each link is indicated by the sum of two ending routers' subscripts. For example, the cost of link R_3 to R_8 is 11. If a router ends a LAN, the cost of its associated link is assumed to be 1.
   (a) Find a reasonably good location for the rendezvous point for this multicast action.
   (b) Form the least-cost tree for this multicast action, and find the root of copying routers.
6. Repeat exercise 5, but this time use the CBT protocol.
7. In Figure 15.16, five LANs, 1 through 5, are networked through their associated routers, R_3, R_8, R_6, R_5, and R_1. The cost of each link is indicated in the figure. A server in LAN 1 wants to send messages to the indicated multicast group.
   (a) For deploying the MOSPF protocol, find the multicast tree.
   (b) For deploying the sparse-mode PIM protocol and using the same tree you obtained in part (a), find the location of the rendezvous point and its associated costs to each multicast group member.

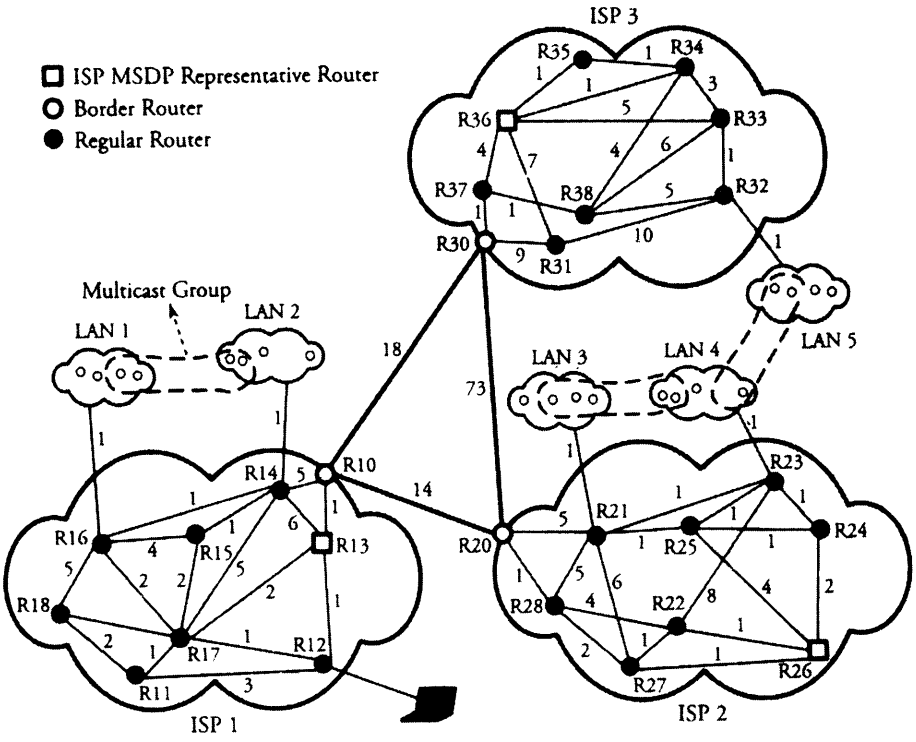

Figure 15.17 Exercise 9: using MSDP for interdomain multicasting and MOSPF and sparse-mode PIM protocols for intradomain multicasting

8. For a global number of copies $F = 7$, use the tree-based multicast algorithm in a $B_{16,2}$ copy network, and show details of stage-by-stage copying.
9. Three major Internet service provider domains—ISP 1, ISP 2, and ISP 3—are connected through three border routers: R10, R20, and R30, respectively, as shown in Figure 15.17. All three ISPs agree to use MSDP for interdomain multicasting. Each domain has selected a router as its ISP MSDP representative, which can also be used as a rendezvous point, if needed. A source in ISP 1 wants to multicast messages to three groups located in various geographical locations, as shown. ISP 1 uses MOSP, and ISP 2 and ISP 3 use sparse-mode PIM for intradomain multicasting. The cost of each link (equal in both directions) is indicated in the figure.

(a) Indicate all the involving routers in ISP 1, and find the total multicast cost to LAN 1 and LAN 2.
(b) Indicate all involving routers in ISP 2 and find the total multicast cost to LAN 3 and LAN 4.
(c) Indicate all involving routers in ISP 3 and find the total multicast cost to LAN 5.

10. For a global number of copies $F = 7$, use the Boolean splitting multicast algorithm in a $D_{16,2}$ copy network. Output ports 1, 2, 3, 7, 10, 12, and 15 receive a copy of packet.
(a) Show details of stage-by-stage copying.
(b) Compare the tree-based multicast algorithm with the Boolean splitting multicast algorithm in terms of speed and performance.

11. We want to show the copying mechanism in the switch fabric of routers whose global number of copies $F = 5$. The switch fabric is an Omega network $\Omega_{8,2}$ that has $r = 6$ stages.
(a) Use the Boolean splitting multicast algorithm. Output ports 1, 2, 4, 5, and 7 receive a copy of the packet. Show the details of stage-by-stage copying.
(b) Use the tree-based multicast algorithm to make five copies of a packet, and show the detail of stage-by-stage copying.

12. Consider a small switch fabric of 4×4 crossbar. Design the multicasting portion of its input port processor. Present all the hardware details.

13. *Computer Simulation Project*. Use the computer program you developed for the simulation of the seven-node network in Chapter 7 and modify it to model the network shown in Figure 15.16. Simulate a sparse-mode PIM for multicasting from router R_1 to four recipients located in a multicast group spread over a wide geographic area.
(a) Construct a multicast group.
(b) Write a computer program to do a sparse-mode multicast tree.
(c) Assign a rendezvous point.
(d) Demonstrate the formation of four packet copies made for the multicast group.

CHAPTER 16

VPNs, Tunneling, and Overlay Networks

Moving on to layers 3 and 4 of the protocol reference model, this chapter introduces some special-purpose networking features: how networks can be *overlaid* or *tunneled*. In networking, tunneling is the encapsulation of a packet from one protocol to another one at the same or higher layer. Among application-specific communication networking cases, *virtual private networks* (VPNs) and *multiprotocol label switching* (MPLS) networks are two important and popular ones discussed in this chapter. These two network infrastructures can also be tied together for certain applications. The topics covered in of this chapter are

- *Virtual private networks (VPNs)*
- *Multiprotocol label switching (MPLS)*
- *Overlay networks*

A VPN is a networking infrastructure whereby a private network makes use of the public network. A VPN maintains privacy by using tunneling protocols and security procedures. The two types of VPNs, each determined by its method of tunneling, are *remote access* and *site to site*.

MPLS networks are a good example of VPNs. In MPLS, multiple labels can be combined in a packet to form a header for efficient tunneling. The *Label Distribution Protocol* (LDP) is a set of rules by which routers exchange information effectively. MPLS uses *traffic engineering* for efficient link bandwidth assignments. Such networks

operate by establishing a secure *tunnel* over a public network. Finally, we look at *overlay networks*. An overlay network is a computer network that creates a virtual topology on top of the physical topology of the public network.

16.1 Virtual Private Networks (VPNs)

A *virtual private network* (VPN) is a data network having connections that make use of public networking facilities. The (VPN) part of public network is set up "virtually" by a private-sector entity to provide public networking services to small entities. With the globalization of businesses, many companies have facilities across the world and use VPNs to maintain fast, secure, and reliable communications across their branches.

VPNs are deployed with privacy through the use of a tunneling protocol and security procedures. Figure 16.1 shows two organizations, 1 and 3, connected through their corresponding routers, forming a tunnel in the public network, such as the Internet. Such a structure gives both private organizations the same capabilities they have on their own networks but at much lower cost. They can do this by using the shared public infrastructure. Creating a VPN benefits an organization benefits by providing

- Extended geographical communication
- Reduced operational cost
- Enhanced organizational management

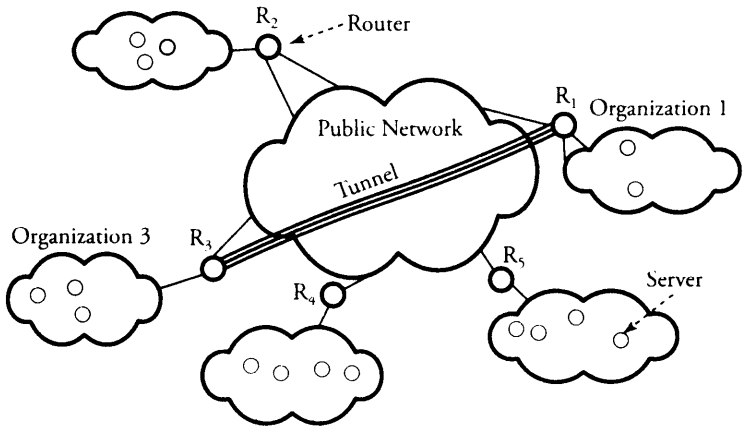

Figure 16.1 Two organizations connected through a tunnel using public facilities

- Enhanced network management with simplified local area networks
- Improved productivity and globalization

But since each user has no control over wires and routers, one of the issues with the Internet is still its lack of security, especially when a tunnel is exposed to the public. Thus, VPNs remain susceptible to security issues when they try to connect between two private networks using a public resource. The challenge in making a practical VPN, therefore, is finding the best security for it. Before discussing VPN security, we focus on types of VPNs. There are two types of VPNs each determined by its method of tunneling, *remote-access* and *site-to-site*. We will explain these two approaches in the next two sections.

16.1.1 Remote-Access VPN

Remote-access VPN is a user-to-LAN connection that an organization uses to connect its users to a private network from various remote locations. Large remote-access VPNs are normally outsourced to an Internet service provider to set up a *network-access server*. Other users, working off campus, can then reach the network-access server and use the VPN software to access the corporate network. Remote-access VPNs allow encrypted connections between an organization's private network and remote users through a third-party service provider.

Tunneling in a remote-access VPN uses mainly the *Point-to-Point Protocol* (PPP). PPP is the carrier for other Internet protocols when communicating over the network between a host computer and a remote point. Besides IPsec, other types of protocols associated with PPP are L2F, PPTP, and L2TP. The *Layer 2 Forwarding* (L2F) protocol uses the authentication scheme supported by PPP. The *Point-to-Point Tunneling Protocol* (PPTP) supports 40-bit and 128-bit encryption and uses the authentication scheme supported by PPP. The *Layer 2 Tunneling Protocol* (L2TP) combines features of both PPTP and L2F.

16.1.2 Site-to-Site VPN

By using effective security techniques, an organization can connect multiple fixed sites over a public network. *Site-to-site* VPNs can be classified as either intranets or extranets.

- *Intranet* VPNs connect an organization's remote-site LANs into a single private network.
- *Extranet* VPNs allow two organizations to work in a shared environment through a tunnel built to connect their LANs.

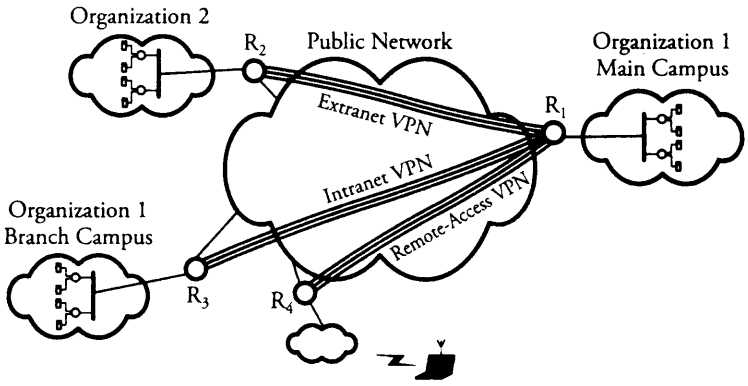

Figure 16.2 Three types of VPNs to and from a headquarter organization

Figure 16.2 shows the three types VPNs discussed so far. Organization 1's main campus and branch campus are connected through an intranet VPN tunnel. The main campus can also be connected to organization 2 through an extranet VPN tunnel. The employees of organization 1 can also access their corporation through a remote-access VPN. Each remote-access member must communicate in a secure medium. The main benefit of using a VPN is *scalability* with a reasonable cost. However, the physical and virtual distances of two communicating organizations have a great impact on the overall cost of building a VPN.

In a site-to-site VPN, *generic routing encapsulation* (GRE) is normally the encapsulating protocol. GRE provides the framework for the encapsulation over an IP-based protocol. IPsec in tunnel mode is sometimes used as the encapsulating protocol. IPsec works well on both remote-access and site-to-site VPNs but must be supported at both tunnel interfaces. The *Layer 2 Tunneling Protocol* (L2TP) can be used in site-to-site VPNs. L2TP fully supports IPsec regulations and can be used as a tunneling protocol for remote-access VPNs.

16.1.3 Tunneling and Point-to-Point Protocol (PPP)

A *tunnel* is a connection that forms a virtual network on top of a physical network. In computer networking, a tunnel resembles a telephone line in a public switched telephone network. VPNs typically rely on tunneling to create a private network that reaches across a public network. Tunneling is a process of encapsulating packets and sending them over the public network. Employees who are located outside an organization's main building can use point-to-point connections to create tunnels

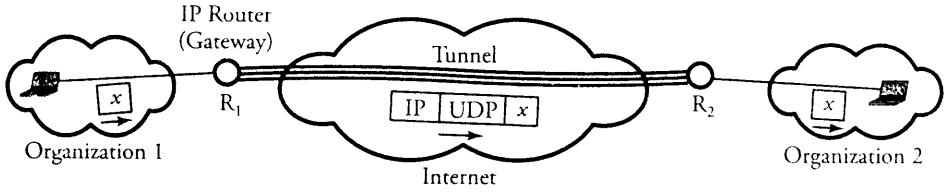

Figure 16.3 A customized protocol packet tunneling through the Internet

through the Internet. Since tunneling connections normally run over the Internet, they need to be secure. A tunnel is a relatively inexpensive connection, since it uses the Internet as its primary form of communication. Besides Internet protocols, tunneling requires two other types of protocols:

1. *Carrier protocols*, through which information travels over the public network
2. *Encapsulating protocols*, through which data is wrapped, encapsulated, and secured

One of the amazing implications of VPNs is that packets that use a protocol not supported on the Internet, such as NetBeui, can be placed inside an IP packet and sent safely over the Internet. VPNs can put a packet that uses a nonroutable IP address inside a packet to extend a private network over the Internet.

Consider the two LANs of the two organizations shown in Figure 16.3. We want to connect these two LANs through the Internet by using tunnels. Assume that the two LANs, as organization 1 and organization 2, want to use their own customized networking protocols, denoted by x, using connectionless datagram IP services. The IP resources can be at the scale of the Internet. Therefore, x-type packets cannot run over the Internet directly. The IP gateway R_1 listens for x-type packets on organization 1, encapsulates x-type packets in the transport-layer UDP datagrams, and transmits them over the Internet to R_2. When R_2 receives the encapsulates x packets, it decapsulates and feeds them into organization 2. This connection—in fact, a tunnel made through the Internet—resembles a direct physical link between the two LANs.

Point-to-Point Protocol (PPP)

The basic notion in tunneling is packet encapsulation from one protocol into the same or higher-layer protocol. Thus, a tunnel can also be defined as an encapsulating protocol for protocols at the lower layers. Tunneling protocols, such as the *Point-to-Point Protocol* (PPP) or the *Point-to-Point Tunneling Protocol* (PPTP) are encapsulating protocols that allow an organization to establish secure connections from one point to another while

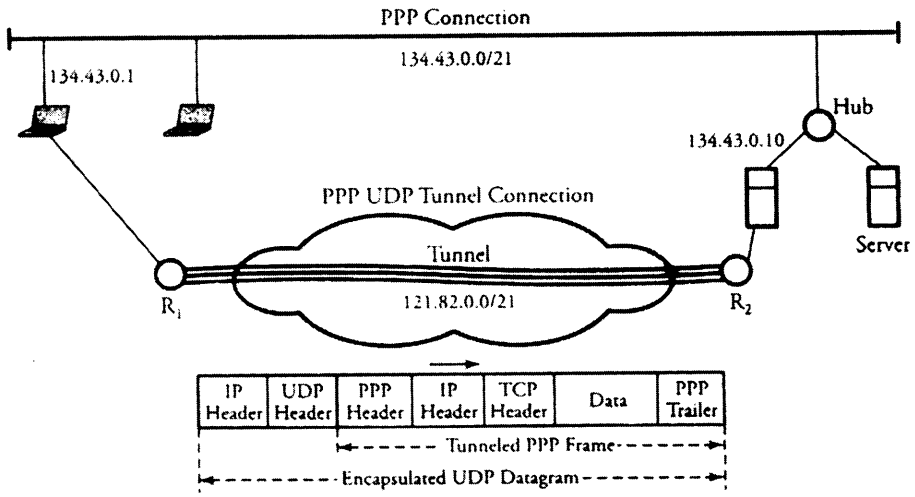

Figure 16.4 A point-to-point protocol (PPP) UDP tunnel connection

using public resources. A PPP connection is a serial connection between a user and an Internet service provider.

Example. In Figure 16.4, a point-to-point protocol (PPP) UDP tunnel connection is established while another virtual PPP connection exists. In this scenario, a user at 134.43.0.1 is communicating with a server at 134,43.0.10. These two end points are connected through their own PPP connection of 134.43.0.0/21, but the transmitted data flows through the tunnel on the 121.82.0.0/21 segments. This tunnel is established at an interface layer in a UDP transport-layer protocol as it appears from the frame format in the figure. Tunneling can also be formed at network-and-transport-layer protocols, where equal layers are involved, such as IP-in-IP tunnels.

16.1.4 Security in VPNs

Without using dedicated hardware, a VPN uses virtual connections routed through the Internet from the company's private network to the remote site. Companies can create their own VPNs to accommodate the needs of remote employees and distant offices. This section looks at methods for keeping VPN connections secure. A well-protected VPN uses firewalls, encryption systems, IPsec features, and an authentication server.

A firewall provides an effective barrier between a private network and the Internet. Firewalls can be set up to restrict the number of open ports to monitor what types of

packets are passed through and which protocols are allowed through. The authentication servers performs authentication, authorization, and accounting for more secure access in a remote-access environment. When a request to establish a session comes in, the request is loaded onto this server. The server then checks who the sender is (authentication), what it is allowed to do (authorization), and what it actually does (accounting and bills).

16.2 Multiprotocol Label Switching (MPLS)

Multiprotocol label switching (MPLS) improves the overall performance and delay characteristics of the Internet. MPLS transmission is a special case of tunneling and is an efficient routing mechanism. Its connection-oriented forwarding mechanism, together with layer 2 label-based lookups, enables *traffic engineering* to implement peer-to-peer VPNs effectively.

MPLS adds some traditional layer 2 capabilities and services, such as traffic engineering, to the IP layer. The separation of the MPLS control and forwarding components has led to multilayer, multiprotocol interoperability between layer 2 and layer 3 protocols. MPLS uses a small label or stack of labels appended to packets and typically makes efficient routing decisions. Another benefit is flexibility in merging IP-based networks with fast-switching capabilities. This technology adds new capabilities to IP-based networks:

- Connection-oriented QoS support
- Traffic engineering
- VPN support
- Multiprotocol support

Traditional IP routing has several limitations, ranging from scalability issues to poor support for traffic engineering. The IP backbone also presents a poor integration with layer 2 existing in large service provider networks. For example, a VPN must use a service provider's IP network and build a private network and run its own traffic shielded from prying eyes. In this case, VPN membership may not be well engineered in ordinary IP networks and can therefore result in an inefficient establishment of tunnels.

MPLS network architectures also support other applications, such as IP multicast routing and QoS extensions. The power of MPLS lies in the number of applications made possible with simple label switching, ranging from traffic engineering to peer-to-peer VPNs. One of the major advantages of MPLS is integration of the routing and

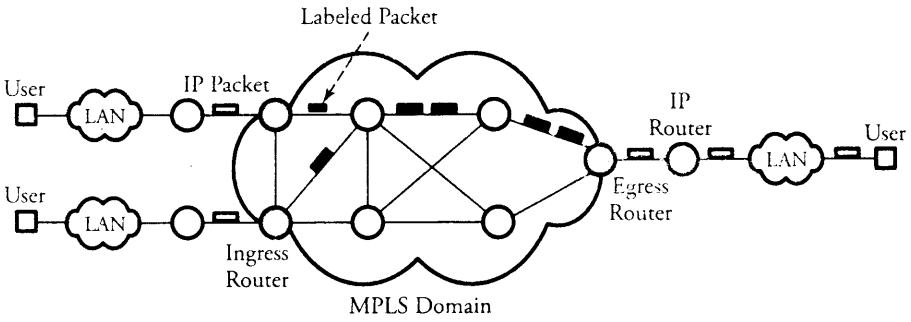

Figure 16.5 An MPLS network

switching layers. The development of the label-switched protocol running over all the existing layer 2 and layer 3 architectures is a major networking development.

16.2.1 MPLS Operation

MPLS is based on the assignment of *labels* to packets. Assigning labels to each packet makes a label-swapping scheme perform its routing process much more efficiently. An MPLS network consists of nodes called *label switch routers* (LSR). An LSR switches labeled packets according to particular switching tables. An LSR has two distinct functional components: a control component and a forwarding component. The control component uses routing protocols, such as OSPF and the *border gateway protocol* (BGP). The control component also facilitates the exchange of information with other LSRs to build and maintain the forwarding table.

A label is a header used by an LSR to forward packets. The header format depends on the network characteristics. LSRs read only labels and do not engage in the network-layer packet headers. One key to the scalability of MPLS is that labels have only local significance between two devices that communicate. When a packet arrives, the forwarding component uses the label of the packet as an index to search the forwarding table for a match. The forwarding component then directs the packet from the input interface to the output interface through the switching fabric.

MPSL Packet Format

MPLS uses *label stacking* to become capable of multilevel hierarchical routing. A label enables the network to perform faster by using smaller forwarding tables, a property that ensures a convenient scalability of the network. Figure 16.6 shows the MPLS header

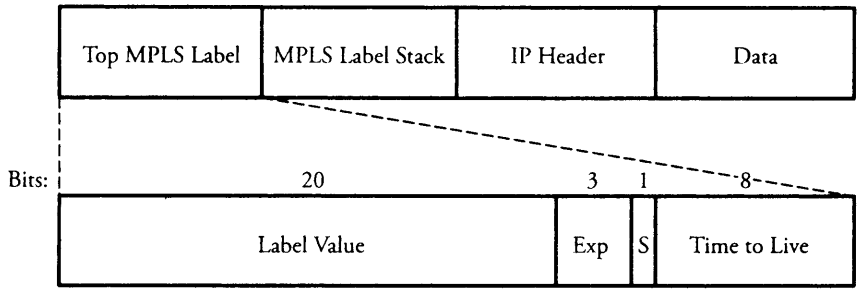

Figure 16.6 MPLS header encapsulation for an IP packet

encapsulation for an IP packet. An MPLS label is a 32-bit field consisting of several fields as follows.

- *Label value* is a 20-bit field label and is significant only locally.
- *Exp* is a 3-bit field reserved for future experimental use.
- *S* is set to 1 for the oldest entry in the stack and to 0 for all other entries.
- *Time to live* is an 8-bit field used to encode a hop-count value to prevent packets from looping forever in the network.

16.2.2 Routing in MPLS Domains

Figure 16.7 shows the label-switching paradigm in an MPLS network. An *ingress* LSR is an edge device that performs the initial packet processing and classification and applies the first label. An ingress LSR creates a new label. A *core* LSR swaps the incoming label with a corresponding next-hop label found from a forwarding table. At the other end of the network, another edge router, the *egress* LSR, is an outbound edge router and pops the label from the packet. It should be noted that multiple labels may be attached to a packet, forming a stack of labels. Label stacking enables multilevel hierarchical routing. For example, BGP labels are used for higher-level hierarchical packet forwarding from one BGP speaker to the other, whereas Interior Gateway Protocol (IGP) labels are used for packet forwarding within an autonomous system. Only the label at the top of the stack determines the forwarding decision.

Once an IP packet enters an MPLS domain, the ingress LSR processes its header information and maps that packet to a *forward equivalence class* (FEC). At this point, a *label switch path* (LSP) through the network must be defined, and the QoS parameters along that path must be established. The QoS parameters define how many resources

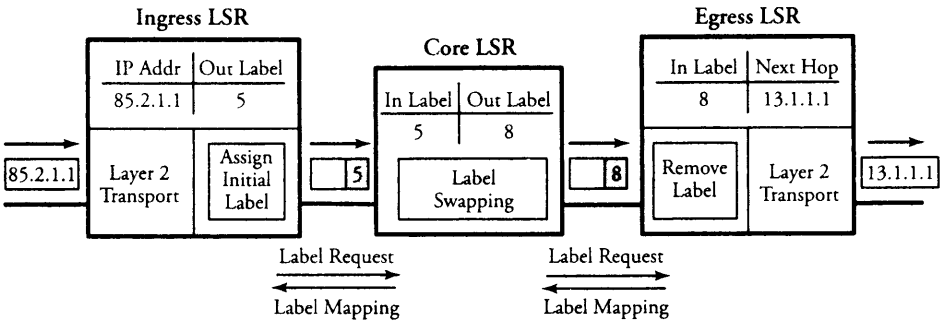

Figure 16.7 Multiple layer 2 switching example in MPLS

are to be used for the path and what queueing and discarding policy are to be used. For these functions, two protocols are used to exchange necessary information among routers: An intradomain routing protocol, such as OSPF, is used to exchange routing information, and the *Label Distribution Protocol* (LDP) assigns labels. At the end of the process, the router appends an appropriate label for FEC purposes and forwards the packet through.

Example. As shown in Figure 16.5, an IP packet with the destination address 85.2.1.1 enters an MPLS domain, and the ingress LSR processes its header and assigns a label, numbered 5. The label swapping from 5 to 8 takes place in the core router; finally, the next-hop IP address is attached to the packet on its way out of the egress router.

Packet forwarding at the core LSR is based on a label-swapping mechanism. Once it receives a labeled packet, the core LSR reads the label as an index to search in the *incoming label map table* for the corresponding next-hop label. The label in the MPLS header is swapped with the out-label and sent on the next hop. This method of packet forwarding simplifies the routing process by replacing the longest-prefix match of IP routing with simple short-label exact-match forwarding. The real benefit of this method is that instead of processing IP headers for forwarding packets, routers process a short label. Once a packet arrives at the egress LSR, its MPLS header is decapsulated, and the stripped packet is routed to its destination.

In summary, an MPLS domain has three label manipulation instructions: An *ingress* LSR creates a new label and pushes it to the label stack of a packet, a *core* LSR swaps the incoming label with a corresponding next-hop label found from the forwarding table, and an *egress* LSR (outbound edge router) pops a label from the label stack. Only the