CHAPTER 8

Switching

Switching is a topic that can be discussed at several layers. We have switching at the physical layer, at the data-link layer, at the network layer, and even logically at the application layer (message switching). We have decided to discuss the general idea behind switching in this chapter, the last chapter related to the physical layer. We particularly discuss circuit-switching, which occurs at the physical layer. We introduce the idea of packet-switching, which occurs at the data-link and network layers, but we postpone the details of these topics until the appropriate chapters. Finally, we talk about the physical structures of the switches and routers.

This chapter is divided into four sections:

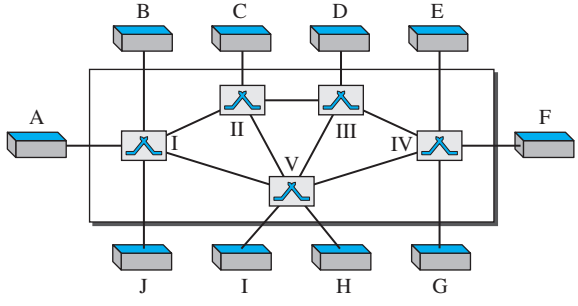
- The first section introduces switching. It mentions three methods of switching: circuit switching, packet switching, and message switching. The section then defines the switching methods that can occur in some layers of the Internet model.
- The second section discusses circuit-switched networks. It first defines three phases in these types of networks. It then describes the efficiency of these networks. The section also discusses the delay in circuit-switched networks.
- The third section briefly discusses packet-switched networks. It first describes datagram networks, listing their characteristics and advantages. The section then describes virtual circuit networks, explaining their features and operations. We will discuss packet-switched networks in more detail in Chapter 18.
- The last section discusses the structure of a switch. It first describes the structure of a circuit switch. It then explains the structure of a packet switch.

8.1 INTRODUCTION

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology). These methods, however, are impractical and wasteful when applied to very large networks. The number and length of the links require too much infrastructure to be cost-efficient, and the majority of those links would be idle most of the time. Other topologies employing multipoint connections, such as a bus, are ruled out because the distances between devices and the total number of devices increase beyond the capacities of the media and equipment.

A better solution is **switching.** A switched network consists of a series of interlinked nodes, called ***switches*****.** Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing. Figure 8.1 shows a switched network.

Figure 8.1 *Switched network*

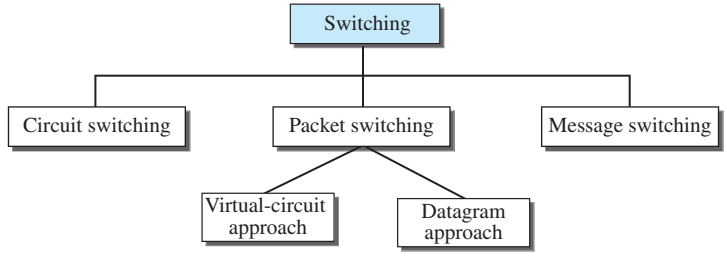


The **end systems** (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links.

8.1.1 Three Methods of Switching

Traditionally, three methods of switching have been discussed: **circuit switching**, **packet switching**, and **message switching**. The first two are commonly used today. The third has been phased out in general communications but still has networking applications. Packet switching can further be divided into two subcategories—virtual-circuit approach and datagram approach—as shown in Figure 8.2. In this chapter, we discuss only circuit switching and packet switching; message switching is more conceptual than practical.

Figure 8.2 *Taxonomy of switched networks*



8.1.2 Switching and TCP/IP Layers

Switching can happen at several layers of the TCP/IP protocol suite.

Switching at Physical Layer

At the physical layer, we can have only circuit switching. There are no packets exchanged at the physical layer. The switches at the physical layer allow signals to travel in one path or another.

Switching at Data-Link Layer

At the data-link layer, we can have packet switching. However, the term *packet* in this case means *frames* or *cells*. Packet switching at the data-link layer is normally done using a virtual-circuit approach.

Switching at Network Layer

At the network layer, we can have packet switching. In this case, either a virtual-circuit approach or a datagram approach can be used. Currently the Internet uses a datagram approach, as we see in Chapter 18, but the tendency is to move to a virtual-circuit approach.

Switching at Application Layer

At the application layer, we can have only message switching. The communication at the application layer occurs by exchanging messages. Conceptually, we can say that communication using e-mail is a kind of message-switched communication, but we do not see any network that actually can be called a message-switched network.

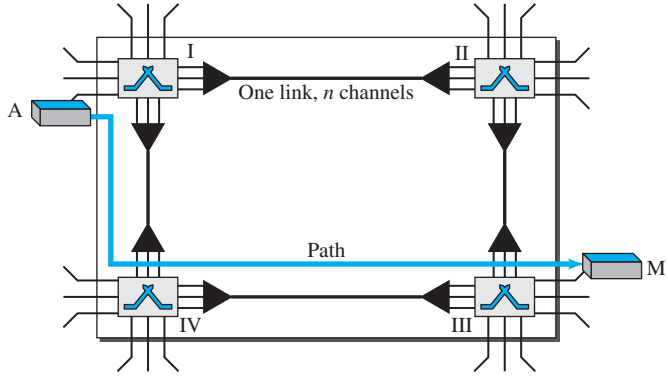
8.2 CIRCUIT-SWITCHED NETWORKS

A **circuit-switched network** consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links. However, each connection uses only one dedicated channel on each link. Each link is normally divided into *n* channels by using FDM or TDM, as discussed in Chapter 6.

A circuit-switched network is made of a set of switches connected by physical links, in which each link is divided into *n* channels.

Figure 8.3 shows a trivial circuit-switched network with four switches and four links. Each link is divided into *n* (*n* is 3 in the figure) channels by using FDM or TDM.

Figure 8.3 *A trivial circuit-switched network*



We have explicitly shown the multiplexing symbols to emphasize the division of the link into channels even though multiplexing can be implicitly included in the switch fabric.

The end systems, such as computers or telephones, are directly connected to a switch. We have shown only two end systems for simplicity. When end system A needs to communicate with end system M, system A needs to request a connection to M that must be accepted by all switches as well as by M itself. This is called the **setup phase;** a circuit (channel) is reserved on each link, and the combination of circuits or channels defines the dedicated path. After the dedicated path made of connected circuits (channels) is established, the **data-transfer phase** can take place. After all data have been transferred, the circuits are torn down.

We need to emphasize several points here:

- Circuit switching takes place at the physical layer.
- Before starting communication, the stations must make a reservation for the resources to be used during the communication. These resources, such as channels (bandwidth in FDM and time slots in TDM), switch buffers, switch processing time, and switch input/output ports, must remain dedicated during the entire duration of data transfer until the **teardown phase.**
- Data transferred between the two stations are not packetized (physical layer transfer of the signal). The data are a continuous flow sent by the source station and received by the destination station, although there may be periods of silence.

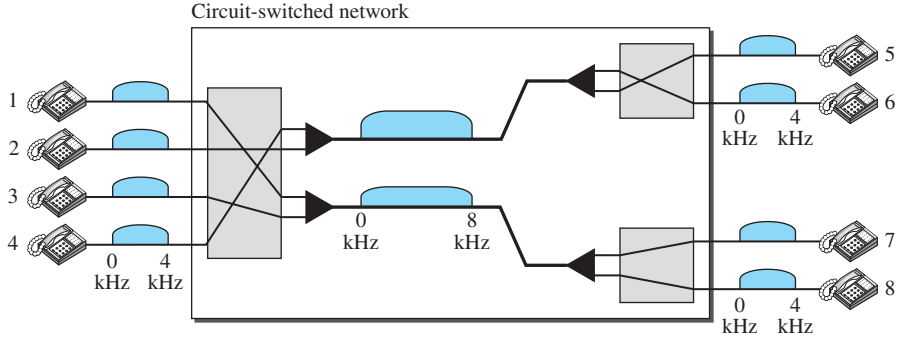
- There is no addressing involved during data transfer. The switches route the data based on their occupied band (FDM) or time slot (TDM). Of course, there is end-to-end addressing used during the setup phase, as we will see shortly.

In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of data transfer until the teardown phase.

Example 8.1

As a trivial example, let us use a circuit-switched network to connect eight telephones in a small area. Communication is through 4-kHz voice channels. We assume that each link uses FDM to connect a maximum of two voice channels. The bandwidth of each link is then 8 kHz. Figure 8.4 shows the situation. Telephone 1 is connected to telephone 7; 2 to 5; 3 to 8; and 4 to 6. Of course the situation may change when new connections are made. The switch controls the connections.

Figure 8.4 *Circuit-switched network used in Example 8.1*



Example 8.2

As another example, consider a circuit-switched network that connects computers in two remote offices of a private company. The offices are connected using a T-1 line leased from a communication service provider. There are two 4×8 (4 inputs and 8 outputs) switches in this network. For each switch, four output ports are folded into the input ports to allow communication between computers in the same office. Four other output ports allow communication between the two offices. Figure 8.5 shows the situation.

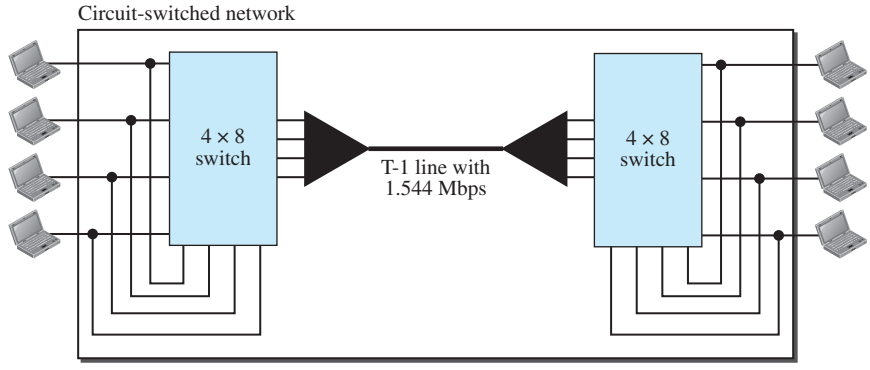
8.2.1 Three Phases

The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

Setup Phase

Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup

Figure 8.5 *Circuit-switched network used in Example 8.2*



means creating dedicated channels between the switches. For example, in Figure 8.3, when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time.

In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established.

Note that end-to-end addressing is required for creating a connection between the two end systems. These can be, for example, the addresses of the computers assigned by the administrator in a TDM network, or telephone numbers in an FDM network.

Data-Transfer Phase

After the establishment of the dedicated circuit (channels), the two parties can transfer data.

Teardown Phase

When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

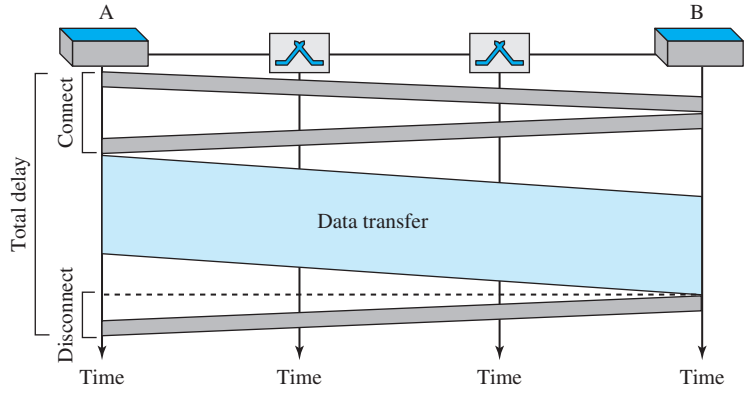
8.2.2 Efficiency

It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections. In a telephone network, people normally terminate the communication when they have finished their conversation. However, in computer networks, a computer can be connected to another computer even if there is no activity for a long time. In this case, allowing resources to be dedicated means that other connections are deprived.

8.2.3 Delay

Although a circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. Figure 8.6 shows the idea of delay in a circuit-switched network when only two switches are involved.

Figure 8.6 *Delay in a circuit-switched network*



As Figure 8.6 shows, there is no waiting time at each switch. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit. The delay caused by the setup is the sum of four parts: the propagation time of the source computer request (slope of the first gray box), the request signal transfer time (height of the first gray box), the propagation time of the acknowledgment from the destination computer (slope of the second gray box), and the signal transfer time of the acknowledgment (height of the second gray box). The delay due to data transfer is the sum of two parts: the propagation time (slope of the colored box) and data transfer time (height of the colored box), which can be very long. The third box shows the time needed to tear down the circuit. We have shown the case in which the receiver requests disconnection, which creates the maximum delay.

8.3 PACKET SWITCHING

In data communications, we need to send messages from one end system to another. If the message is going to pass through a **packet-switched network**, it needs to be divided into packets of fixed or variable size. The size of the packet is determined by the network and the governing protocol.

In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand. The allocation is done on a first-come, first-served basis. When a switch receives a packet, no matter what the source or destination is, the packet must wait if there are other packets being processed. As with

other systems in our daily life, this lack of reservation may create delay. For example, if we do not have a reservation at a restaurant, we might have to wait.

In a packet-switched network, there is no resource reservation; resources are allocated on demand.

We can have two types of packet-switched networks: datagram networks and virtual-circuit networks.

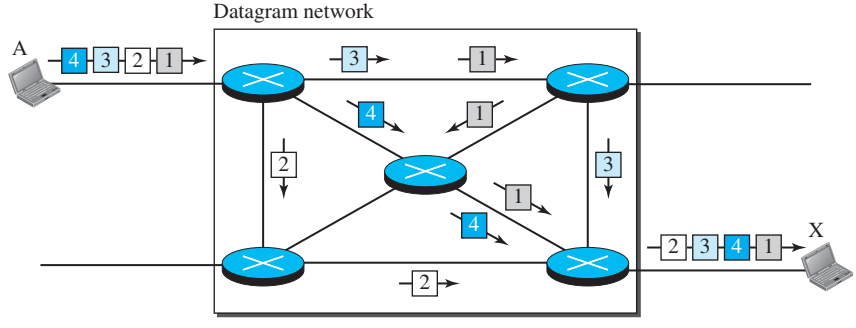
8.3.1 Datagram Networks

In a **datagram network**, each packet is treated independently of all others. Even if a packet is part of a multipacket transmission, the network treats it as though it existed alone. Packets in this approach are referred to as ***datagrams***.

Datagram switching is normally done at the network layer. We briefly discuss datagram networks here as a comparison with circuit-switched and virtual-circuit-switched networks. In Chapter 18 of this text, we go into greater detail.

Figure 8.7 shows how the datagram approach is used to deliver four packets from station A to station X. The switches in a datagram network are traditionally referred to as routers. That is why we use a different symbol for the switches in the figure.

Figure 8.7 *A datagram network with four switches (routers)*



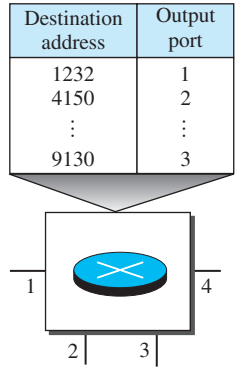
In this example, all four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X. This approach can cause the datagrams of a transmission to arrive at their destination out of order with different delays between the packets. Packets may also be lost or dropped because of a lack of resources. In most protocols, it is the responsibility of an upper-layer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application.

The datagram networks are sometimes referred to as *connectionless networks*. The term *connectionless* here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

Routing Table

If there are no setup or teardown phases, how are the packets routed to their destinations in a datagram network? In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. This is different from the table of a circuit-switched network (discussed later) in which each entry is created when the setup phase is completed and deleted when the teardown phase is over. Figure 8.8 shows the routing table for a switch.

Figure 8.8 *Routing table in a datagram network*



A switch in a datagram network uses a routing table that is based on the destination address.

Destination Address

Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be forwarded. This address, unlike the address in a virtual-circuit network, remains the same during the entire journey of the packet.

The destination address in the header of a packet in a datagram network remains the same during the entire journey of the packet.

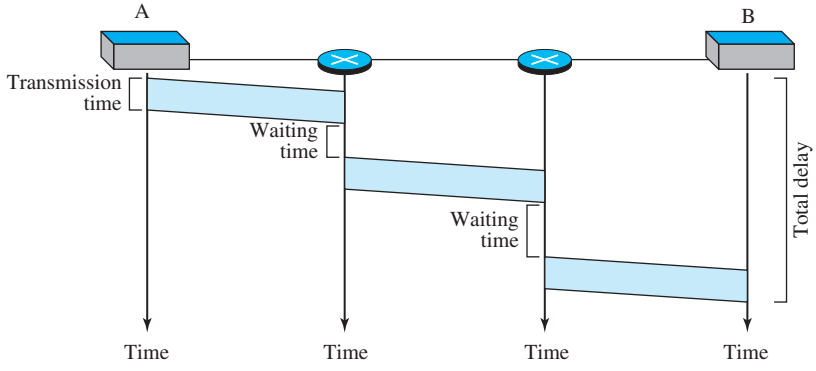
Efficiency

The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred. If a source sends a packet and there is a delay of a few minutes before another packet can be sent, the resources can be reallocated during these minutes for other packets from other sources.

Delay

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded. In addition, since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message. Figure 8.9 gives an example of delay in a datagram network for one packet.

Figure 8.9 *Delay in a datagram network*



The packet travels through two switches. There are three transmission times ($3T$), three propagation delays (slopes 3τ of the lines), and two waiting times ($w_1 + w_2$). We ignore the processing time in each switch. The total delay is

$$\textbf{Total delay} = 3T + 3\tau + w_1 + w_2$$

8.3.2 Virtual-Circuit Networks

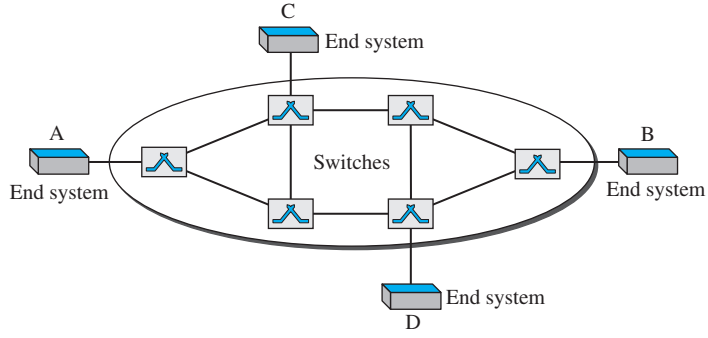
A **virtual-circuit network** is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what the next switch should be and the channel on which the packet is being carried), not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet. The answer will be clear when we discuss virtual-circuit identifiers in the next section.
4. As in a circuit-switched network, all packets follow the same path established during the connection.

5. A virtual-circuit network is normally implemented in the data-link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Figure 8.10 is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

Figure 8.10 *Virtual-circuit network*



Addressing

In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

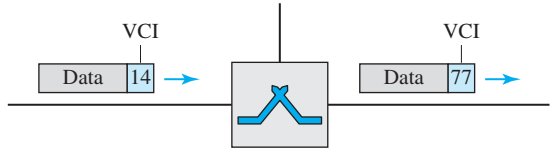
Global Addressing

A source or a destination needs to have a global address—an address that can be unique in the scope of the network or internationally if the network is part of an international network. However, we will see that a global address in virtual-circuit networks is used only to create a virtual-circuit identifier, as discussed next.

Virtual-Circuit Identifier

The identifier that is actually used for data transfer is called the ***virtual-circuit identifier*** **(VCI)** or the ***label.*** A VCI, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCI. Figure 8.11 shows how the VCI in a data frame changes from one switch to another. Note that a VCI does not need to be a large number since each switch can use its own unique set of VCIs.

Figure 8.11 *Virtual-circuit identifier*



Three Phases

As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: setup, data transfer, and teardown. In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection. In the teardown phase, the source and destination inform the switches to delete the corresponding entry. Data transfer occurs between these two phases. We first discuss the data-transfer phase, which is more straightforward; we then talk about the setup and teardown phases.

Data-Transfer Phase

To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit. The table, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up. We show later how the switches make their table entries, but for the moment we assume that each switch has a table with entries for all active virtual circuits. Figure 8.12 shows such a switch and its corresponding table.

Figure 8.12 *Switch and tables in a virtual-circuit network*

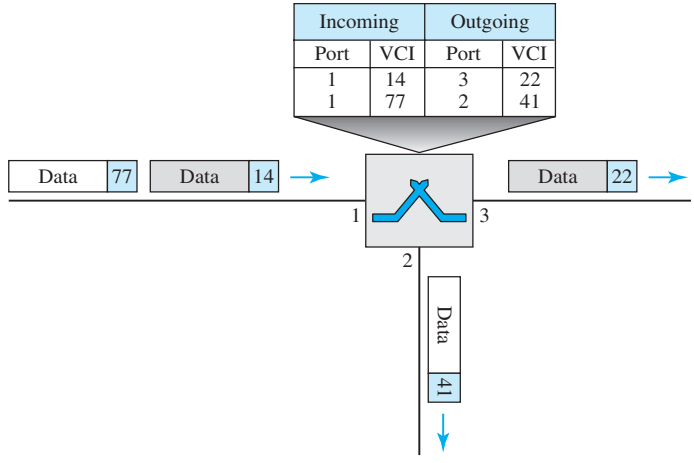


Figure 8.12 shows a frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3.

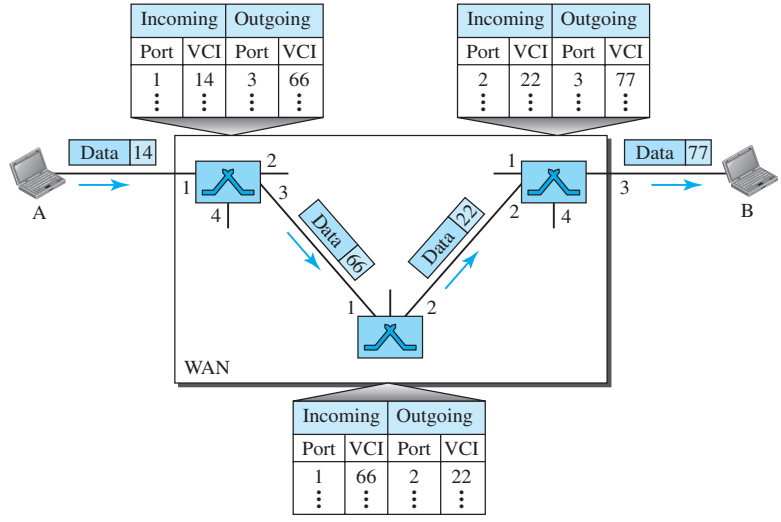
Figure 8.13 shows how a frame from source A reaches destination B and how its VCI changes during the trip. Each switch changes the VCI and routes the frame.

The data-transfer phase is active until the source sends all its frames to the destination. The procedure at the switch is the same for each frame of a message. The process creates a virtual circuit, not a real circuit, between the source and destination.

Setup Phase

In the setup phase, a switch creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to B. Two steps are required: the setup request and the acknowledgment.

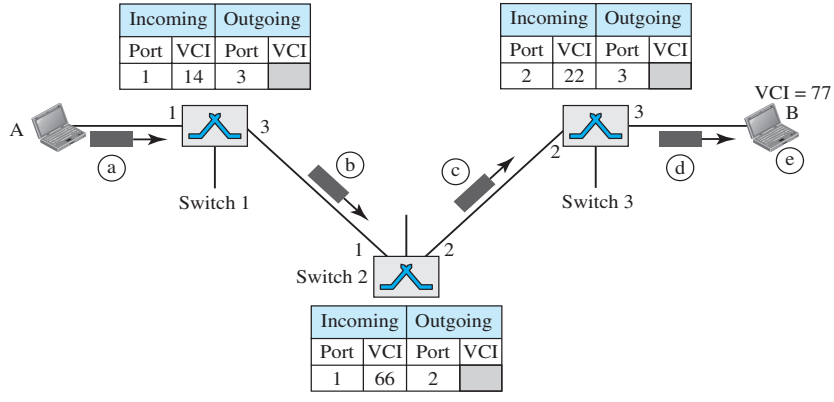
Figure 8.13 *Source-to-destination data transfer in a virtual-circuit network*



Setup Request

A setup request frame is sent from the source to the destination. Figure 8.14 shows the process.

Figure 8.14 *Setup request in a virtual-circuit network*



a. Source A sends a setup frame to switch 1.

b. Switch 1 receives the setup request frame. It knows that a frame going from A to B goes out through port 3. How the switch has obtained this information is a point covered in future chapters. The switch, in the setup phase, acts as a packet switch; it has a routing table which is different from the switching table. For the moment, assume that it knows the output port. The switch creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The switch assigns the incoming port (1) and chooses an available incoming VCI (14) and the

outgoing port (3). It does not yet know the outgoing VCI, which will be found during the acknowledgment step. The switch then forwards the frame through port 3 to switch 2.

c. Switch 2 receives the setup request frame. The same events happen here as at switch 1; three columns of the table are completed: in this case, incoming port (1), incoming VCI (66), and outgoing port (2).

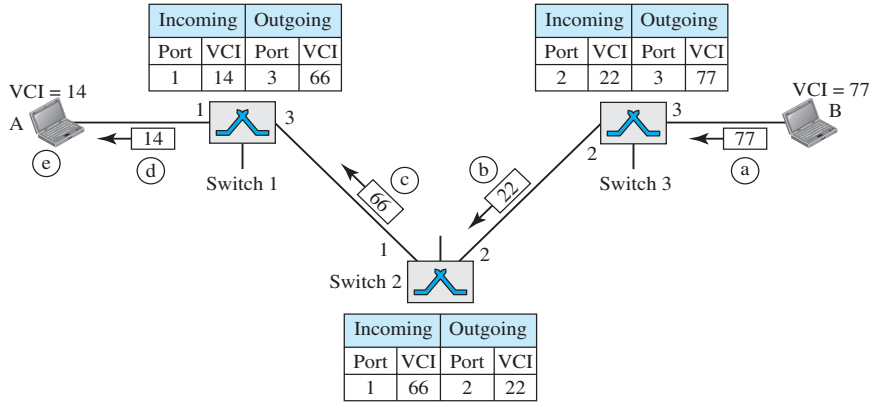
d. Switch 3 receives the setup request frame. Again, three columns are completed: incoming port (2), incoming VCI (22), and outgoing port (3).

e. Destination B receives the setup frame, and if it is ready to receive frames from A, it assigns a VCI to the incoming frames that come from A, in this case 77. This VCI lets the destination know that the frames come from A, and not other sources.

Acknowledgment

A special frame, called the *acknowledgment frame,* completes the entries in the switching tables. Figure 8.15 shows the process.

Figure 8.15 *Setup acknowledgment in a virtual-circuit network*



a. The destination sends an acknowledgment to switch 3. The acknowledgment carries the global source and destination addresses so the switch knows which entry in the table is to be completed. The frame also carries VCI 77, chosen by the destination as the incoming VCI for frames from A. Switch 3 uses this VCI to complete the outgoing VCI column for this entry. Note that 77 is the incoming VCI for destination B, but the outgoing VCI for switch 3.

b. Switch 3 sends an acknowledgment to switch 2 that contains its incoming VCI in the table, chosen in the previous step. Switch 2 uses this as the outgoing VCI in the table.

c. Switch 2 sends an acknowledgment to switch 1 that contains its incoming VCI in the table, chosen in the previous step. Switch 1 uses this as the outgoing VCI in the table.

d. Finally switch 1 sends an acknowledgment to source A that contains its incoming VCI in the table, chosen in the previous step.

e. The source uses this as the outgoing VCI for the data frames to be sent to destination B.

Teardown Phase

In this phase, source A, after sending all frames to B, sends a special frame called a *teardown request*. Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

Efficiency

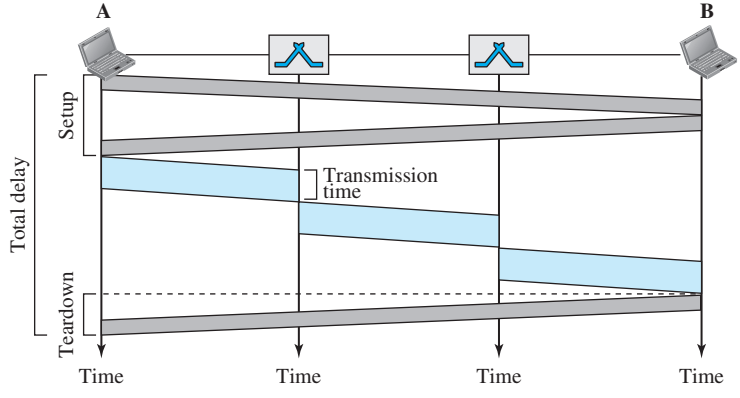
As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data-transfer phase. In the first case, the delay for each packet is the same; in the second case, each packet may encounter different delays. There is one big advantage in a virtual-circuit network even if resource allocation is on demand. The source can check the availability of the resources, without actually reserving it. Consider a family that wants to dine at a restaurant. Although the restaurant may not accept reservations (allocation of the tables is on demand), the family can call and find out the waiting time. This can save the family time and effort.

In virtual-circuit switching, all packets belonging to the same source and destination travel the same path, but the packets may arrive at the destination with different delays if resource allocation is on demand.

Delay in Virtual-Circuit Networks

In a virtual-circuit network, there is a one-time delay for setup and a one-time delay for teardown. If resources are allocated during the setup phase, there is no wait time for individual packets. Figure 8.16 shows the delay for a packet traveling through two switches in a virtual-circuit network.

Figure 8.16 *Delay in a virtual-circuit network*



The packet is traveling through two switches (routers). There are three transmission times ($3T$), three propagation times (3τ), data transfer depicted by the sloping lines, a setup delay (which includes transmission and propagation in two directions),

and a teardown delay (which includes transmission and propagation in one direction). We ignore the processing time in each switch. The total delay time is

Total delay = 3*T* + 3τ + setup delay + teardown delay

Circuit-Switched Technology in WANs

As we will see in Chapter 14, virtual-circuit networks are used in switched WANs such as ATM networks. The data-link layer of these technologies is well suited to the virtual-circuit technology.

Switching at the data-link layer in a switched WAN is normally implemented by using virtual-circuit techniques.

8.4 STRUCTURE OF A SWITCH

We use switches in circuit-switched and packet-switched networks. In this section, we discuss the structures of the switches used in each type of network.

8.4.1 Structure of Circuit Switches

Circuit switching today can use either of two technologies: the space-division switch or the time-division switch.

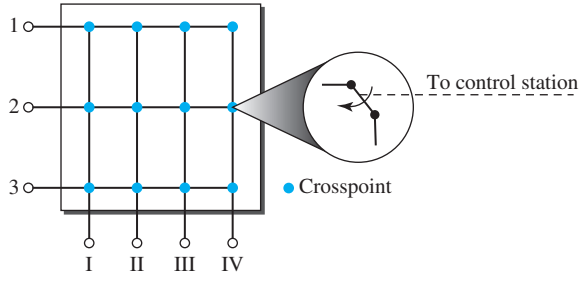
Space-Division Switch

In **space-division switching,** the paths in the circuit are separated from one another spatially. This technology was originally designed for use in analog networks but is used currently in both analog and digital networks. It has evolved through a long history of many designs.

Crossbar Switch

A **crossbar switch** connects *n* inputs to *m* outputs in a grid, using electronic microswitches (transistors) at each **crosspoint** (see Figure 8.17). The major limitation of this design is the number of crosspoints required. To connect *n* inputs to *m* outputs using a

Figure 8.17 *Crossbar switch with three inputs and four outputs*

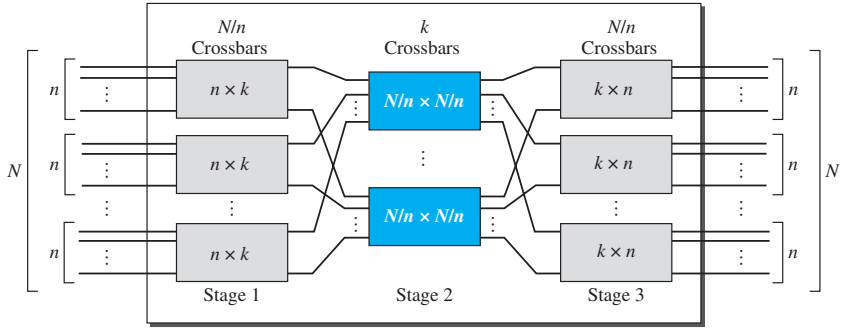


crossbar switch requires $n \times m$ crosspoints. For example, to connect 1000 inputs to 1000 outputs requires a switch with 1,000,000 crosspoints. A crossbar switch [?] with this number of crosspoints is impractical. Such a switch is also inefficient because statistics show that, in practice, fewer than 25 percent of the crosspoints are in use at any given time. The rest are idle.

Multistage Switch

The solution to the limitations of the crossbar switch is the **multistage switch,** which combines crossbar switches in several (normally three) stages, as shown in Figure 8.18. In a single crossbar switch, only one row or column (one path) is active for any connection. So we need $N \times N$ crosspoints. If we can allow multiple paths inside the switch, we can decrease the number of crosspoints. Each crosspoint in the middle stage can be accessed by multiple crosspoints in the first or third stage.

Figure 8.18 *Multistage switch*



To design a three-stage switch, we follow these steps:

1. We divide the N input lines into groups, each of n lines. For each group, we use one crossbar of size $n \times k$, where k is the number of crossbars in the middle stage. In other words, the first stage has N/n crossbars of $n \times k$ crosspoints.
2. We use k crossbars, each of size $(N/n) \times (N/n)$ in the middle stage.
3. We use N/n crossbars, each of size $k \times n$ at the third stage.

We can calculate the total number of crosspoints as follows:

$$\frac{N}{n}(n \times k) + k\left(\frac{N}{n} \times \frac{N}{n}\right) + \frac{N}{n}(k \times n) = 2kN + k\left(\frac{N}{n}\right)^2$$

In a three-stage switch, the total number of crosspoints is

$$2kN + k\left(\frac{N}{n}\right)^2$$

which is much smaller than the number of crosspoints in a single-stage switch (N^2).

Example 8.3

Design a three-stage, 200 × 200 switch ($N = 200$) with $k = 4$ and $n = 20$.

Solution

In the first stage we have N/n or 10 crossbars, each of size 20 × 4. In the second stage, we have 4 crossbars, each of size 10 × 10. In the third stage, we have 10 crossbars, each of size 4 × 20. The total number of crosspoints is $2kN + k(N/n)^2$, or 2000 crosspoints. This is 5 percent of the number of crosspoints in a single-stage switch (200 × 200 = 40,000).

The multistage switch in Example 8.3 has one drawback—**blocking** during periods of heavy traffic. The whole idea of multistage switching is to share the crosspoints in the middle-stage crossbars. Sharing can cause a lack of availability if the resources are limited and all users want a connection at the same time. *Blocking* refers to times when one input cannot be connected to an output because there is no path available between them—all the possible intermediate switches are occupied.

In a single-stage switch, blocking does not occur because every combination of input and output has its own crosspoint; there is always a path. (Cases in which two inputs are trying to contact the same output do not count. That path is not blocked; the output is merely busy.) In the multistage switch described in Example 8.3, however, only four of the first 20 inputs can use the switch at a time, only four of the second 20 inputs can use the switch at a time, and so on. The small number of crossbars at the middle stage creates blocking.

In large systems, such as those having 10,000 inputs and outputs, the number of stages can be increased to cut down on the number of crosspoints required. As the number of stages increases, however, possible blocking increases as well. Many people have experienced blocking on public telephone systems in the wake of a natural disaster when the calls being made to check on or reassure relatives far outnumber the regular load of the system.

Clos investigated the condition of nonblocking in multistage switches and came up with the following formula. In a nonblocking switch, the number of middle-stage switches must be at least $2n - 1$. In other words, we need to have $k \geq 2n - 1$.

Note that the number of crosspoints is still smaller than that in a single-stage switch. Now we need to minimize the number of crosspoints with a fixed N by using the Clos criteria. We can take the derivative of the equation with respect to n (the only variable) and find the value of n that makes the result zero. This n must be equal to or greater than $(N/2)^{1/2}$. In this case, the total number of crosspoints is greater than or equal to $4N\,[(2N)^{1/2} - 1]$. In other words, the minimum number of crosspoints according to the Clos criteria is proportional to $N^{3/2}$.

According to Clos criterion: $\boldsymbol{n = (N/2)^{1/2}}$ ***and*** $\boldsymbol{k \geq 2n - 1}$
Total number of crosspoints $\boldsymbol{\geq 4N\,[(2N)^{1/2} - 1]}$

Example 8.4

Redesign the previous three-stage, 200 × 200 switch, using the Clos criteria with a minimum number of crosspoints.

Solution

We let $n = (200/2)^{1/2}$, or $n = 10$. We calculate $k = 2n - 1 = 19$. In the first stage, we have 200/10, or 20, crossbars, each with 10 × 19 crosspoints. In the second stage, we have 19 crossbars,

each with 10 × 10 crosspoints. In the third stage, we have 20 crossbars each with 19 × 10 crosspoints. The total number of crosspoints is 20(10 × 19) + 19(10 × 10) + 20(19 × 10) = 9500. If we use a single-stage switch, we need 200 × 200 = 40,000 crosspoints. The number of crosspoints in this three-stage switch is 24 percent that of a single-stage switch. More points are needed than in Example 8.3 (5 percent). The extra crosspoints are needed to prevent blocking.

A multistage switch that uses the Clos criteria and a minimum number of crosspoints still requires a huge number of crosspoints. For example, to have a 100,000 input/output switch, we need something close to 200 million crosspoints (instead of 10 billion). This means that if a telephone company needs to provide a switch to connect 100,000 telephones in a city, it needs 200 million crosspoints. The number can be reduced if we accept blocking. Today, telephone companies use time-division switching or a combination of space- and time-division switches, as we will see shortly.

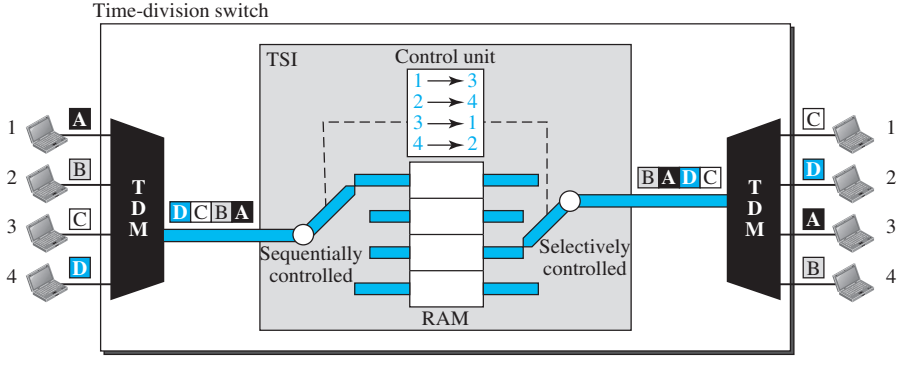
Time-Division Switch

Time-division switching uses time-division multiplexing (TDM) inside a switch. The most popular technology is called the **time-slot interchange (TSI).**

Time-Slot Interchange

Figure 8.19 shows a system connecting four input lines to four output lines. Imagine that each input line wants to send data to an output line according to the following pattern: (1 → 3), (2 → 4), (3 → 1), and (4 → 2), in which the arrow means "to."

Figure 8.19 *Time-slot interchange*



The figure combines a TDM multiplexer, a TDM demultiplexer, and a TSI consisting of random access memory (RAM) with several memory locations. The size of each location is the same as the size of a single time slot. The number of locations is the same as the number of inputs (in most cases, the numbers of inputs and outputs are equal). The RAM fills up with incoming data from time slots in the order received. Slots are then sent out in an order based on the decisions of a control unit.

Time- and Space-Division Switch Combinations

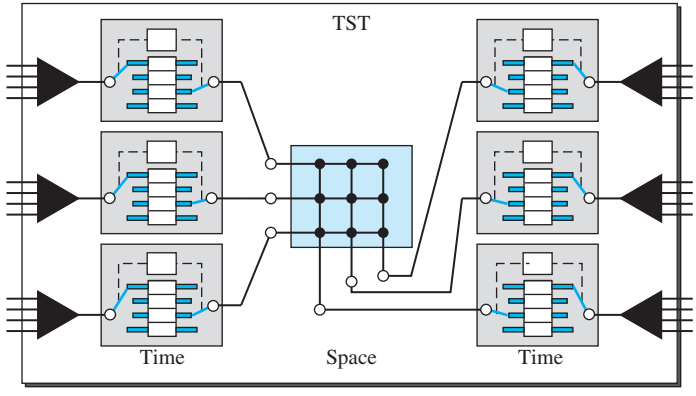
When we compare space-division and time-division switching, some interesting facts emerge. The advantage of space-division switching is that it is instantaneous. Its disadvantage is the number of crosspoints required to make space-division switching acceptable in terms of blocking.

The advantage of time-division switching is that it needs no crosspoints. Its disadvantage, in the case of TSI, is that processing each connection creates delays. Each time slot must be stored by the RAM, then retrieved and passed on.

In a third option, we combine space-division and time-division technologies to take advantage of the best of both. Combining the two results in switches that are optimized both physically (the number of crosspoints) and temporally (the amount of delay). Multistage switches of this sort can be designed as **time-space-time (TST) switches.**

Figure 8.20 shows a simple TST switch that consists of two time stages and one space stage and has 12 inputs and 12 outputs. Instead of one time-division switch, it divides the inputs into three groups (of four inputs each) and directs them to three time-slot interchanges. The result is that the average delay is one-third of what would result from using one time-slot interchange to handle all 12 inputs.

Figure 8.20 *Time-space-time switch*

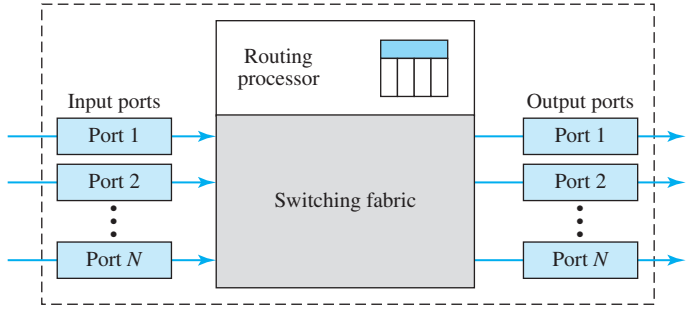


The last stage is a mirror image of the first stage. The middle stage is a space-division switch (crossbar) that connects the TSI groups to allow connectivity between all possible input and output pairs (e.g., to connect input 3 of the first group to output 7 of the second group).

8.4.2 Structure of Packet Switches

A switch used in a packet-switched network has a different structure from a switch used in a circuit-switched network.We can say that a packet switch has four components: **input ports, output ports,** the **routing processor,** and the **switching fabric,** as shown in Figure 8.21.

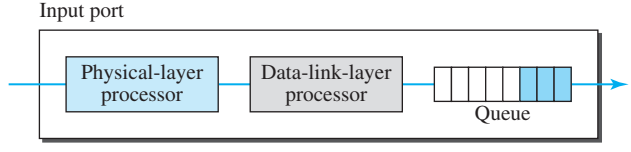
Figure 8.21 *Packet switch components*



Input Ports

An input port performs the physical and data-link functions of the packet switch. The bits are constructed from the received signal. The packet is decapsulated from the frame. Errors are detected and corrected. The packet is now ready to be routed by the network layer. In addition to a physical-layer processor and a data-link processor, the input port has buffers (queues) to hold the packet before it is directed to the switching fabric. Figure 8.22 shows a schematic diagram of an input port.

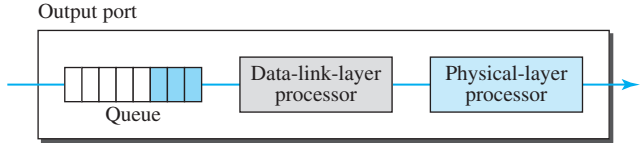
Figure 8.22 *Input port*



Output Port

The output port performs the same functions as the input port, but in the reverse order. First the outgoing packets are queued, then the packet is encapsulated in a frame, and finally the physical-layer functions are applied to the frame to create the signal to be sent on the line. Figure 8.23 shows a schematic diagram of an output port.

Figure 8.23 *Output port*



Routing Processor

The routing processor performs the functions of the network layer. The destination address is used to find the address of the next hop and, at the same time, the output port number from which the packet is sent out. This activity is sometimes referred to as **table lookup** because the routing processor searches the routing table. In the newer packet switches, this function of the routing processor is being moved to the input ports to facilitate and expedite the process.

Switching Fabrics

The most difficult task in a packet switch is to move the packet from the input queue to the output queue. The speed with which this is done affects the size of the input/output queue and the overall delay in packet delivery. In the past, when a packet switch was actually a dedicated computer, the memory of the computer or a bus was used as the switching fabric. The input port stored the packet in memory; the output port retrieved the packet from memory. Today, packet switches are specialized mechanisms that use a variety of switching fabrics. We briefly discuss some of these fabrics here.

Crossbar Switch

The simplest type of switching fabric is the crossbar switch, discussed in the previous section.

Banyan Switch

A more realistic approach than the crossbar switch is the **banyan switch** (named after the banyan tree). A banyan switch is a multistage switch with microswitches at each stage that route the packets based on the output port represented as a binary string. For n inputs and n outputs, we have $\log_2 n$ stages with $n/2$ microswitches at each stage. The first stage routes the packet based on the high-order bit of the binary string. The second stage routes the packet based on the second high-order bit, and so on. Figure 8.24 shows a banyan switch with eight inputs and eight outputs. The number of stages is $\log_2(8) = 3$.

Figure 8.24 *A banyan switch*

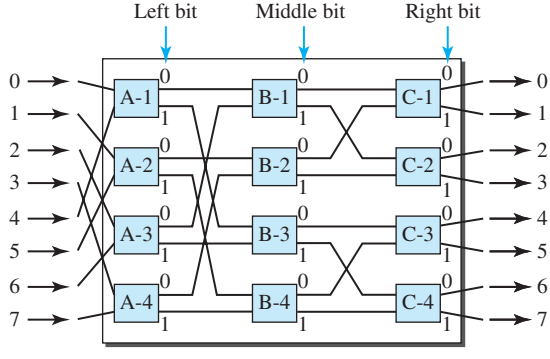
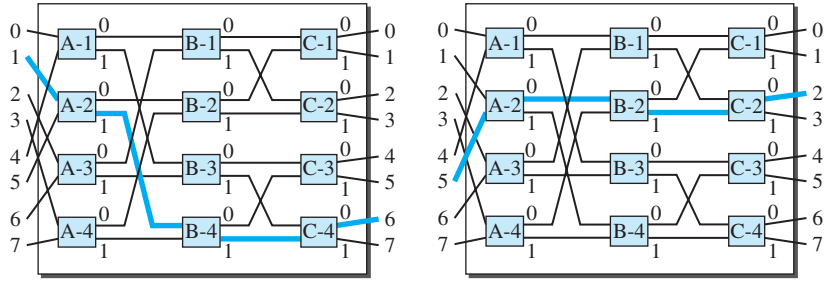


Figure 8.25 shows the operation. In part a, a packet has arrived at input port 1 and must go to output port 6 (110 in binary). The first microswitch (A-2) routes the packet

Figure 8.25 *Examples of routing in a banyan switch*



a. Input 1 sending a cell to output 6 (110)

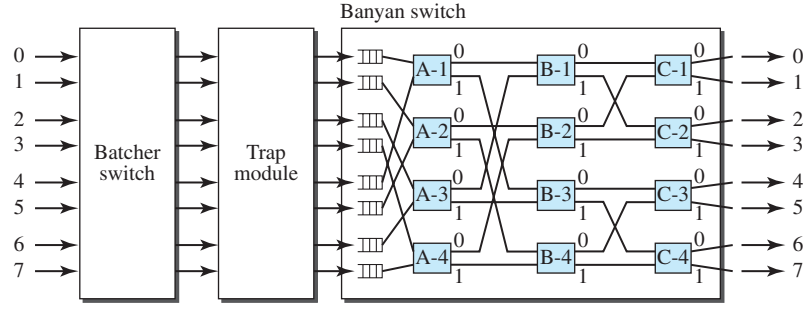
b. Input 5 sending a cell to output 2 (010)

based on the first bit (1), the second microswitch (B-4) routes the packet based on the second bit (1), and the third microswitch (C-4) routes the packet based on the third bit (0). In part b, a packet has arrived at input port 5 and must go to output port 2 (010 in binary). The first microswitch (A-2) routes the packet based on the first bit (0), the second microswitch (B-2) routes the packet based on the second bit (1), and the third microswitch (C-2) routes the packet based on the third bit (0).

Batcher-Banyan Switch The problem with the banyan switch is the possibility of internal collision even when two packets are not heading for the same output port. We can solve this problem by sorting the arriving packets based on their destination port.

K. E. Batcher designed a switch that comes before the banyan switch and sorts the incoming packets according to their final destinations. The combination is called the **Batcher-banyan switch.** The sorting switch uses hardware merging techniques, but we do not discuss the details here. Normally, another hardware module called a **trap** is added between the Batcher switch and the banyan switch (see Figure 8.26) The trap module prevents duplicate packets (the packets with the same output destination) from passing to the banyan switch simultaneously. Only one packet for each destination is allowed at each tick; if there is more than one, they wait for the next tick.

Figure 8.26 *Batcher-banyan switch*



8.5 END-CHAPTER MATERIALS

8.5.1 Recommended Reading

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Switching is discussed in [Sta04] and [GW04]. Circuit-switching is fully discussed in [BEL01].

8.5.2 Key terms

banyan switch
Batcher-banyan switch
blocking
circuit switching
circuit-switched network
crossbar switch
crosspoint
data-transfer phase
datagram
datagram network
end system
input port
message switching
multistage switch
output port
packet switching
packet-switched network
routing processor
setup phase
space-division switching
switch
switching
switching fabric
table lookup
teardown phase
time-division switching
time-slot interchange (TSI)
time-space-time (TST) switch
trap
virtual-circuit identifier (VCI)
virtual-circuit network

8.5.3 Summary

A switched network consists of a series of interlinked nodes, called *switches*. Traditionally, three methods of switching have been important: circuit switching, packet switching, and message switching.

We can divide today's networks into three broad categories: circuit-switched networks, packet-switched networks, and message-switched networks. Packet-switched networks can also be divided into two subcategories: virtual-circuit networks and datagram networks. A circuit-switched network is made of a set of switches connected by physical links, in which each link is divided into n channels. Circuit switching takes place at the physical layer. In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of the data-transfer phase until the teardown phase.

In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand. In a datagram network, each packet is treated independently of all others. Packets in this approach are referred to as datagrams. There are no setup or teardown phases. A virtual-circuit network is a cross between a

circuit-switched network and a datagram network. It has some characteristics of both. Circuit switching uses either of two technologies: the space-division switch or the time-division switch. A switch in a packet-switched network has a different structure from a switch used in a circuit-switched network. We can say that a packet switch has four types of components: input ports, output ports, a routing processor, and switching fabric.

8.6 PRACTICE SET

8.6.1 Quizzes

A set of interactive quizzes for this chapter can be found on the book website. It is strongly recommended that the student take the quizzes to check his/her understanding of the materials before continuing with the practice set.

8.6.2 Questions

Q8-1. Describe the need for switching and define a switch.

Q8-2. List the three traditional switching methods. Which are the most common today?

Q8-3. What are the two approaches to packet switching?

Q8-4. Compare and contrast a circuit-switched network and a packet-switched network.

Q8-5. What is the role of the address field in a packet traveling through a datagram network?

Q8-6. What is the role of the address field in a packet traveling through a virtual-circuit network?

Q8-7. Compare space-division and time-division switches.

Q8-8. What is TSI and what is its role in time-division switching?

Q8-9. Compare and contrast the two major categories of circuit switches.

Q8-10. List four major components of a packet switch and their functions.

8.6.3 Problems

P8-1. A path in a digital circuit-switched network has a data rate of 1 Mbps. The exchange of 1000 bits is required for the setup and teardown phases. The distance between two parties is 5000 km. Answer the following questions if the propagation speed is 2×10^8 m:

- **a.** What is the total delay if 1000 bits of data are exchanged during the data-transfer phase?
- **b.** What is the total delay if 100,000 bits of data are exchanged during the data-transfer phase?
- **c.** What is the total delay if 1,000,000 bits of data are exchanged during the data-transfer phase?
- **d.** Find the delay per 1000 bits of data for each of the above cases and compare them. What can you infer?

P8-2. Five equal-size datagrams belonging to the same message leave for the destination one after another. However, they travel through different paths as shown in Table 8.1.

Table 8.1 *P8-2*

Datagram	*Path Length*	*Visited Switches*
1	3200 km	1, 3, 5
2	11,700 km	1, 2, 5
3	12,200 km	1, 2, 3, 5
4	10,200 km	1, 4, 5
5	10,700 km	1, 4, 3, 5

We assume that the delay for each switch (including waiting and processing) is 3, 10, 20, 7, and 20 ms respectively. Assuming that the propagation speed is 2×10^8 m, find the order the datagrams arrive at the destination and the delay for each. Ignore any other delays in transmission.

P8-3. Transmission of information in any network involves end-to-end addressing and sometimes local addressing (such as VCI). Table 8.2 shows the types of networks and the addressing mechanism used in each of them.

Table 8.2 *P8-3*

Network	*Setup*	*Data Transfer*	*Teardown*
Circuit-switched	End-to-end		End-to-end
Datagram		End-to-end	
Virtual-circuit	End-to-end	Local	End-to-end

Answer the following questions:

a. Why does a circuit-switched network need end-to-end addressing during the setup and teardown phases? Why are no addresses needed during the data transfer phase for this type of network?

b. Why does a datagram network need only end-to-end addressing during the data transfer phase, but no addressing during the setup and teardown phases?

c. Why does a virtual-circuit network need addresses during all three phases?

P8-4. We mentioned that two types of networks, datagram and virtual-circuit, need a routing or switching table to find the output port from which the information belonging to a destination should be sent out, but a circuit-switched network has no need for such a table. Give the reason for this difference.

P8-5. An entry in the switching table of a virtual-circuit network is normally created during the setup phase and deleted during the teardown phase. In other words, the entries in this type of network reflect the current connections, the activity in the network. In contrast, the entries in a routing table of a datagram network do not depend on the current connections; they show the configuration of the network and how any packet should be routed to a final destination. The entries may remain the same even if there is no activity in the network. The routing tables, however, are updated if there are changes in the network. Can you explain the reason for these two different characteristics? Can we say that

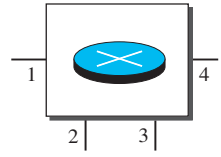
a virtual-circuit is a *connection-oriented* network and a datagram network is a *connectionless* network because of the above characteristics?

P8-6. The minimum number of columns in a datagram network is two; the minimum number of columns in a virtual-circuit network is four. Can you explain the reason? Is the difference related to the type of addresses carried in the packets of each network?

P8-7. Figure 8.27 shows a switch (router) in a datagram network.

Figure 8.27 *Problem P8-7*

Destination address	Output port
1233	3
1456	2
3255	1
4470	4
7176	2
8766	3
9144	2



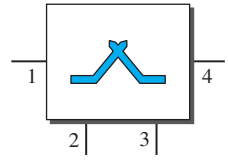
Find the output port for packets with the following destination addresses:

a. Packet 1: 7176
b. Packet 2: 1233
c. Packet 3: 8766
d. Packet 4: 9144

P8-8. Figure 8.28 shows a switch in a virtual-circuit network.

Figure 8.28 *Problem P8-8*

Incoming		Outgoing	
Port	VCI	Port	VCI
1	14	3	22
2	71	4	41
2	92	1	45
3	58	2	43
3	78	2	70
4	56	3	11



Find the output port and the output VCI for packets with the following input port and input VCI addresses:

a. Packet 1: 3, 78
b. Packet 2: 2, 92
c. Packet 3: 4, 56
d. Packet 4: 2, 71

P8-9. Answer the following questions:

a. Can a routing table in a datagram network have two entries with the same destination address? Explain.

b. Can a switching table in a virtual-circuit network have two entries with the same input port number? With the same output port number? With the same incoming VCIs? With the same outgoing VCIs? With the same incoming values (port, VCI)? With the same outgoing values (port, VCI)?

P8-10. It is obvious that a router or a switch needs to search to find information in the corresponding table. The searching in a routing table for a datagram network is based on the destination address; the searching in a switching table in a virtual-circuit network is based on the combination of incoming port and incoming VCI. Explain the reason and define how these tables must be ordered (sorted) based on these values.

P8-11. Consider an $n \times k$ crossbar switch with n inputs and k outputs.

- **a.** Can we say that the switch acts as a multiplexer if $n > k$?
- **b.** Can we say that the switch acts as a demultiplexer if $n < k$?

P8-12. We need a three-stage space-division switch with $N = 100$. We use 10 crossbars at the first and third stages and 4 crossbars at the middle stage.

- **a.** Draw the configuration diagram.
- **b.** Calculate the total number of crosspoints.
- **c.** Find the possible number of simultaneous connections.
- **d.** Find the possible number of simultaneous connections if we use a single crossbar (100×100).
- **e.** Find the blocking factor, the ratio of the number of connections in part c and in part d.

P8-13. Repeat Problem 8-12 if we use 6 crossbars at the middle stage.

P8-14. Redesign the configuration of Problem 8-12 using the Clos criteria.

P8-15. We need to have a space-division switch with 1000 inputs and outputs. What is the total number of crosspoints in each of the following cases?

- **a.** Using a single crossbar.
- **b.** Using a multi-stage switch based on the Clos criteria.

P8-16. We need a three-stage time-space-time switch with $N = 100$. We use 10 TSIs at the first and third stages and 4 crossbars at the middle stage.

- **a.** Draw the configuration diagram.
- **b.** Calculate the total number of crosspoints.
- **c.** Calculate the total number of memory locations we need for the TSIs.

8.7 SIMULATION EXPERIMENTS

8.7.1 Applets

We have created some Java applets to show some of the main concepts discussed in this chapter. It is strongly recommended that the students activate these applets on the book website and carefully examine the protocols in action.