

## UNIT- 2

### Physical Layer and Overview of PL Switching

#### 2.1 MULTIPLEXING

Multiplexing is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.

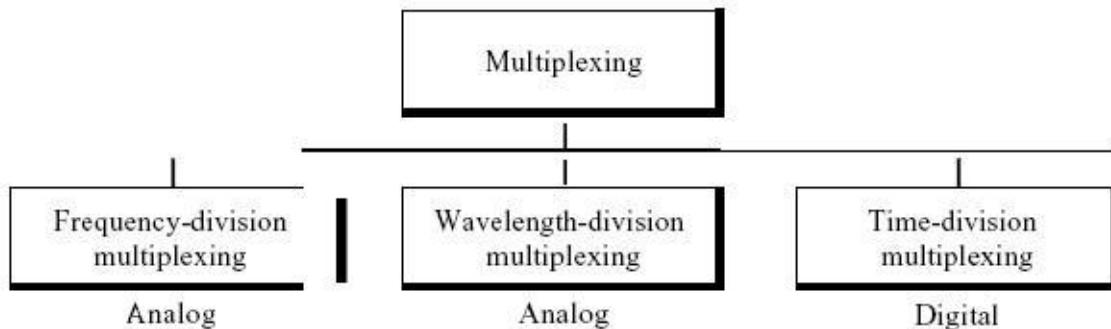
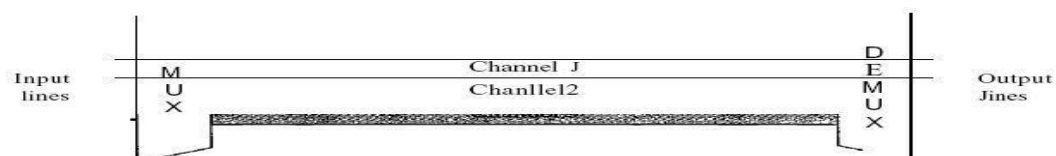


Figure *Categories of multiplexing*

##### 2.1.1 Frequency-Division Multiplexing

Frequency-division multiplexing (FDM) is an analog technique that can be applied when the bandwidth of a link (in hertz) is greater than the combined bandwidths of the signals to be transmitted. In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link. Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal. These bandwidth ranges are the channels through which the various signals travel. Channels can be separated by strips of unused bandwidth-guard bands-to prevent signals from overlapping. In addition, carrier frequencies must not interfere with the original data frequencies.

Figure 1 gives a conceptual view of FDM. In this illustration, the transmission path is divided into three parts, each representing a channel that carries one transmission.



**Fig: FDM**

We consider FDM to be an analog multiplexing technique; however, this does not mean that FDM cannot be used to combine sources sending digital signals. A digital signal can be converted to an analog signal before FDM is used to multiplex them.

### *Multiplexing Process*

Figure 2 is a conceptual illustration of the multiplexing process. Each source generates a signal of a similar frequency range. Inside the multiplexer, these similar signals modulates different carrier frequencies ( $f_1, f_2$ , and  $f_3$ ). The resulting modulated signals are then combined into a single composite signal that is sent out over a media link that has enough bandwidth to accommodate it.

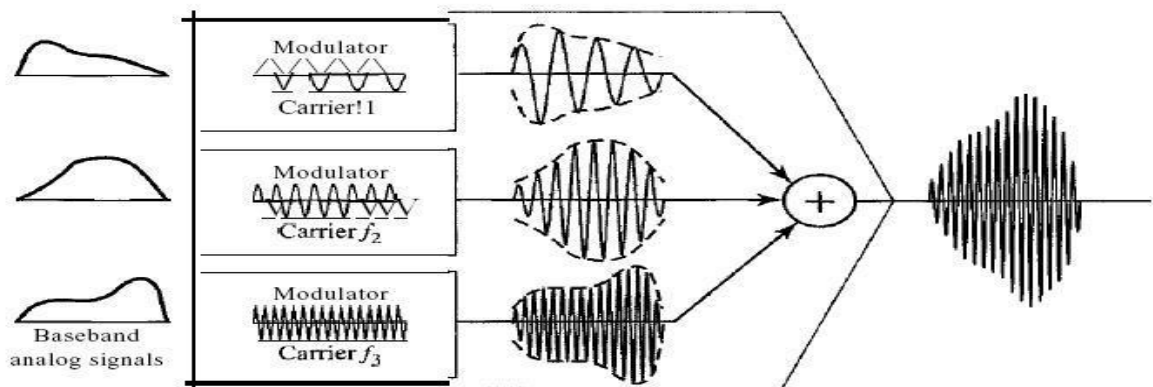


Figure 2 FDM process

### *Demultiplexing Process*

The demultiplexer uses a series of filters to decompose the multiplexed signal into its constituent component signals. The individual signals are then passed to a demodulator that separates them from their carriers and passes them to the output lines. Figure 3 is a conceptual illustration of demultiplexing process.

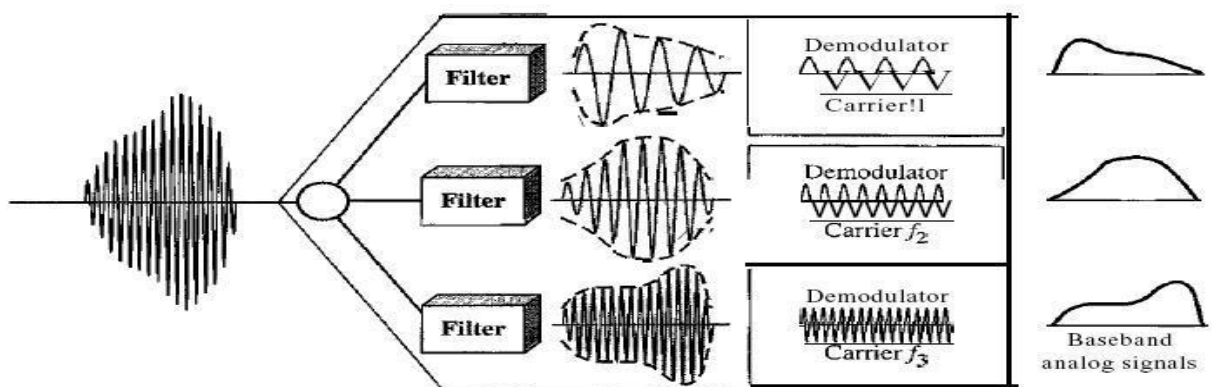


Figure 3 FDM de-multiplexing example

## **2.1.2 Wavelength-Division Multiplexing**

Wavelength-division multiplexing (WDM) is designed to use the high-data-rate capability of fiber-optic cable. The optical fiber data rate is higher than the data rate of metallic

transmission cable. Using a fiber-optic cable for one single line wastes the available bandwidth. Multiplexing allows us to combine several lines into one.

WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve optical signals transmitted through fiber-optic channels. The idea is the same: We are combining different signals of different frequencies. The difference is that the frequencies are very high.

Figure 4 gives a conceptual view of a WDM multiplexer and demultiplexer. Very narrow bands of light from different sources are combined to make a wider band of light. At the receiver, the signals are separated by the demultiplexer.

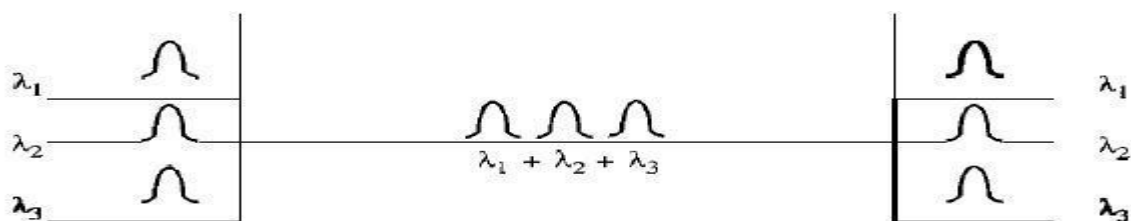


Figure 4 *Wavelength-division multiplexing*

Although WDM technology is very complex, the basic idea is very simple. We want to combine multiple light sources into one single light at the multiplexer and do the reverse at the demultiplexer. The combining and splitting of light sources are easily handled by a prism. Recall from basic physics that a prism bends a beam of light based on the angle of incidence and the frequency. Using this technique, a multiplexer can be made to combine several input beams of light, each containing a narrow band of frequencies, into one output beam of a wider band of frequencies. A demultiplexer can also be made to reverse the process. Figure 5 shows the concept.

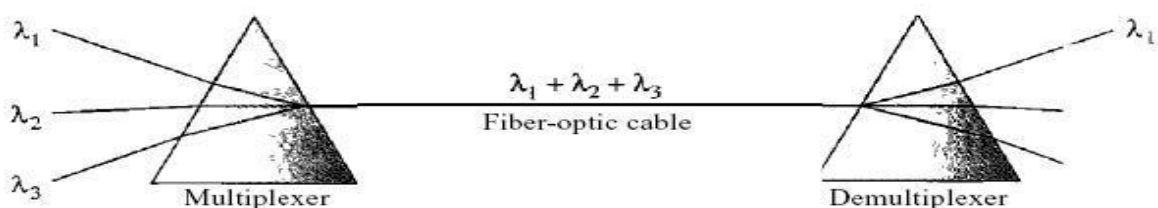


Figure 5 *Prisms in wavelength-division multiplexing and demultiplexing*

### 2.1.3 Synchronous Time-Division Multiplexing

Time-division multiplexing (TDM) is a digital process that allows several connections to share the high bandwidth of a line. Instead of sharing a portion of the bandwidth as in FDM, time is shared. Each connection occupies a portion of time in the link. Figure 6 gives a conceptual view of TDM. Note that the same link is used as in FDM; here, however, the link is shown sectioned by time rather than by frequency. In the figure, portions of signals 1,2,3,

and 4 occupy the link sequentially.

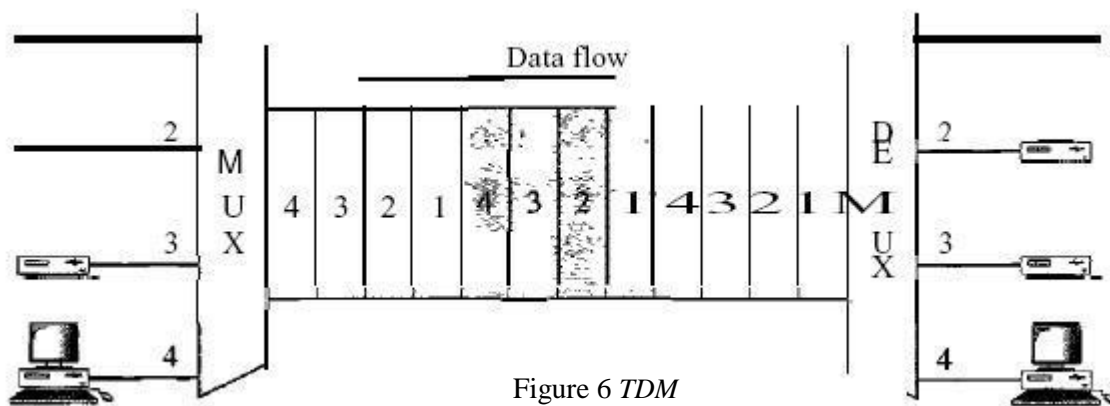


Figure 6 TDM

Note that in Figure 6 we are concerned with only multiplexing, not switching. This means that all the data in a message from source 1 always go to one specific destination, be it 1, 2, 3, or 4. The delivery is fixed and unvarying, unlike switching. We also need to remember that TDM is, in principle, a digital multiplexing technique. Digital data from different sources are combined into one timeshared link. However, this does not mean that the sources cannot produce analog data; analog data can be sampled, changed to digital data, and then multiplexed by using TDM.

We can divide TDM into two different schemes: synchronous and statistical.

In synchronous TDM, each input connection has an allotment in the output even if it is not sending data.

### Time Slots and Frames

In synchronous TDM, the data flow of each input connection is divided into units, where each input occupies one input time slot. A unit can be 1 bit, one character, or one block of data. Each input unit becomes one output unit and occupies one output time slot. However, the duration of an output time slot is  $n$  times shorter than the duration of an input time slot. If an input time slot is  $T$  s, the output time slot is  $T/n$  s, where  $n$  is the number of connections. In other words, a unit

in the output connection has a shorter duration; it travels faster. Figure 7 shows an example of synchronous TDM where  $n$  is 3.

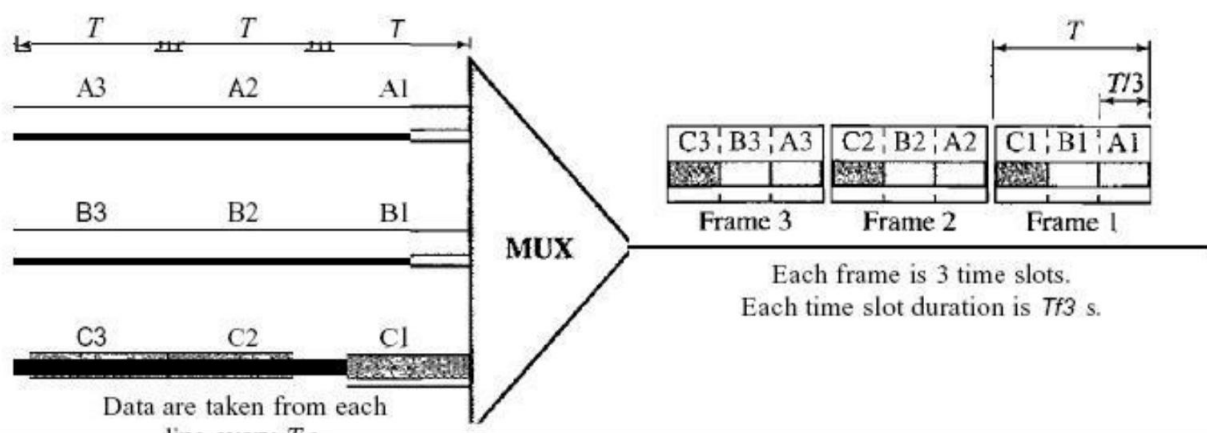


Figure 7 *Synchronous time-division multiplexing*

In synchronous TDM, a round of data units from each input connection is collected into a frame (we will see the reason for this shortly). If we have  $n$  connections, a frame is divided into  $n$  time slots and one slot is allocated for each unit, one for each input line. If the duration of the input unit is  $T$ , the duration of each slot is  $T/n$  and the duration of each frame is  $T$  (unless a frame carries some other information, as we will see shortly).

The data rate of the output link must be  $n$  times the data rate of a connection to guarantee the flow of data. In Figure 7, the data rate of the link is 3 times the data rate of a connection; likewise, the duration of a unit on a connection is 3 times that of the time slot (duration of a unit on the link). In the figure we represent the data prior to multiplexing as 3 times the size of the data after multiplexing. This is just to convey the idea that each unit is 3 times longer in duration before multiplexing than after. Time slots are grouped into frames. A frame consists of one complete cycle of time slots, with one slot dedicated to each sending device. In a system with  $n$  input lines, each frame has  $n$  slots, with each slot allocated to carrying data from a specific input line.

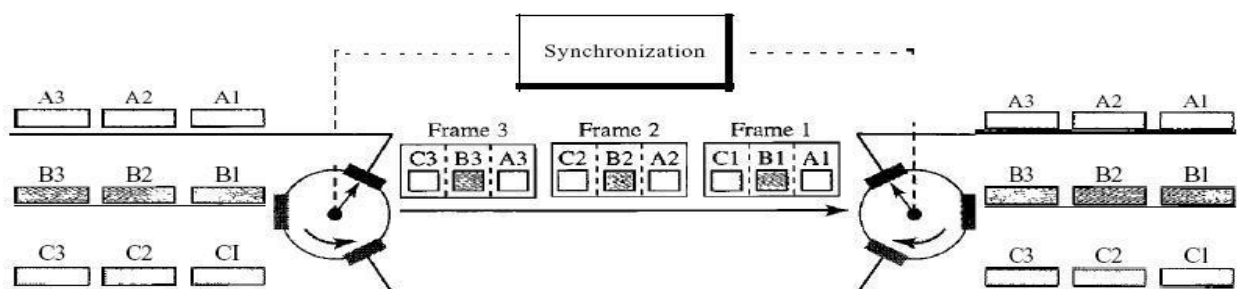
### *Interleaving*

TDM can be visualized as two fast-rotating switches, one on the multiplexing side and the other on the demultiplexing side. The switches are synchronized and rotate at the same speed, but in opposite directions. On the multiplexing side, as the switch opens in front of a connection, that connection has the opportunity to send a unit onto the path. This process is called **interleaving**.

On the demultiplexing side, as the switch opens in front of a connection, that connection has the opportunity to receive a unit from the path.

Figure 8 shows the interleaving process for the connection shown in Figure 7.

In this figure, we assume that no switching is involved and that the data from the first connection at the multiplexer site go to the first connection at the demultiplexer



**Figure 8** *Interleaving*

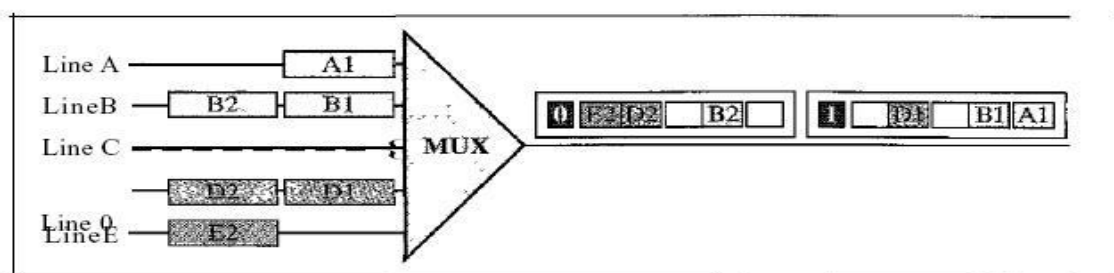
- *Empty Slots*
- *Data Rate Management*
  - **Multilevel Multiplexing**
  - **Multiple-Slot Allocation**
  - **Pulse Stuffing**
- *Frame Synchronizing*
- *Digital Signal Service*

## 2.1.4 Statistical Time-Division Multiplexing

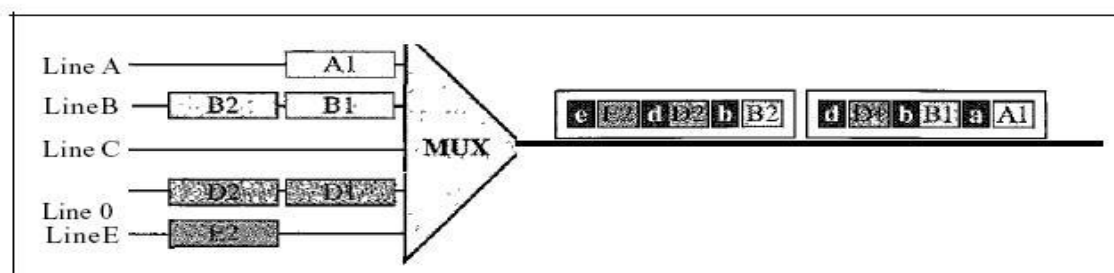
### *Addressing*

Figure a also shows a major difference between slots in synchronous TDM and statistical TDM. An output slot in synchronous TDM is totally occupied by data; in statistical TDM, a slot needs to carry data as well as the address of the destination.

In synchronous TDM, there is no need for addressing; synchronization and preassigned relationships between the inputs and outputs serve as an address. We know, for example, that input 1 always goes to input 2. If the multiplexer and the demultiplexer are synchronized, this is guaranteed. In statistical multiplexing, there is no fixed relationship between the inputs and outputs because there are no preassigned or reserved slots. We need to include the address of the receiver inside each slot to show where it is to be delivered. The addressing in its simplest form can be  $n$  bits to define  $N$  different output lines with  $n = \log_2 N$ . For example, for eight different output lines, we need a 3-bit address.



a. Synchronous TDM



b. Statistical TDM

Figure a TDM slot comparison

### *Slot Size*

Since a slot carries both data and an address in statistical TDM, the ratio of the data size to address size must be reasonable to make transmission efficient. For example, it would be inefficient to send 1 bit per slot as data when the address is 3 bits. This would mean an overhead of 300 percent. In statistical TDM, a block of data is usually many bytes while the address is just a few bytes.

### *No Synchronization Bit*

There is another difference between synchronous and statistical TDM, but this time it is at the frame level. The frames in statistical TDM need not be synchronized, so we do not need synchronization bits.

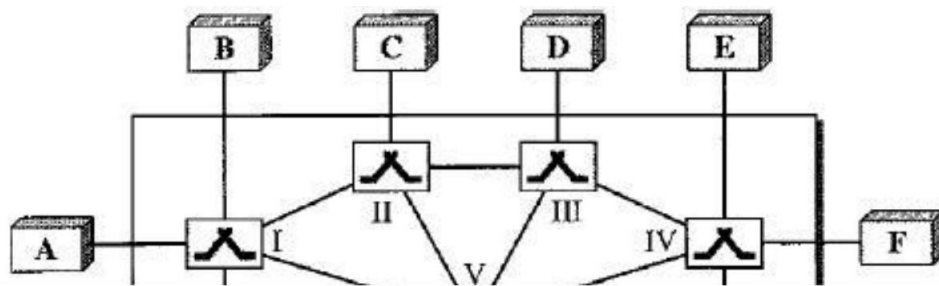
### *Bandwidth*

In statistical TDM, the capacity of the link is normally less than the sum of the capacities of each channel. The designers of statistical TDM define the capacity of the link based on the statistics of the load for each channel. If on average only  $x$  percent of the input slots are filled, the capacity of the link reflects this. Of course, during peak times, some slots need to wait.

## **2.2 Switching**

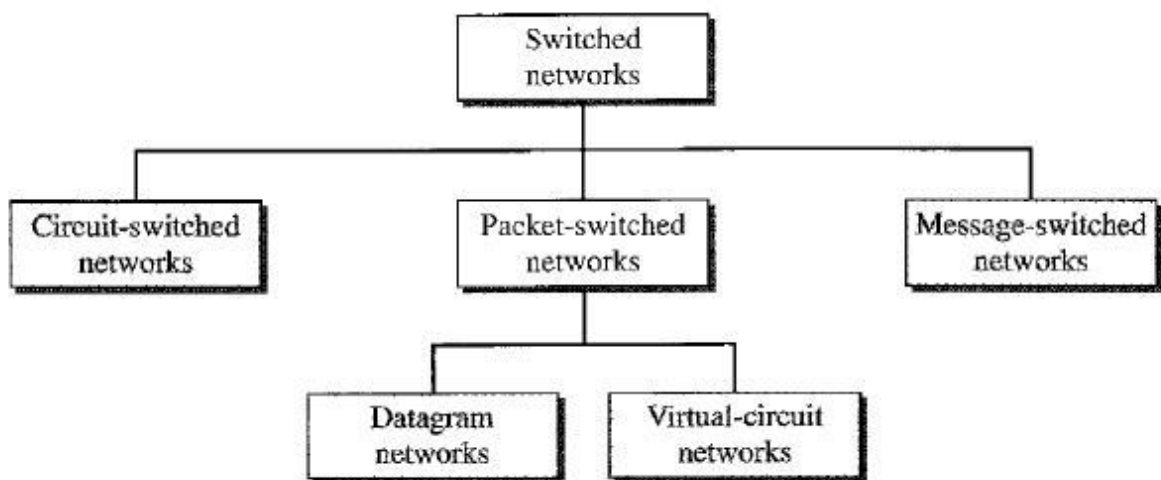
A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other 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 shows a 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.

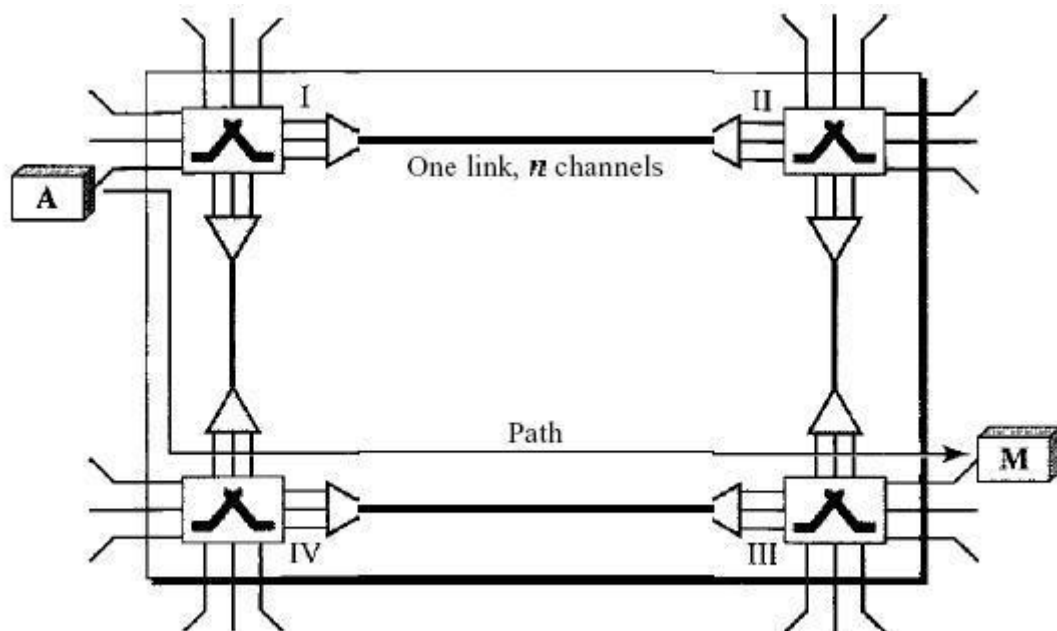
### *Taxonomy of switched networks*



### **2.2.1 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

Figure 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.



*Fig: A trivial circuit-switched network*

### 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 means creating dedicated channels between the switches. For example, in Figure, 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.

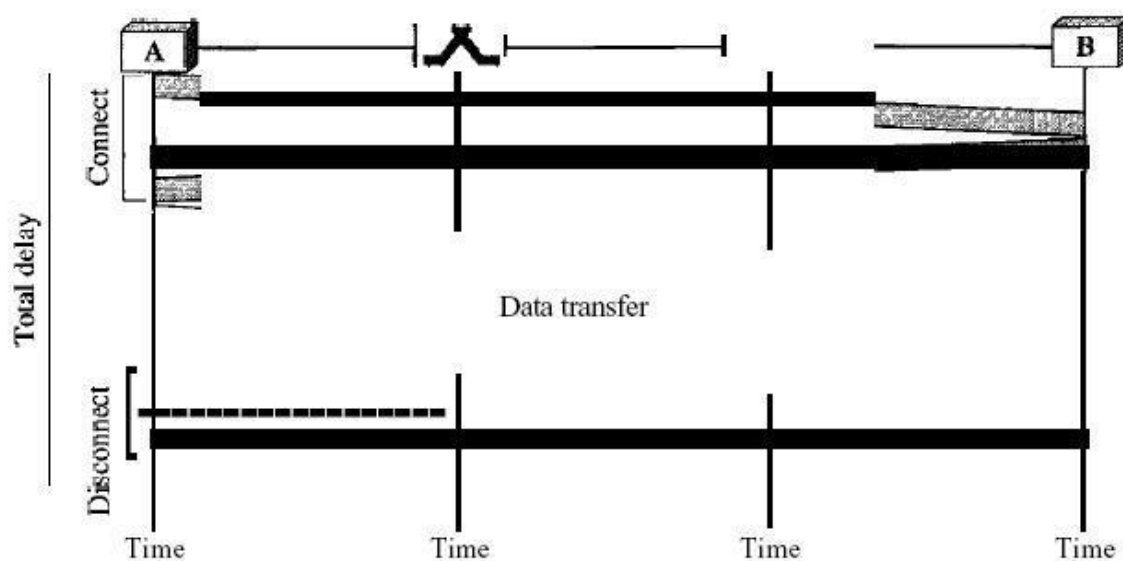
#### **Efficiency:**

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.

## 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. As Figure 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.



*Fig: Delay in a circuit-switched network*

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.

## 2.2.2 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. Figure 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.

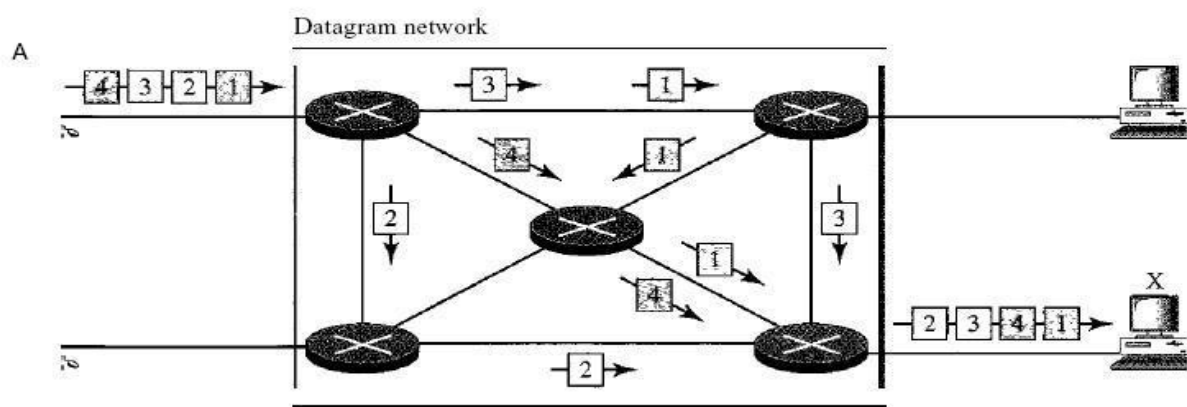


Fig: A datagram network with four switches (routers)

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 in which

each entry is created when the setup phase is completed and deleted when the teardown phase is over. Figure shows the routing table for a switch.

*Fig: Routing table in a datagram network*

### *Destination Address*

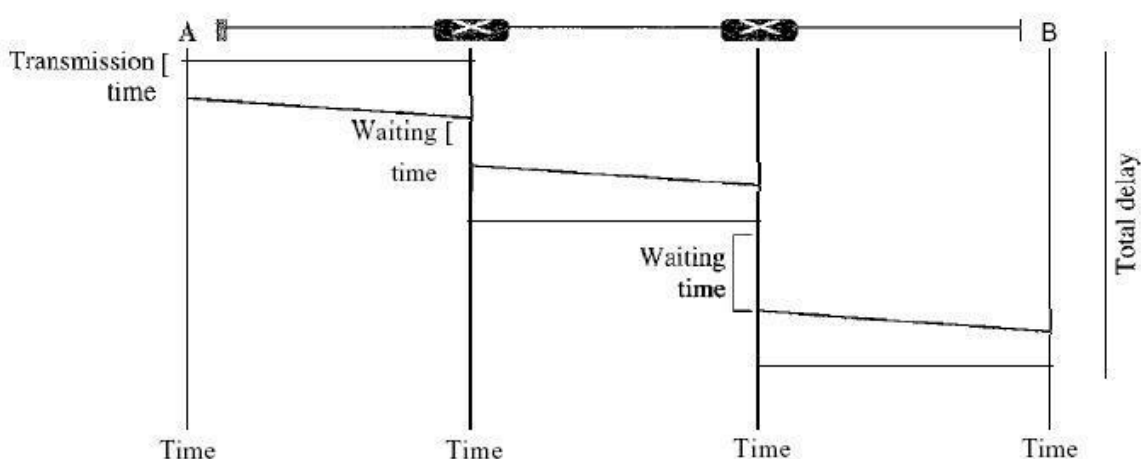
Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be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



*Fig: Delay in a datagram network*

The packet travels through two switches. There are three transmission times ( $3T$ ), three propagation delays (slopes  $3t$  of the lines), and two waiting times ( $W1 + w2$ ). We ignore the processing time in each switch. The total delay is

$$\text{Total delay} = 3T + 3t + W1 + W2$$

### 2.2.3 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 should be the next switch 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 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.

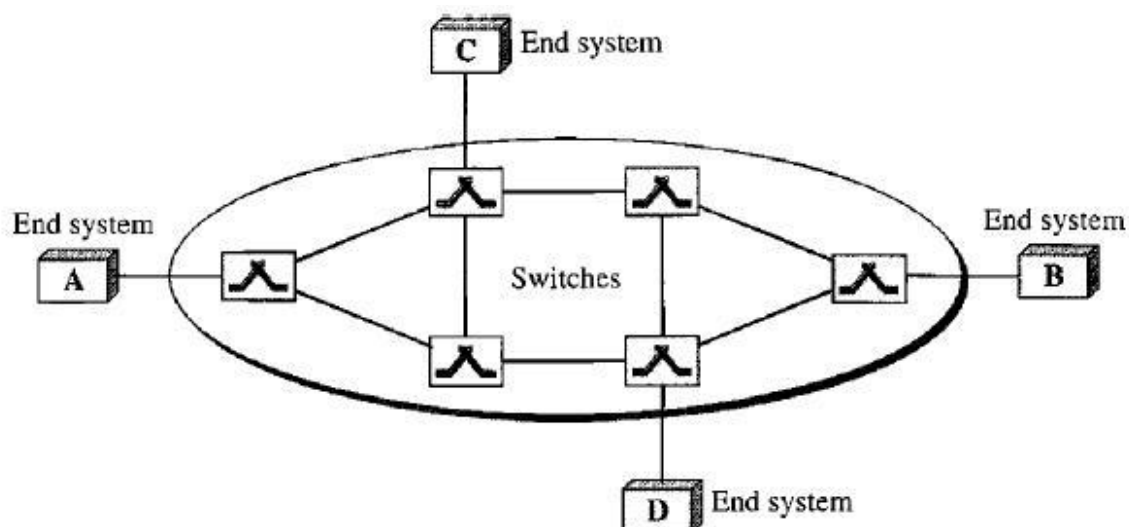


Fig a: *Virtual-circuit network*

## 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). 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.

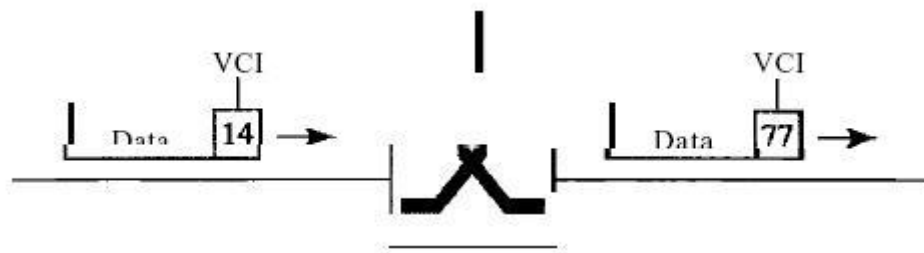


Figure 1 *Virtual-circuit identifier*

## 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 2 shows such a switch and its corresponding table. And also 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 3 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.

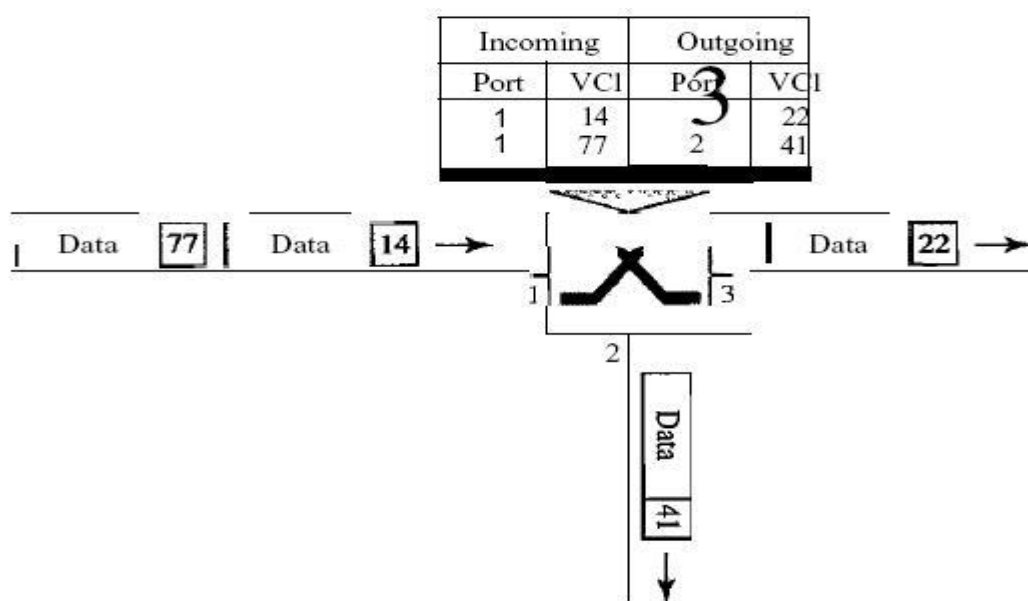


Figure 2 Switch and tables in a virtual-circuit network

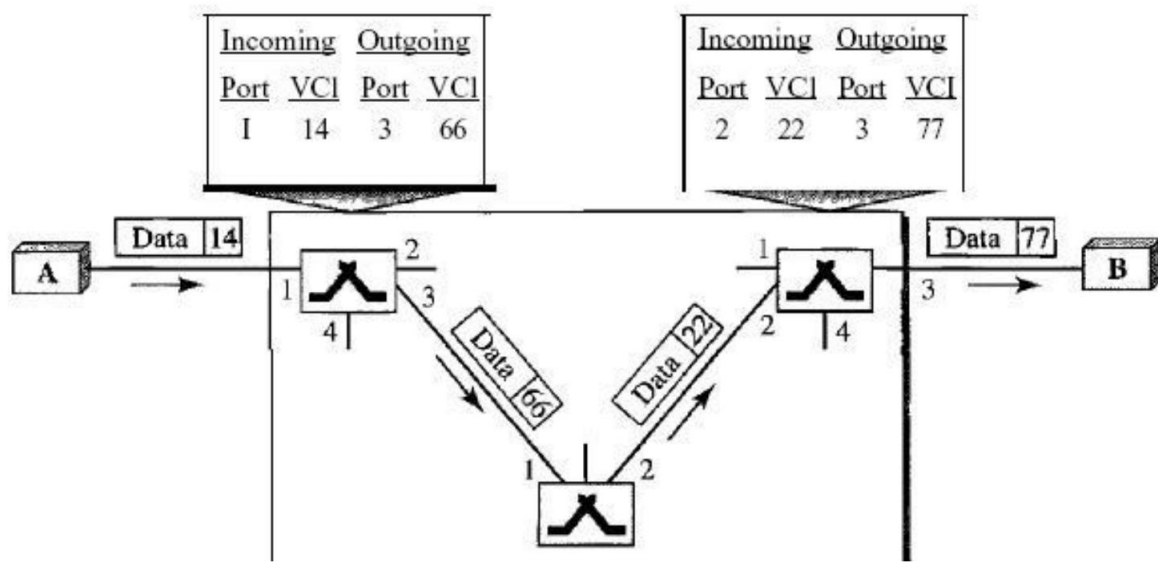
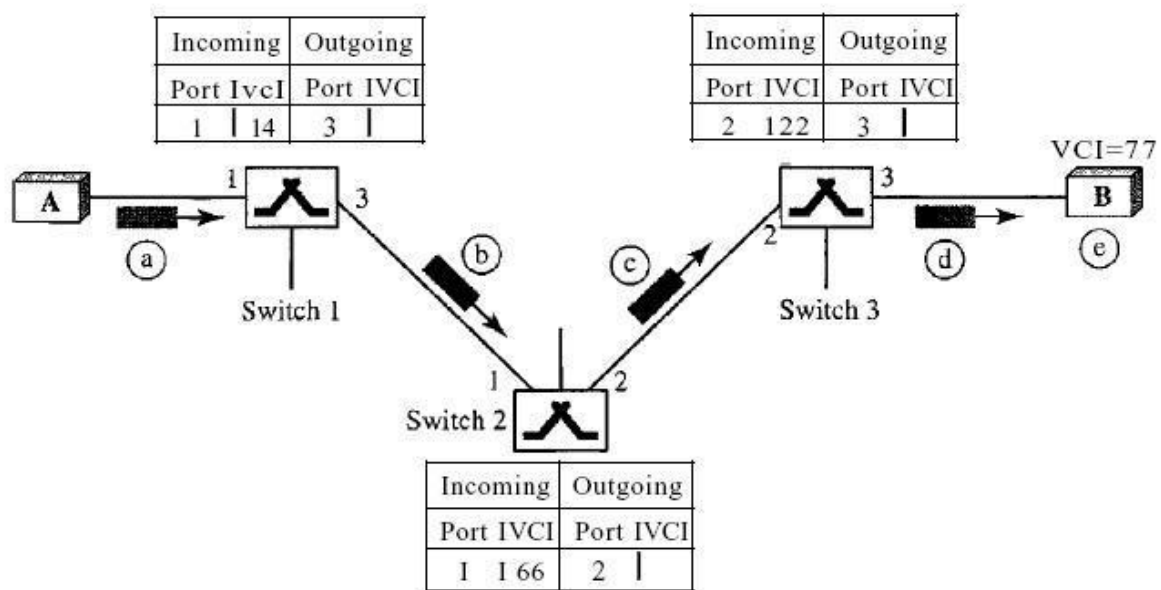


Figure 3 *Source-to-destination data transfer in a virtual-circuit network*

## Setup Request

A setup request frame is sent from the source to the destination.

Figure 4 shows the process.



**Figure 4** Setup request in a virtual-circuit network

- Source A sends a setup frame to switch 1.
- 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.
- 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).
- Switch 3 receives the setup request frame. Again, three columns are completed: incoming port (2), incoming VCI (22), and outgoing port (3).
- 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.

- 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.
- 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.
- 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.
- Finally switch 1 sends an acknowledgment to source A that contains its incoming VCI in the table, chosen in the previous step.
- The source uses this as the outgoing VCI for the data frames to be sent to destination B.

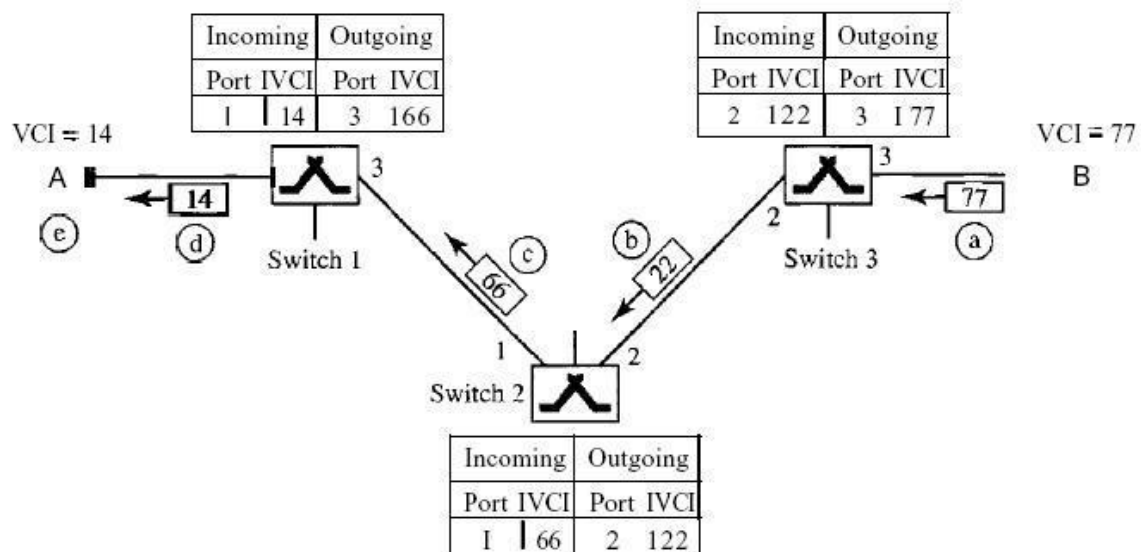


Figure 5 Setup acknowledgments in a virtual-circuit network

### Teardown Phase

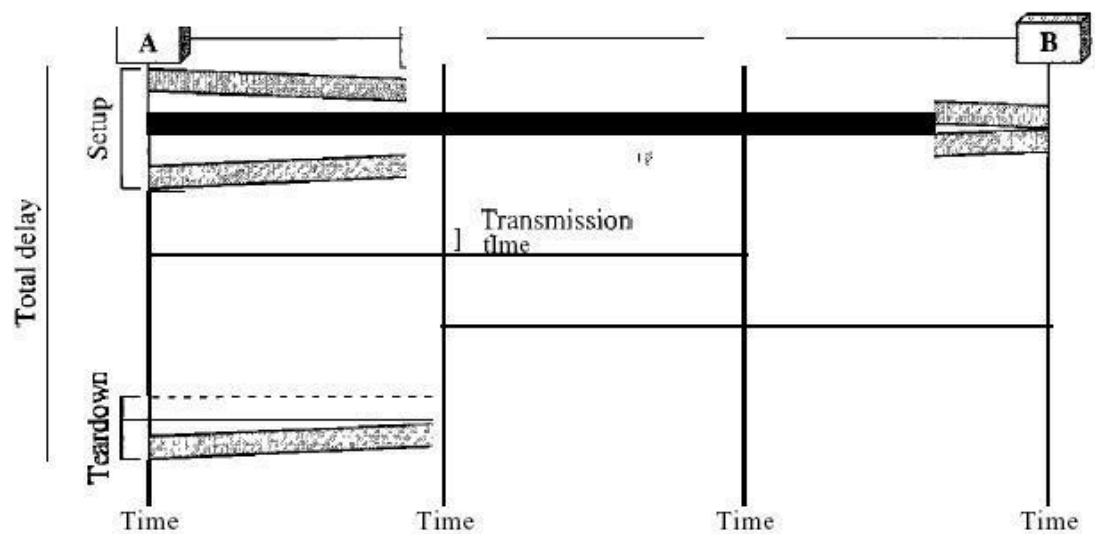
In this phase, source A, after sending all frames to B, sends a special frame called a *teardown request*. Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

## 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.

## 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. Below Figure shows the delay for a packet traveling through two switches in a virtual-circuit network.



*Fig: Delay in a virtual-circuit network*

The packet is traveling through two switches (routers). There are three transmission times ( $3T$ ), three propagation times ( $3t$ ), 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

$$\text{Total delay} = 3T + 3t + \text{setup delay} + \text{teardown delay}$$