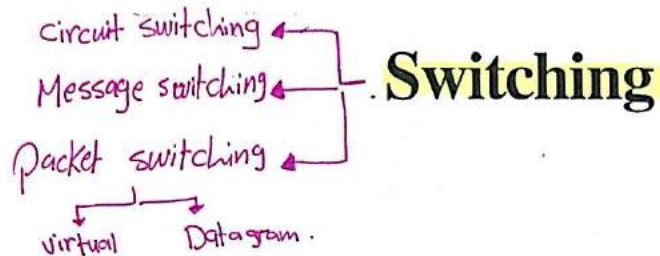


CHAPTER 8



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

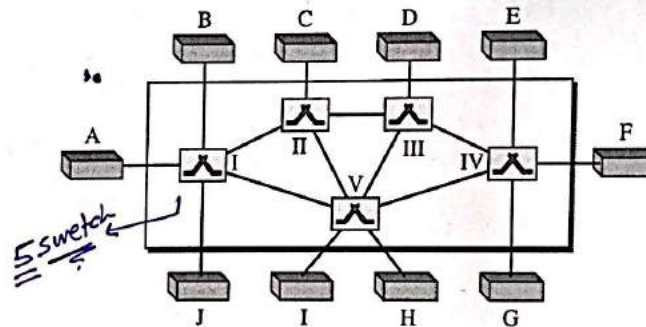
A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other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

star and mesh

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.

star topology
تربوئی
> star
Bus
Complexity > cost
mesh

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.

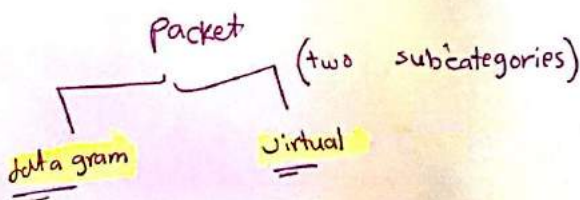
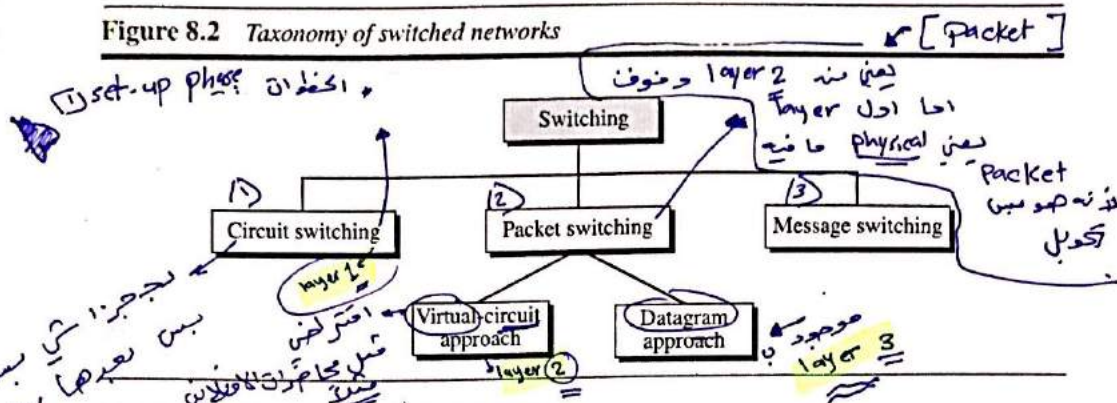


Figure 8.2 Taxonomy of switched networks



8.1.2 Switching and TCP/IP Layers

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

Switching at Physical Layer → (circuit switch)

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 → (virtual-circuit)

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 → (virtual or Datagram)

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 → (message switching)

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.

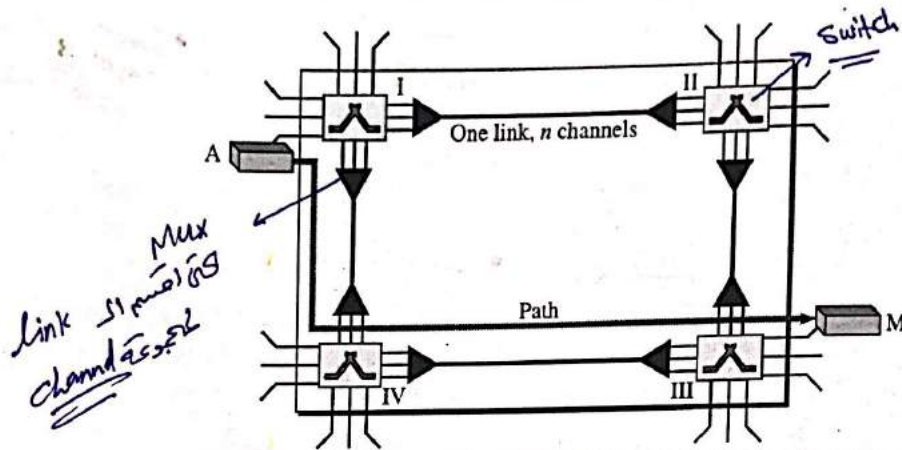
8.2 CIRCUIT-SWITCHED NETWORKS

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.

Handwritten note: accept reservation

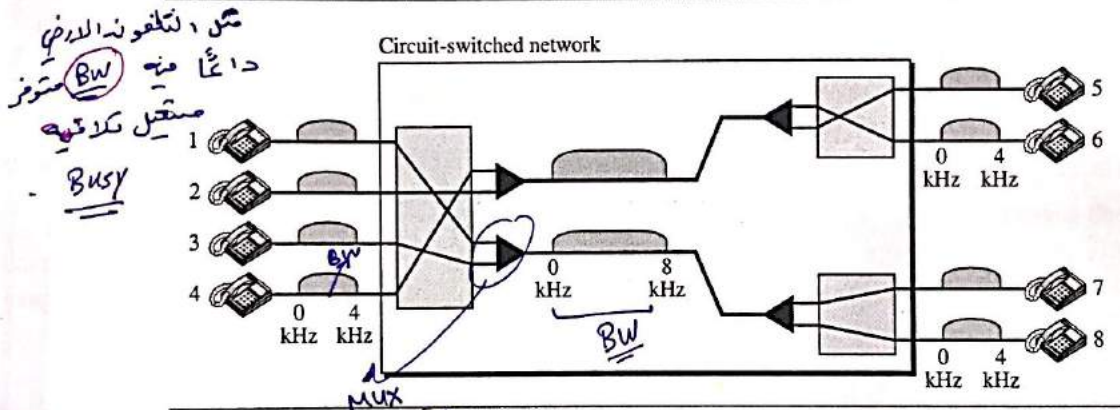
- There is no addressing involved during data transfer. The switches route the data based on their occupied band (FDM) or time slot (TDM). Of course, there is end-to-end addressing used during the setup phase, as we will see shortly.

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 (8) 2 Channel

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.

1) Setup Phase

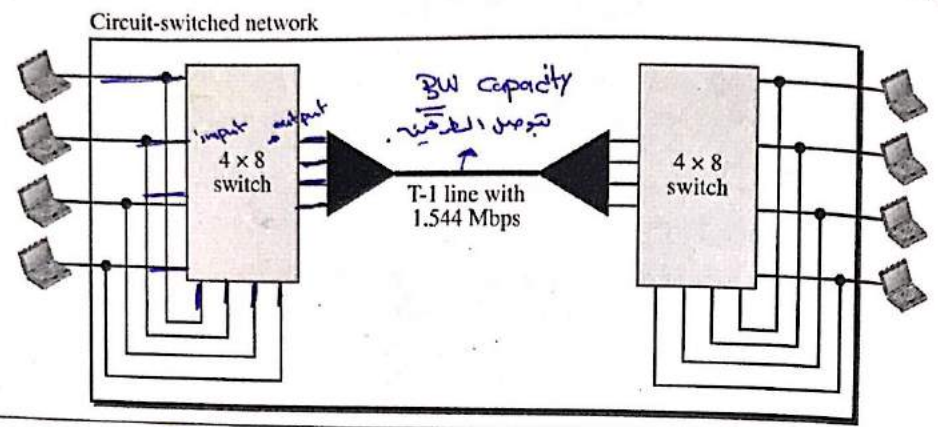
Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup

[dedicated link between the switch]

* المسج بال Circuit switch بتحويل حركته لانه انا ما قطعنا

212 PART II PHYSICAL LAYER
 * كما يجرى في Datagram او الحزم يعني Packet switch بصير سيرة
 Switching [out of order]

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.

2) Data-Transfer Phase

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

3) 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.

Delay :-

1) Propagation Delay = $\frac{\text{Distance}}{\text{Propagation speed}}$

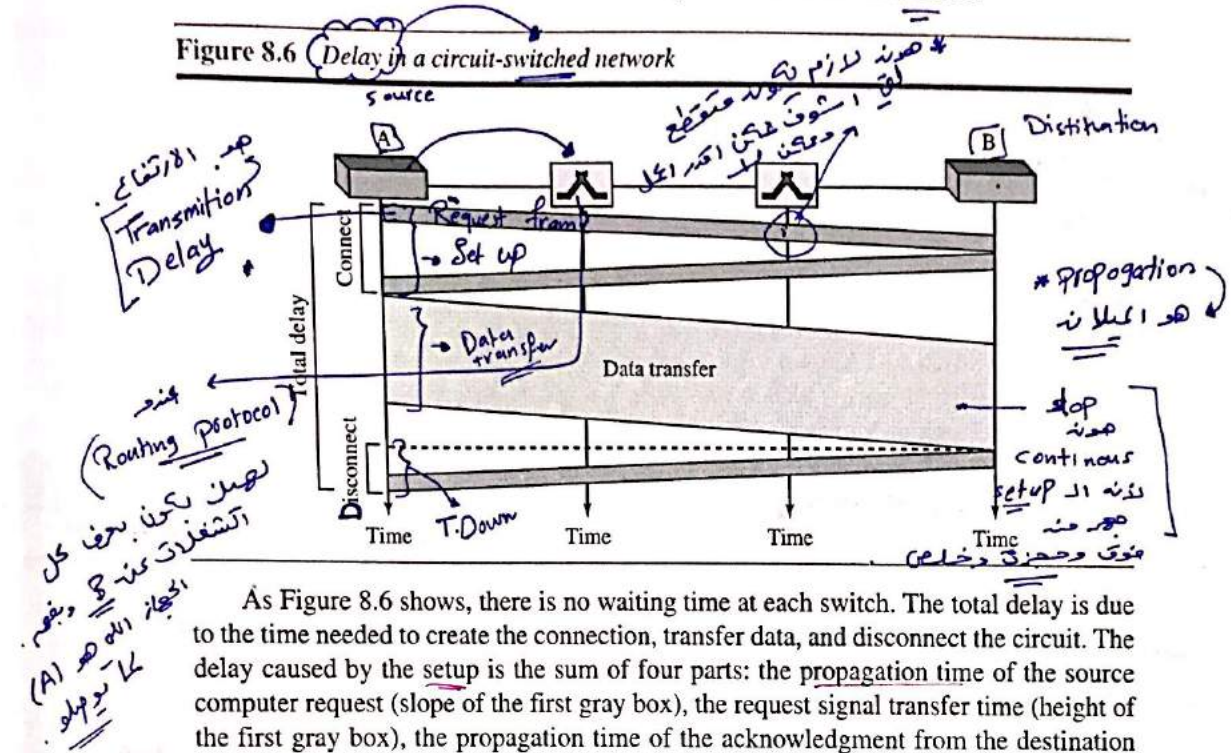
2) Transmission Delay = $\frac{\text{Message size}}{\text{Trans speed}}$

3) Queue Delay =

8.2.3 Delay (uniform) 4) Processing 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

في اكثر منه Path
عكس انه اعني فيه
وقل واحد
انه distance
عكس تكون فيه Queue

* ممكن في طريق يكون مغلق هو في يكون
فيه عند تاخير طبعاً

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.

في layer 3
Router
مشی صلیبی
Switch

8.3.1 Datagram Networks

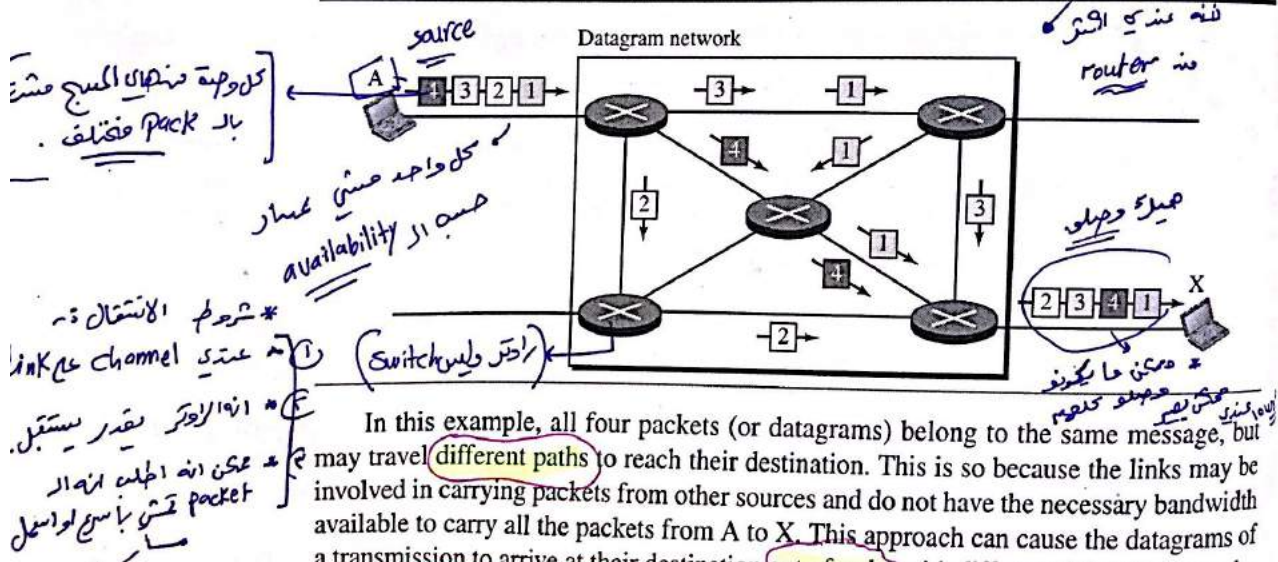
In a **datagram network**, each packet is treated independently of all others. Even if a packet is part of a multipacket transmission, the network treats it as though it existed alone. Packets in this approach are referred to as **datagrams**.

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.

Layer 3 مبدع روترانا ب 3 یا

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 :-

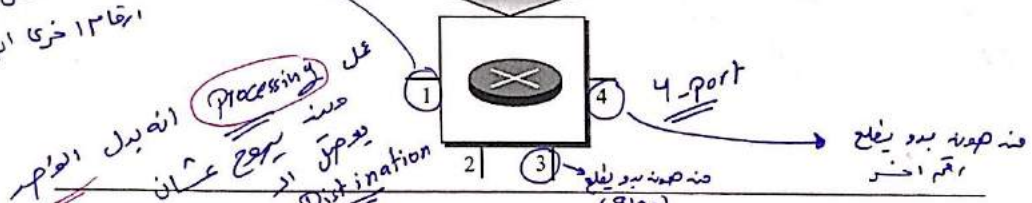
If there are no setup or teardown phases, how are the packets routed to their destinations in a datagram network? In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination 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.

لذا نحن نحتاج
 - setup
 - Routing Table
 - Destination address

Figure 8.8 Routing table in a datagram network

Destination address	Output port
1232	1
4150	2
⋮	⋮
9130	3

منه صوره بيو ريلاج 1234 ولما نيل ريلاج
 ارقام اخرى ايضا



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 :-

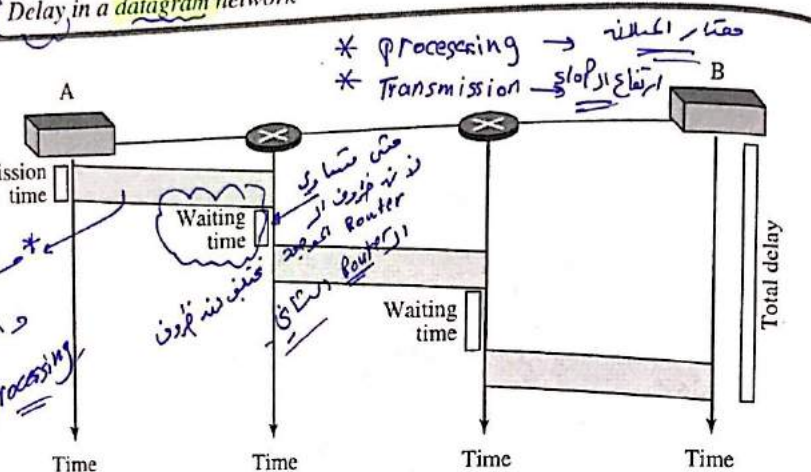
The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred. If a source sends a packet and there is a delay of a few minutes before another packet can be sent, the resources can be reallocated during these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

Circuit Delay
 * Setup
 * Teardown
 * Data Transfer
 * Propagation
 * Queue + Processing



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

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

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.

- As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
- Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
- 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.
- As in a circuit-switched network, all packets follow the same path established during the connection.

* Reservation
 * Queue
 * Processing



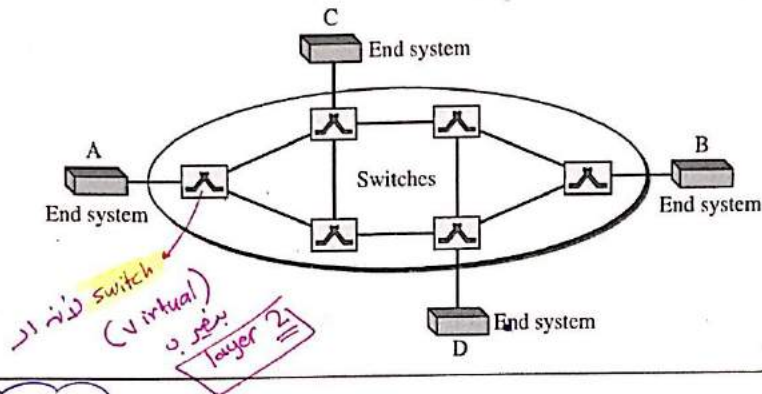
...
 * Delay
 ...

صحة التي تستخدم في الانترنت
 انه ليس (virtual) عن موضوع او
 Diagram ولكن نطرح
order

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).

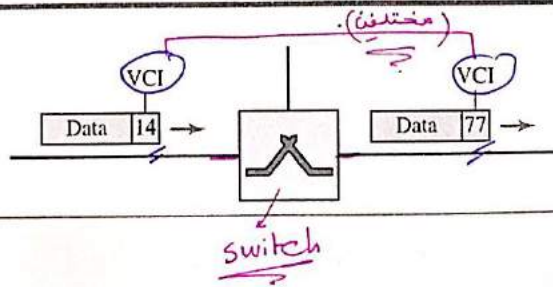
Global Addressing

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



(Label)
 كما ان (frame) يوصل الى switch يكون له (VCI) كما يطرح يكون له (VCI) مختلف

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

لازم كل switch ان يملك اوسه (table entry)

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

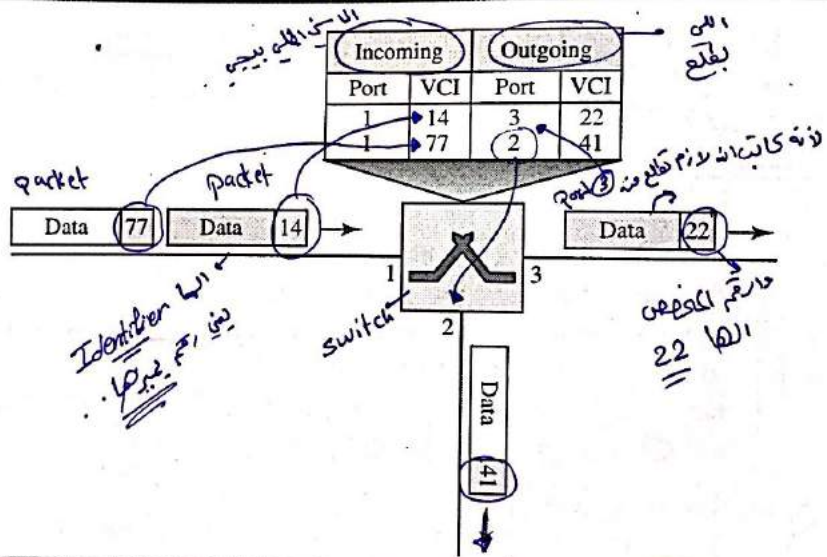


Figure 8.12 shows a frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3.

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.

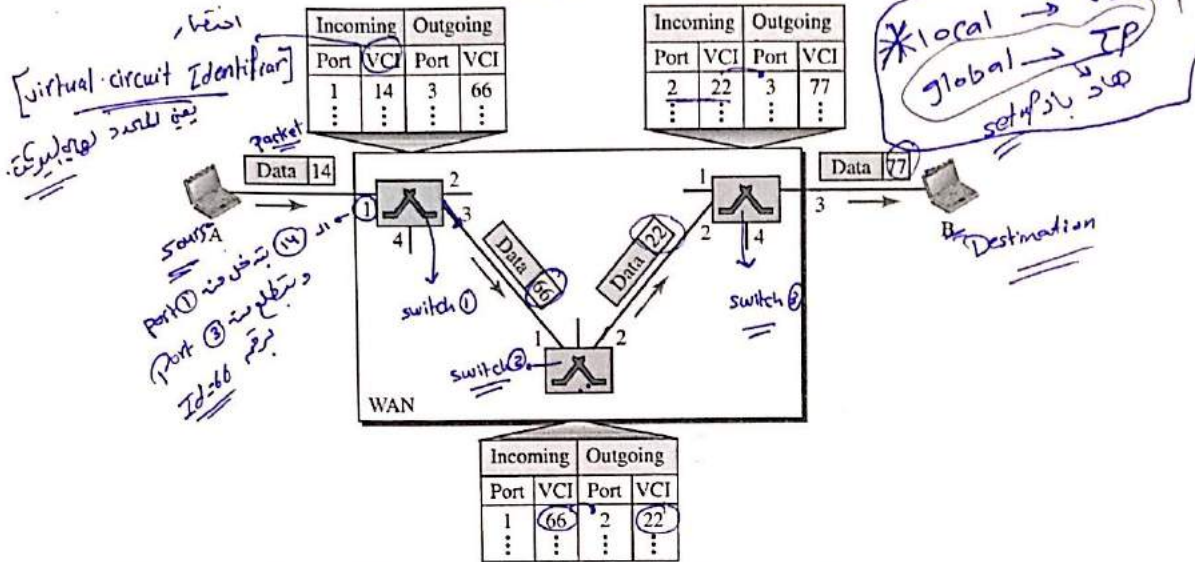
Setup Phase

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.

2

setup يعني طلب الخدمة الثانية التي بعد الـ setup

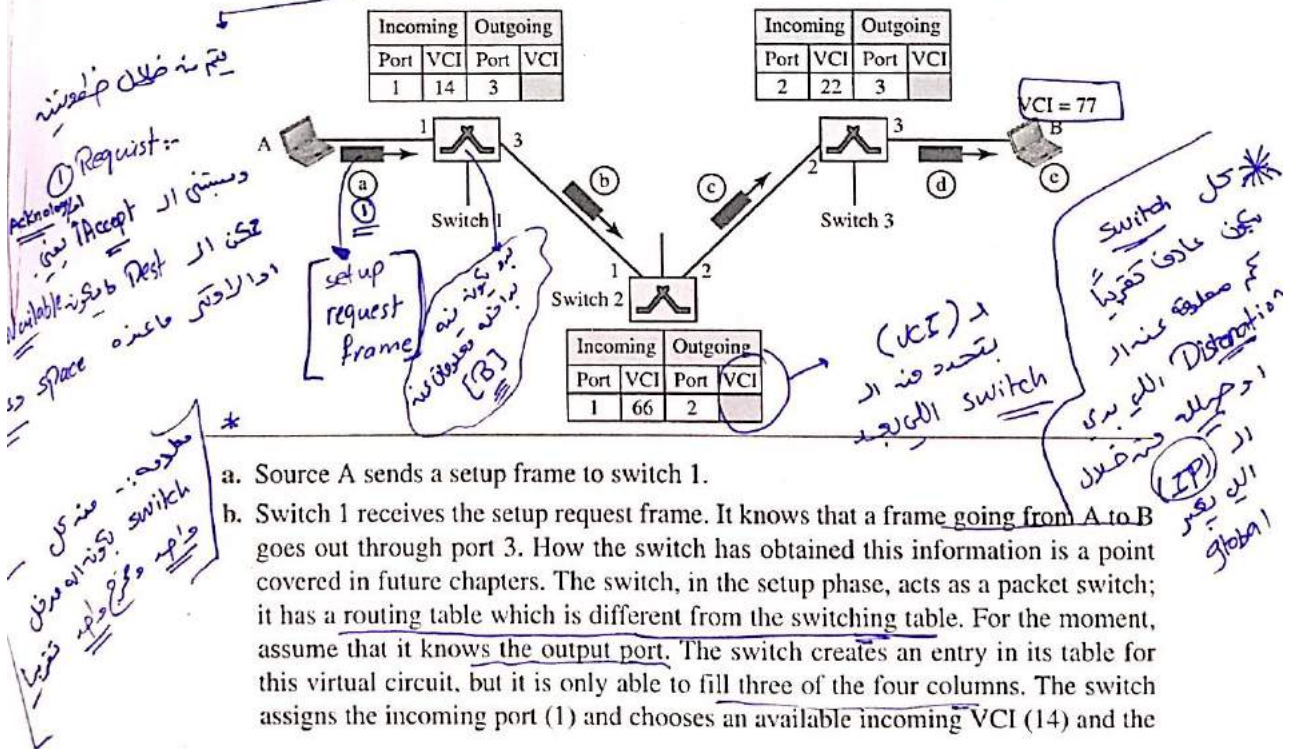
Figure 8.13 Source-to-destination data transfer in a virtual-circuit network



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



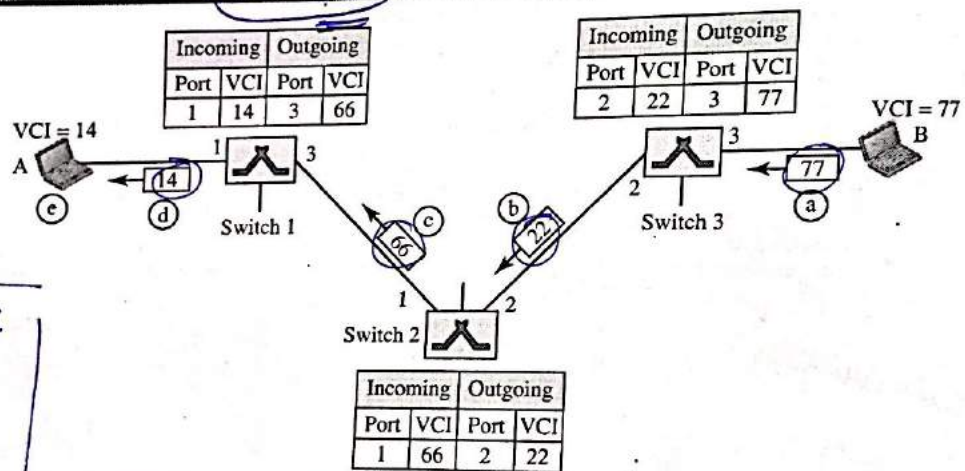
- a. Source A sends a setup frame to switch 1.
- b. Switch 1 receives the setup request frame. It knows that a frame going from A to B goes out through port 3. How the switch has obtained this information is a point covered in future chapters. The switch, in the setup phase, acts as a packet switch; 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.
- 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.

Figure 8.15 Setup acknowledgment in a virtual-circuit network



digit 32 = IP
 في جدول الـ VCI
 الـ IP الـ 32

- 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.

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

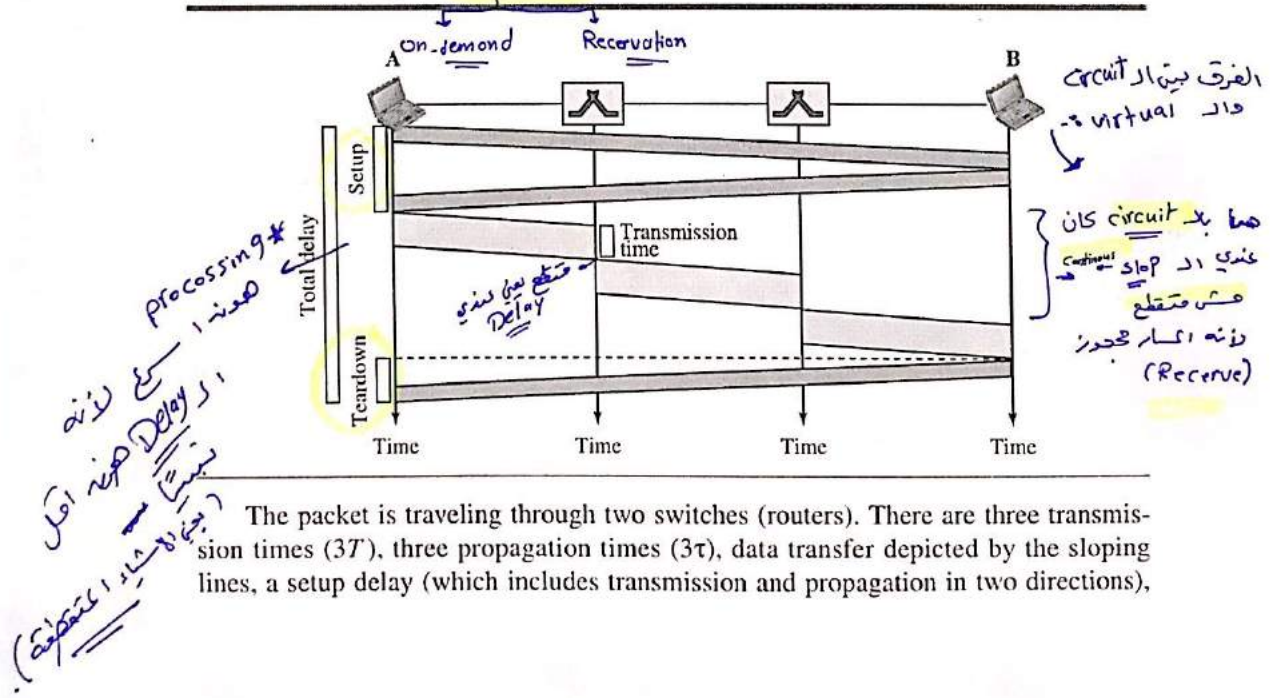
As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data-transfer phase. In the first case, the delay for each packet is the same; in the second case, each packet may encounter different delays. There is one big advantage in 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),

CHAPTER 9

Introduction to Data-Link Layer

* MAC Address → Hardware address
ما يتغير (ثابت)

The TCP/IP protocol suite does not define any protocol in the data-link layer or physical layer. These two layers are territories of networks that when connected make up the Internet. These networks, wired or wireless, provide services to the upper three layers of the TCP/IP suite. This may give us a clue that there are several standard protocols in the market today. For this reason, we discuss the data-link layer in several chapters. This chapter is an introduction that gives the general idea and common issues in the data-link layer that relate to all networks.

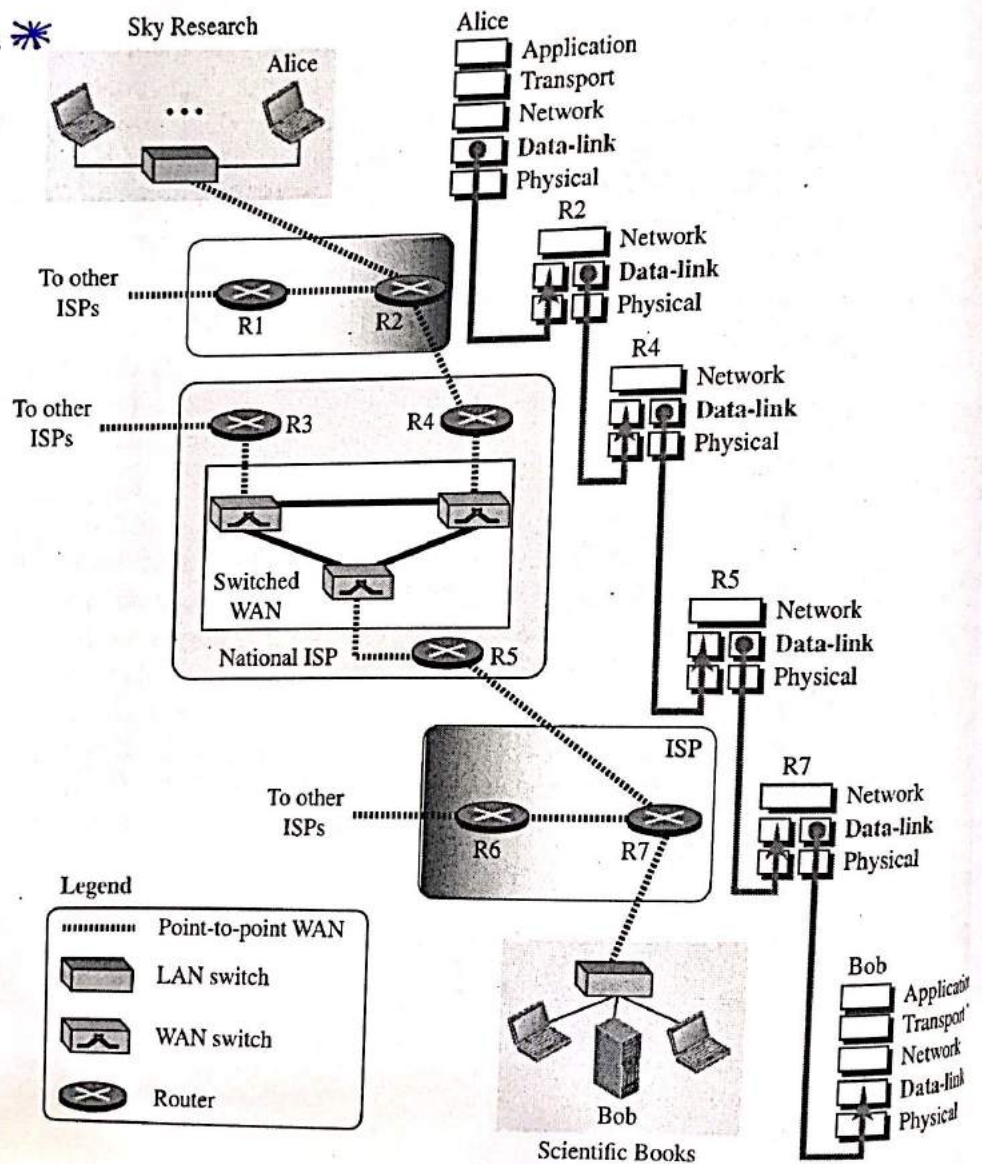
- The first section introduces the data-link layer. It starts with defining the concept of links and nodes. The section then lists and briefly describes the services provided by the data-link layer. It next defines two categories of links: point-to-point and broadcast links. The section finally defines two sublayers at the data-link layer that will be elaborated on in the next few chapters.
- The second section discusses link-layer addressing. It first explains the rationale behind the existence of an addressing mechanism at the data-link layer. It then describes three types of link-layer addresses to be found in some link-layer protocols. The section discusses the Address Resolution Protocol (ARP), which maps the addresses at the network layer to addresses at the data-link layer. This protocol helps a packet at the network layer find the link-layer address of the next node for delivery of the frame that encapsulates the packet. To show how the network layer helps us to find the data-link-layer addresses, a long example is included in this section that shows what happens at each node when a packet is travelling through the Internet.

9.1 INTRODUCTION

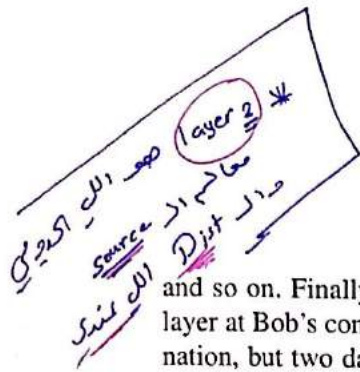
The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to travel through these networks. Figure 9.1 shows the same scenario we discussed in Chapter 8 but we are now interested in communication at the data-link layer. Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

Figure 9.1 Communication at the data-link layer

∴ الهدف من ال layer
 =
 reduce complexity.



The data-link layer at Alice's computer communicates with the data-link layer at router R2. The data-link layer at router R2 communicates with the data-link layer at router R4.

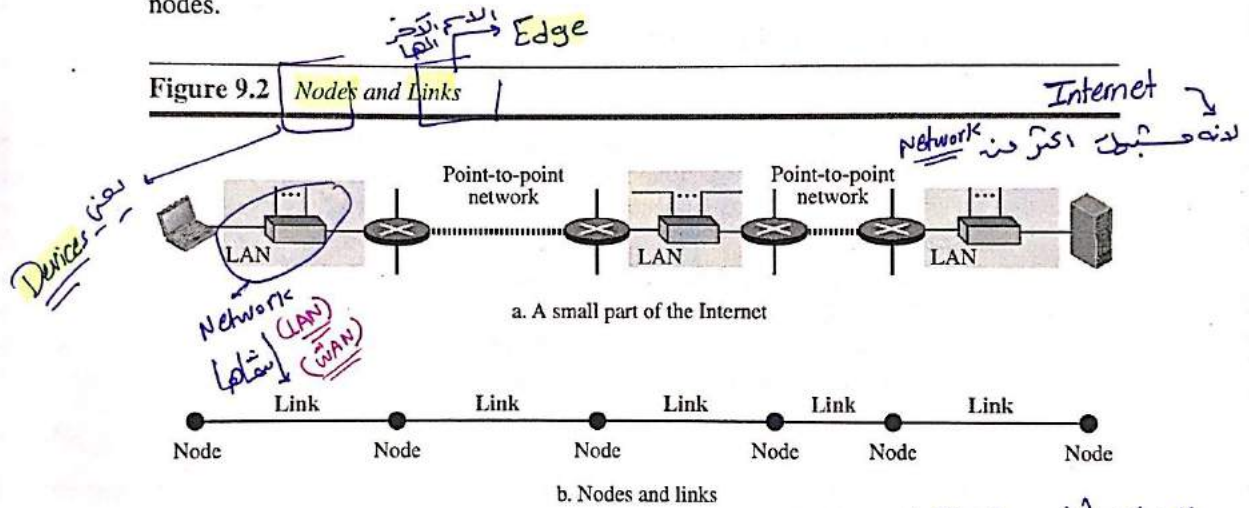


* IP → end to end
 * MAC → node to node
 لأنه في domain مختلف

and so on. Finally, the data-link layer at router R7 communicates with the data-link layer at Bob's computer. Only one data-link layer is involved at the source or the destination, but two data-link layers are involved at each router. The reason is that Alice's and Bob's computers are each connected to a single network, but each router takes input from one network and sends output to another network. Note that although switches are also involved in the data-link-layer communication, for simplicity we have not shown them in the figure.

9.1.1 Nodes and Links

Communication at the data-link layer is node-to-node. A data unit from one point in the Internet needs to pass through many networks (LANs and WANs) to reach another point. These LANs and WANs are connected by routers. It is customary to refer to the two end hosts and the routers as **nodes** and the networks in between as **links**. Figure 9.2 is a simple representation of links and nodes when the path of the data unit is only six nodes.



① Fram
 ② Flow Control
 ③ Error Control

كل layer service
 معتمدين على layer السابق
 لا يمكن ان يكونوا مستقلين

The first node is the source host; the last node is the destination host. The other four nodes are four routers. The first, the third, and the fifth links represent the three LANs; the second and the fourth links represent the two WANs.

9.1.2 Services

The data-link layer is located between the physical and the network layers. The data-link layer provides services to the network layer; it receives services from the physical layer. Let us discuss services provided by the data-link layer.

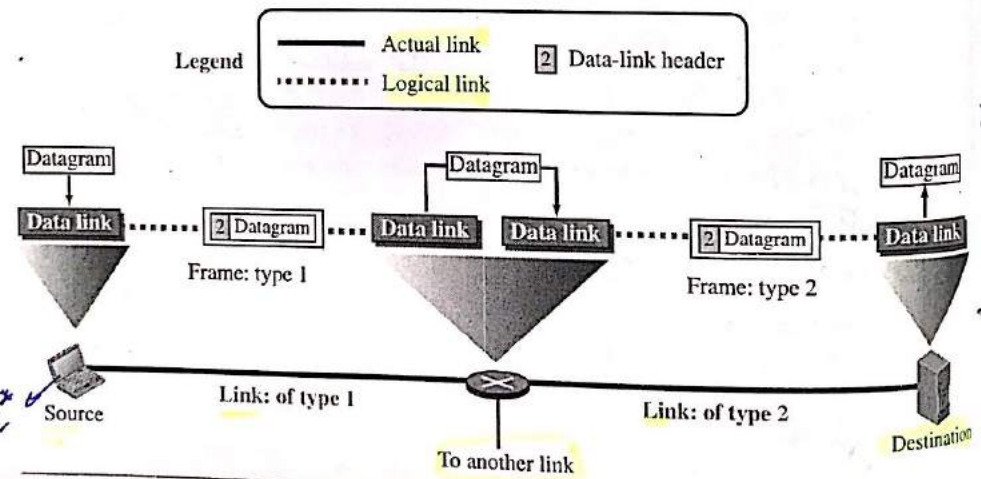
The duty scope of the data-link layer is **node-to-node**. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path. For this purpose, the data-link layer of the sending node needs to **encapsulate** the datagram received from the network in a frame, and the data-link layer of the receiving node needs to **decapsulate** the datagram from the frame. In other words, the data-link layer of the source host needs only to

* Flow control :-
 انه لازم ال sender يتأكد من انه ال Receiver
 كم بقدر نستقبل انه كم ال Capacity
 ال مستخدمه عشان اعمى كمان اجزاء

encapsulate, the data-link layer of the destination host needs to decapsulate, but each intermediate node needs to both encapsulate and decapsulate. One may ask why we need encapsulation and decapsulation at each intermediate node. The reason is that each link may be using a different protocol with a different frame format. Even if one link and the next are using the same protocol, encapsulation and decapsulation are needed because the link-layer addresses are normally different. An analogy may help in this case. Assume a person needs to travel from her home to her friend's home in another city. The traveller can use three transportation tools. She can take a taxi to go to the train station in her own city, then travel on the train from her own city to the city where her friend lives, and finally reach her friend's home using another taxi. Here we have a source node, a destination node, and two intermediate nodes. The traveller needs to get into the taxi at the source node, get out of the taxi and get into the train at the first intermediate node (train station in the city where she lives), get out of the train and get into another taxi at the second intermediate node (train station in the city where her friend lives), and finally get out of the taxi when she arrives at her destination. A kind of encapsulation occurs at the source node, encapsulation and decapsulation occur at the intermediate nodes, and decapsulation occurs at the destination node. Our traveller is the same, but she uses three transporting tools to reach the destination.

Figure 9.3 shows the encapsulation and decapsulation at the data-link layer. For simplicity, we have assumed that we have only one router between the source and destination. The datagram received by the data-link layer of the source host is encapsulated in a frame. The frame is logically transported from the source host to the router. The frame is decapsulated at the data-link layer of the router and encapsulated at another frame. The new frame is logically transported from the router to the destination host. Note that, although we have shown only two data-link layers at the router, the router actually has three data-link layers because it is connected to three physical links.

Figure 9.3 A communication with only three nodes



With the contents of the above figure in mind, we can list the services provided by a data-link layer as shown below.

تصنيف الحزم
 لتتناسب مع
 نوع الوصل
 MAC
 لتتناسب مع
 نوع الوصل
 Distribution
 للبيانات
 بين أجهزة
 الشبكة

Framing

* packet نبي
* frame (frame) نبي

Definitely, the first service provided by the data-link layer is **framing**. The data-link layer at each node needs to **encapsulate** the datagram (packet received from the network layer) in a **frame** before sending it to the next node. The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame, we will see in future chapters that a frame may have both a header and a trailer. Different data-link layers have different formats for framing.

A packet at the data-link layer is normally called a **frame**.

Flow Control

* Producer → sending
* Consumer → receiving

Whenever we have a **producer** and a **consumer**, we need to think about **flow control**. If the producer produces items that cannot be **consumed**, accumulation of items occurs. The sending data-link layer at the end of a link is a **producer of frames**; the receiving data-link layer at the other end of a link is a **consumer**. If the **rate of produced frames is higher than the rate of consumed frames**, frames at the receiving end need to be **buffered** while waiting to be consumed (processed). Definitely, we cannot have an **unlimited buffer size** at the receiving side. We have two choices. The first choice is to let the receiving data-link layer **drop** the frames if its **buffer is full**. The second choice is to let the receiving data-link layer **send a feedback** to the sending data-link layer to ask it to **stop or slow down**. Different data-link-layer protocols use different strategies for flow control. Since flow control also occurs at the transport layer, with a higher degree of importance, we discuss this issue in Chapter 23 when we talk about the transport layer.

Error Control :-

لا تحول الـ frame
إلى إشارة كهربية
تنتقل عبر
الوسط
منه لنا (bit) لا نستطيع
دفعه

At the sending node, a frame in a data-link layer needs to be **changed to bits**, transformed to **electromagnetic signals**, and transmitted through the **transmission media**. At the receiving node, electromagnetic signals are **received**, transformed to bits, and put together to create a **frame**. Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The **error needs first to be detected**. After detection, it needs to be either **corrected** at the receiver node or **discarded and retransmitted by the sending node**. Since error detection and correction is an issue in every layer (node-to-node or host-to-host), we have **dedicated** all of Chapter 10 to this issue.

Congestion Control :-

تأخر
في نقل
البيانات
Delay

Although a link may be congested with **frames**, which may result in **frame loss**, most data-link-layer protocols do not directly use a **congestion control** to alleviate congestion, although some wide-area networks do. In general, congestion control is considered an issue in the **network layer** or the **transport layer** because of its **end-to-end nature**. We will discuss congestion control in the network layer and the transport layer in later chapters.

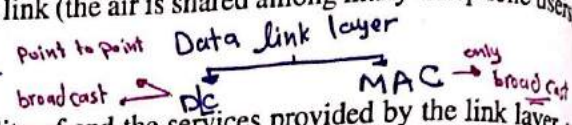
9.1.3 Two Categories of Links

Although two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the **data-link layer controls how the medium is used**. We can have a **data-link layer** that **uses the whole capacity of the medium**; we can also

one to
all

have a data-link layer that uses only part of the capacity of the link. In other words, a link can have a point-to-point link or a broadcast link. In a point-to-point link, the link is dedicated to the two devices; in a broadcast link, the link is shared between several pairs of devices. For example, when two friends use the traditional home phones to chat, they are using a point-to-point link; when the same two friends use their cell phones, they are using a broadcast link (the air is shared among many cell phone users).

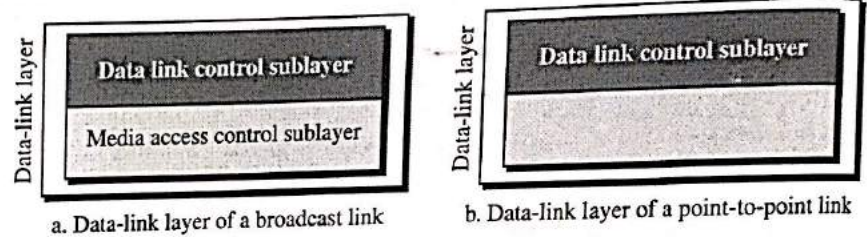
9.1.4 Two Sublayers



المطلوب اسألهم
وربما انظروا

To better understand the functionality of and the services provided by the link layer, we can divide the data-link layer into two sublayers: data link control (DLC) and media access control (MAC). This is not unusual because, as we will see in later chapters, LAN protocols actually use the same strategy. The data link control sublayer deals with issues common to both point-to-point and broadcast links; the media access control sublayer deals only with issues specific to broadcast links. In other words, we separate these two types of links at the data-link layer, as shown in Figure 9.4.

Figure 9.4 Dividing the data-link layer into two sublayers



We discuss the DLC and MAC sublayers later, each in a separate chapter. In addition, we discuss the issue of error detection and correction, a duty of the data-link and other layers, also in a separate chapter.

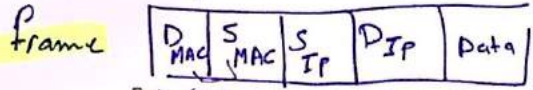
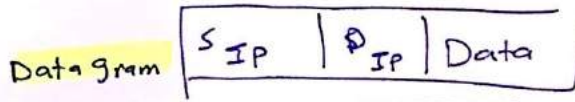
IP address
(32 bit)

9.2 LINK-LAYER ADDRESSING

بعض الأجهزة مؤقتة
إذا لم تكن
فإنها مجرد عنوان
لكنها تأتي بأهمية

The next issue we need to discuss about the data-link layer is the link-layer addresses. In Chapter 18, we will discuss IP addresses as the identifiers at the network layer that define the exact points in the Internet where the source and destination hosts are connected. However, in a connectionless internetwork such as the Internet we cannot make a datagram reach its destination using only IP addresses. The reason is that each datagram in the Internet, from the same source host to the same destination host, may take a different path. The source and destination IP addresses define the two ends but cannot define which links the datagram should pass through.

We need to remember that the IP addresses in a datagram should not be changed. If the destination IP address in a datagram changes, the packet never reaches its destination; if the source IP address in a datagram changes, the destination host or a router can never communicate with the source if a response needs to be sent back or an error needs to be reported back to the source (see ICMP in Chapter 19).



معلومات

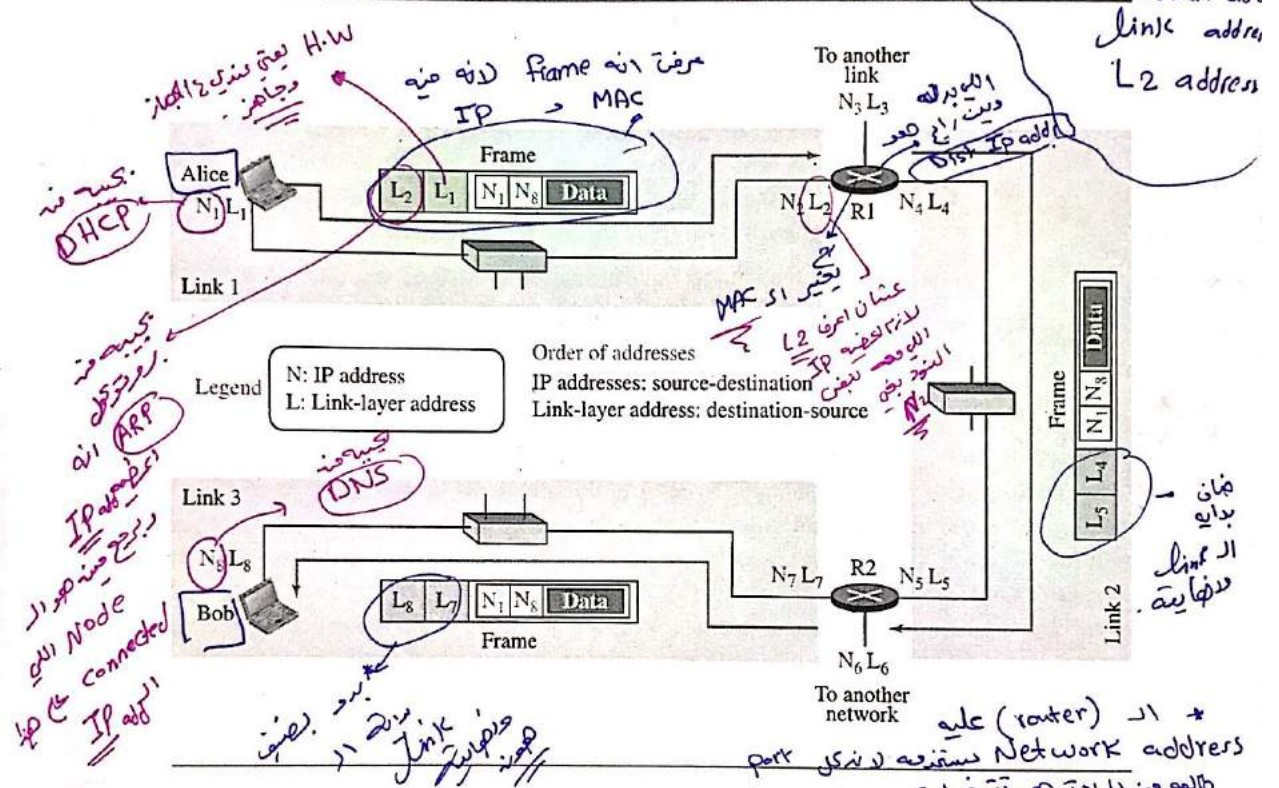
IP → Logical Add
 → software Add
 L3 address
 Network address
 connection add

The above discussion shows that we need another addressing mechanism in a connectionless internetwork: the link-layer addresses of the two nodes. A link-layer address is sometimes called a link address, sometimes a physical address, and sometimes a MAC address. We use these terms interchangeably in this book.

يعني انه اد
 connect
 ولي
 بالجوهر

Since a link is controlled at the data-link layer, the addresses need to belong to the data-link layer. When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another. Figure 9.5 demonstrates the concept in a small internet.

Figure 9.5 IP addresses and link-layer addresses in a small internet



In the internet in Figure 9.5, we have three links and two routers. We also have shown only two hosts: Alice (source) and Bob (destination). For each host, we have shown two addresses, the IP addresses (N) and the link-layer addresses (L). Note that a router has as many pairs of addresses as the number of links the router is connected to. We have shown three frames, one in each link. Each frame carries the same datagram with the same source and destination addresses (N1 and N8), but the link-layer addresses of the frame change from link to link. In link 1, the link-layer addresses are L1 and L2. In link 2, they are L4 and L5. In link 3, they are L7 and L8. Note that the IP addresses and the link-layer addresses are not in the same order. For IP addresses, the source address comes before the destination address; for link-layer addresses, the destination address comes before the source. The datagrams and

تالعه من الادره في تتوفر
 لانه كبرية في
 الين كبرية في
 شدة ف
 يمكن
 يكون
 الراوتر
 نفسه
 انا source
 او
 Distribution ف لازم يكونه

ال (IP) وعليه الانترنت (IP) لانه
 connected مع اكثر من نتورك

frames are designed in this way, and we follow the design. We may raise several questions:

- If the IP address of a router does not appear in any datagram sent from a source to a destination, why do we need to assign IP addresses to routers? The answer is that in some protocols a router may act as a sender or receiver of a datagram. For example, in routing protocols we will discuss in Chapters 20 and 21, a router is a sender or a receiver of a message. The communications in these protocols are between routers.
- Why do we need more than one IP address in a router, one for each interface? The answer is that an interface is a connection of a router to a link. We will see that an IP address defines a point in the Internet at which a device is connected. A router with n interfaces is connected to the Internet at n points. This is the situation of a house at the corner of a street with two gates; each gate has the address related to the corresponding street.
- How are the source and destination IP addresses in a packet determined? The answer is that the host should know its own IP address, which becomes the source IP address in the packet. As we will discuss in Chapter 26, the application layer uses the services of DNS to find the destination address of the packet and passes it to the network layer to be inserted in the packet.
- How are the source and destination link-layer addresses determined for each link? Again, each hop (router or host) should know its own link-layer address, as we discuss later in the chapter. The destination link-layer address is determined by using the Address Resolution Protocol, which we discuss shortly.
- What is the size of link-layer addresses? The answer is that it depends on the protocol used by the link. Although we have only one IP protocol for the whole Internet, we may be using different data-link protocols in different links. This means that we can define the size of the address when we discuss different link-layer protocols.

كيفية الاقبال
Destination address of the packet
البروتوكول الذي يستخدمه
destination link-layer address

9.2.1 Three Types of addresses

Some link-layer protocols define three types of addresses: unicast, multicast, and broadcast.

Unicast Address

Each host or each interface of a router is assigned a unicast address. Unicasting means one-to-one communication. A frame with a unicast address destination is destined only for one entity in the link.

Example 9.1

As we will see in Chapter 13, the unicast link-layer addresses in the most common LAN, Ethernet, are 48 bits (six bytes) that are presented as 12 hexadecimal digits separated by colons; for example, the following is a link-layer address of a computer.

A3:34:45:11:92:F1

Multicast Address

Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link).

كلمة حويدة لسوا
كل واحد

الانسان
منه حياجه لمصروفه
منه حياجه كبره

عدد even

48
MAC

منه
12
عدد

multicast add
unicast ← even

Example 9.2

As we will see in Chapter 13, the multicast link-layer addresses in the most common LAN, Ethernet, are 48 bits (six bytes) that are presented as 12 hexadecimal digits separated by colons. The second digit, however, needs to be an even number in hexadecimal. The following shows a multicast address:

A2:34:45:11:92:F1

Broadcast Address

Some link-layer protocols define a broadcast address. Broadcasting means one-to-all communication. A frame with a destination broadcast address is sent to all entities in the link.

Example 9.3

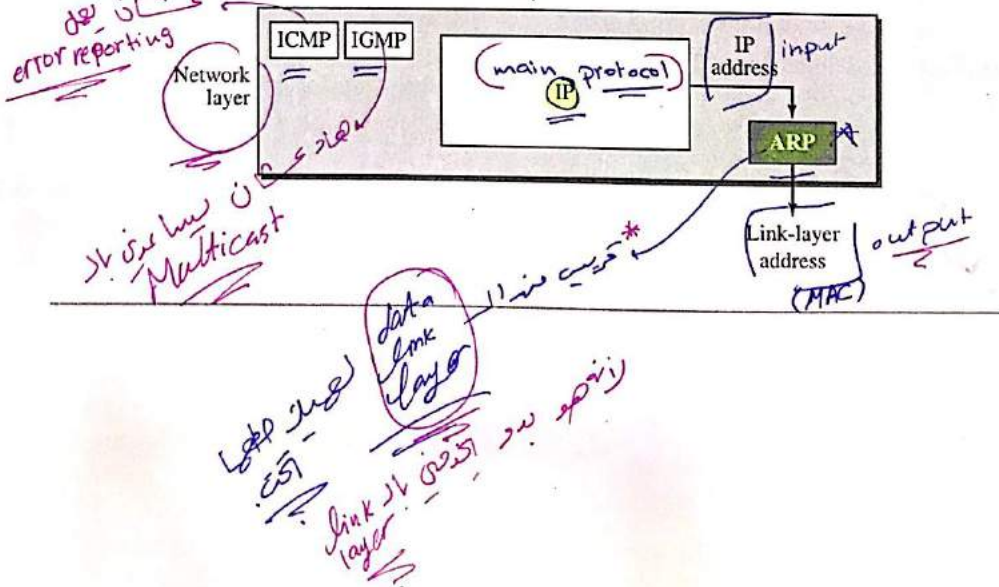
As we will see in Chapter 13, the broadcast link-layer addresses in the most common LAN, Ethernet, are 48 bits, all 1s, that are presented as 12 hexadecimal digits separated by colons. The following shows a broadcast address:

FF:FF:FF:FF:FF:FF

9.2.2 Address Resolution Protocol (ARP)

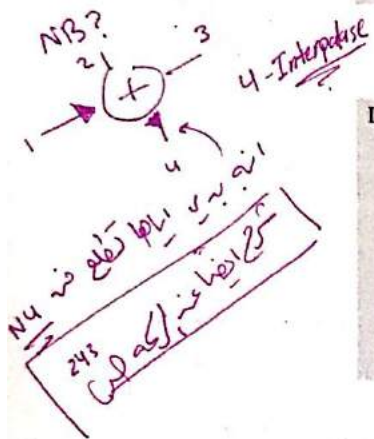
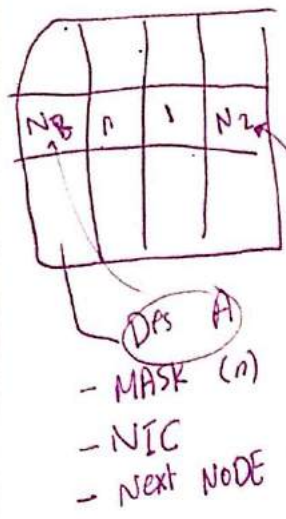
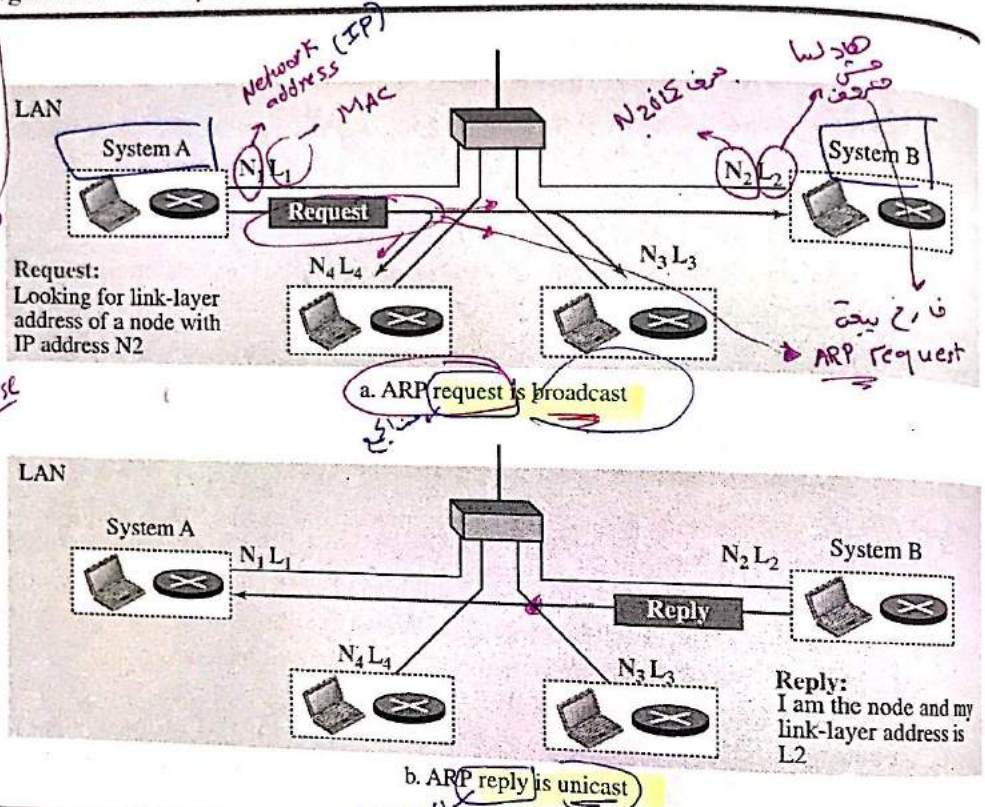
Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router. Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router knows the IP address of the destination host. However, the IP address of the next node is not helpful in moving a frame through a link: we need the link-layer address of the next node. This is the time when the Address Resolution Protocol (ARP) becomes helpful. The ARP protocol is one of the auxiliary protocols defined in the network layer, as shown in Figure 9.6. It belongs to the network layer, but we discuss it in this chapter because it maps an IP address to a logical-link address. ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

Figure 9.6 Position of ARP in TCP/IP protocol suite



Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address, which we discuss for each protocol later (see Figure 9.7).

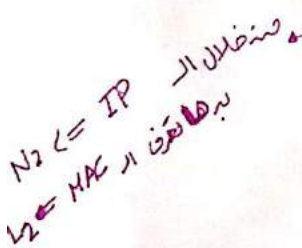
Figure 9.7 ARP operation



Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and link-layer addresses. The packet is unicast directly to the node that sent the request packet.

In Figure 9.7a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address N_2 . System A needs to pass the packet to the data-link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of (N_2) .

This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 9.7b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.



Caching

A question that is often asked is this: If system A can broadcast a frame to find the link-layer address of system B, why can't system A send the datagram for system B using a broadcast frame? In other words, instead of sending one broadcast frame (ARP request), one unicast frame (ARP response), and another unicast frame (for sending the datagram), system A can encapsulate the datagram and send it to the network. System B receives it and keep it; other systems discard it.

To answer the question, we need to think about the efficiency. It is probable that system A has more than one datagram to send to system B in a short period of time. For example, if system B is supposed to receive a long e-mail or a long file, the data do not fit in one datagram.

Let us assume that there are 20 systems connected to the network (link): system A, system B, and 18 other systems. We also assume that system A has 10 datagrams to send to system B in one second.

- a. Without using ARP, system A needs to send 10 broadcast frames. Each of the 18 other systems need to receive the frames, decapsulate the frames, remove the datagram and pass it to their network-layer to find out the datagrams do not belong to them. This means processing and discarding 180 broadcast frames.
- b. Using ARP, system A needs to send only one broadcast frame. Each of the 18 other systems need to receive the frames, decapsulate the frames, remove the ARP message and pass the message to their ARP protocol to find that the frame must be discarded. This means processing and discarding only 18 (instead of 180) broadcast frames. After system B responds with its own data-link address, system A can store the link-layer address in its cache memory. The rest of the nine frames are only unicast. Since processing broadcast frames is expensive (time consuming), the first method is preferable.

Packet Format

Figure 9.8 shows the format of an ARP packet. The names of the fields are self-explanatory. The *hardware type* field defines the type of the link-layer protocol; Ethernet is given the type 1. The *protocol type* field defines the network-layer protocol: IPv4 protocol is $(0800)_{16}$. The source hardware and source protocol addresses are variable-length fields defining the link-layer and network-layer addresses of the sender. The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses. An ARP packet is encapsulated directly into a data-link frame. The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.

Example 9.4

A host with IP address $N1$ and MAC address $L1$ has a packet to send to another host with IP address $N2$ and physical address $L2$ (which is unknown to the first host). The two hosts are on the same network. Figure 9.9 shows the ARP request and response messages.

Figure 9.8 ARP packet

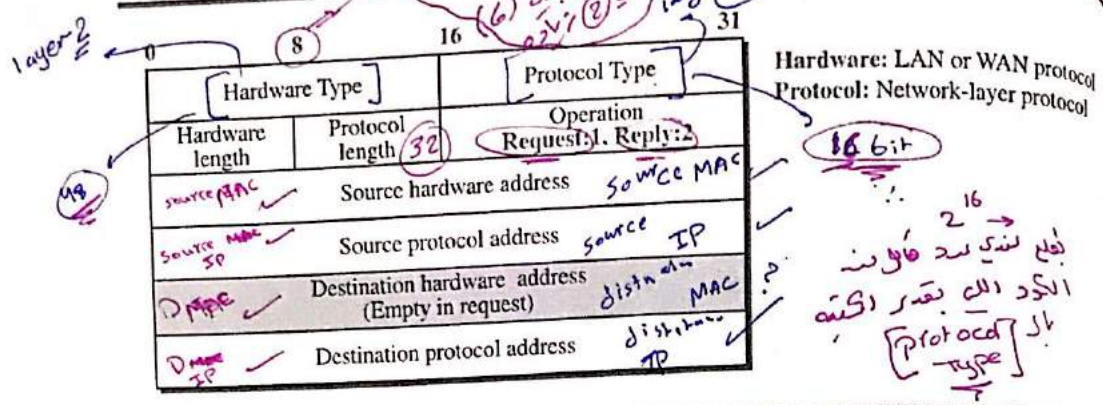
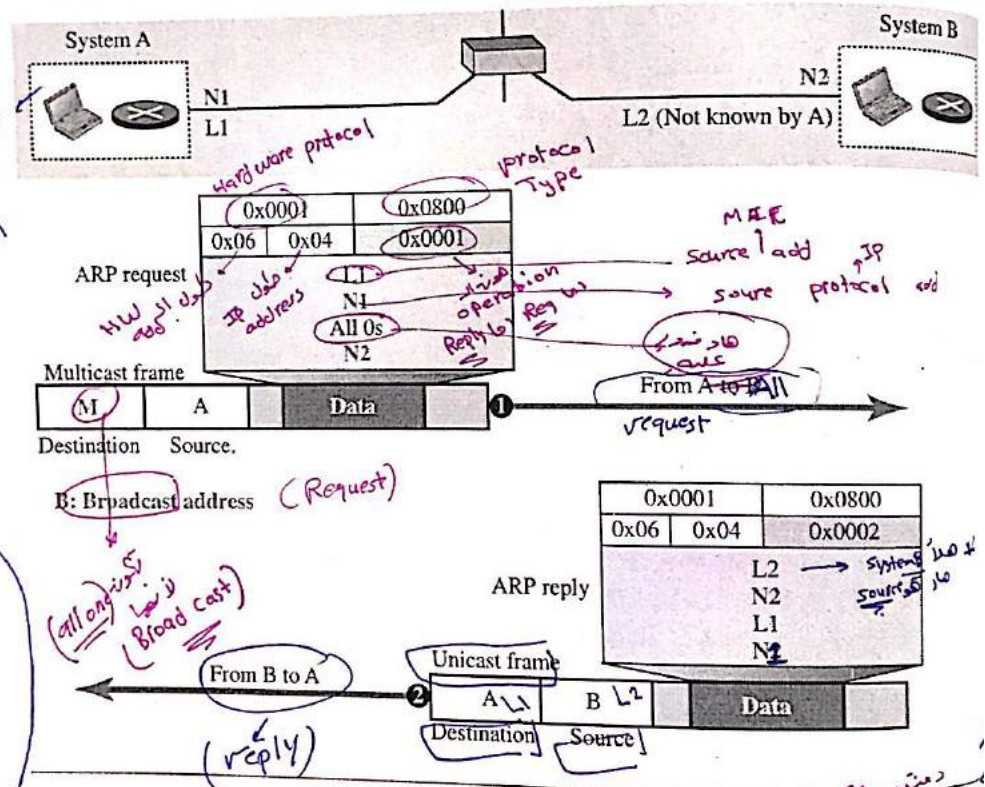


Figure 9.9 Example 9.4



بدر بعرف ان
MAC
نوع سبوت
System B
ARP
N2 or N1

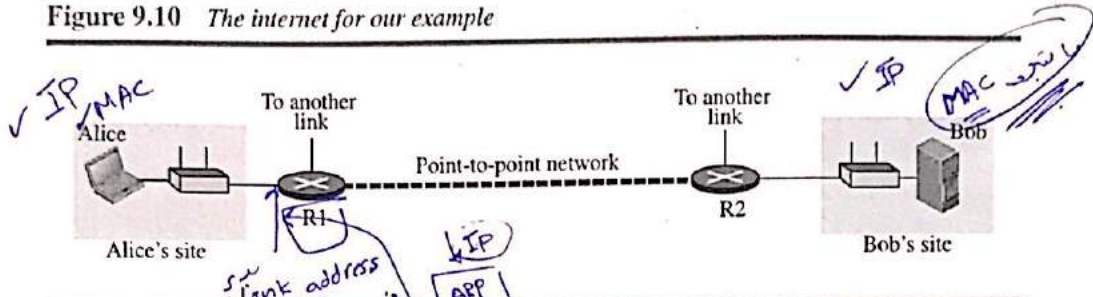
انا لما بعرف ان
بي اوصيه لا يوصل
لهيلا
IP datagram = ARP Protocol
لانه سبوت
Table

9.2.3 An Example of Communication

To show how communication is done at the data-link layer and how link-layer addresses are found, let us go through a simple example. Assume Alice needs to send a datagram to Bob, who is three nodes away in the Internet. How Alice finds the network-layer address of Bob is what we discover in Chapter 26 when we discuss DNS. For the moment, assume that Alice knows the network-layer (IP) address of Bob. In other words, Alice's host is given the data to be sent, the IP address of Bob, and the

IP address of Alice's host (each host needs to know its IP address). Figure 9.10 shows the part of the internet for our example.

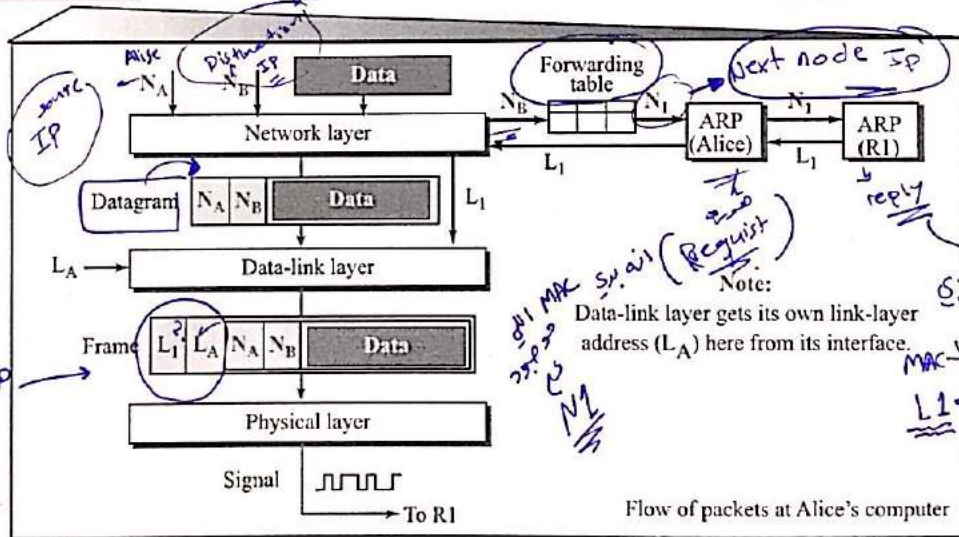
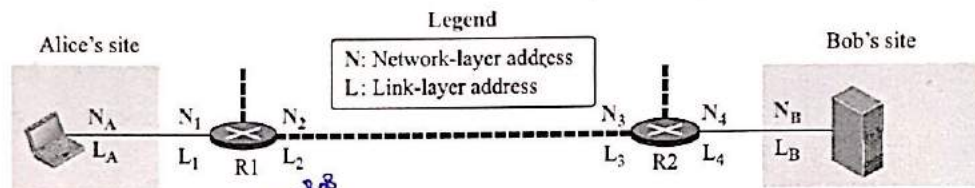
Figure 9.10 The internet for our example



Activities at Alice's Site

We will use symbolic addresses to make the figures more readable. Figure 9.11 shows what happens at Alice's site.

Figure 9.11 Flow of packets at Alice's computer



من الـ MAC
انـه الـ IP
الـ MAC

الـ MAC الـ
الـ IP الـ
الـ MAC الـ
الـ IP الـ

ويكونه بالـ
الـ IP الـ
الـ MAC الـ
الـ IP الـ

The network layer knows it's given N_A , N_B , and the packet, but it needs to find the link-layer address of the next node. The network layer consults its routing table and tries to find which router is next (the default router in this case) for the destination N_B . As we will discuss in Chapter 18, the routing table gives N_1 , but the network layer

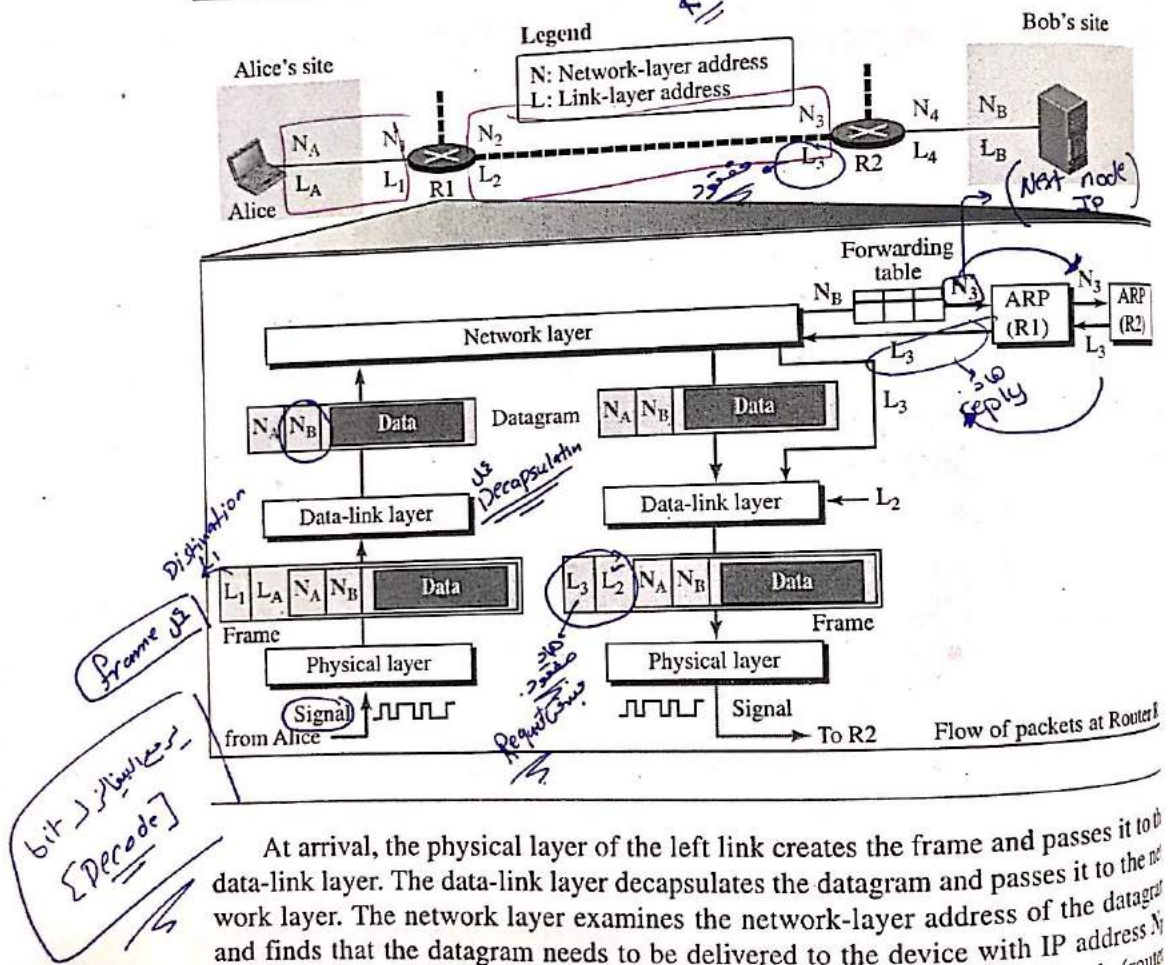
needs to find the link-layer address of router R1. It uses its ARP to find the link-layer address L_1 . The network layer can now pass the datagram with the link-layer address to the data-link layer.

The data-link layer knows its own link-layer address, L_A . It creates the frame and passes it to the physical layer, where the address is converted to signals and sent through the media.

Activities at Router R1

Now let us see what happens at Router R1. Router R1, as we know, has only three lower layers. The packet received needs to go up through these three layers and come down. Figure 9.12 shows the activities.

Figure 9.12 Flow of activities at router R1



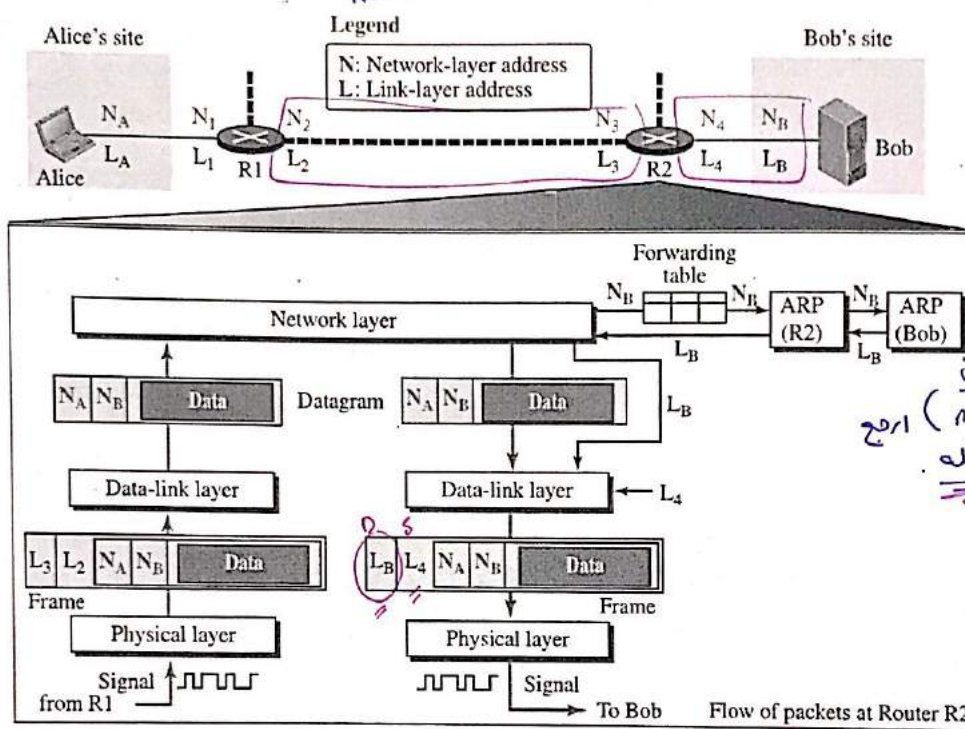
At arrival, the physical layer of the left link creates the frame and passes it to the data-link layer. The data-link layer decapsulates the datagram and passes it to the network layer. The network layer examines the network-layer address of the datagram and finds that the datagram needs to be delivered to the device with IP address N_B . The network layer consults its routing table to find out which is the next node (router) in the path to N_B . The forwarding table returns N_3 . The IP address of router R2 is the same link with R1. The network layer now uses the ARP to find the link-layer address of this router, which comes up as L_3 . The network layer passes the datagram and L_3 to the data-link layer belonging to the link at the right side. The link layer

encapsulates the datagram, adds L3 and L2 (its own link-layer address), and passes the frame to the physical layer. The physical layer encodes the bits to signals and sends them through the medium to R2.

Activities at Router R2

Activities at router R2 are almost the same as in R1, as shown in Figure 9.13.

Figure 9.13 Activities at router R2.



* بی بی
اعلی
Decapsulation
مانندی یعنی
new frame
اربع
اسکله

Activities at Bob's Site

Now let us see what happens at Bob's site. Figure 9.14 shows how the signals at Bob's site are changed to a message. At Bob's site there are no more addresses or mapping needed. The signal received from the link is changed to a frame. The frame is passed to the data-link layer, which decapsulates the datagram and passes it to the network layer. The network layer decapsulates the message and passes it to the transport layer.

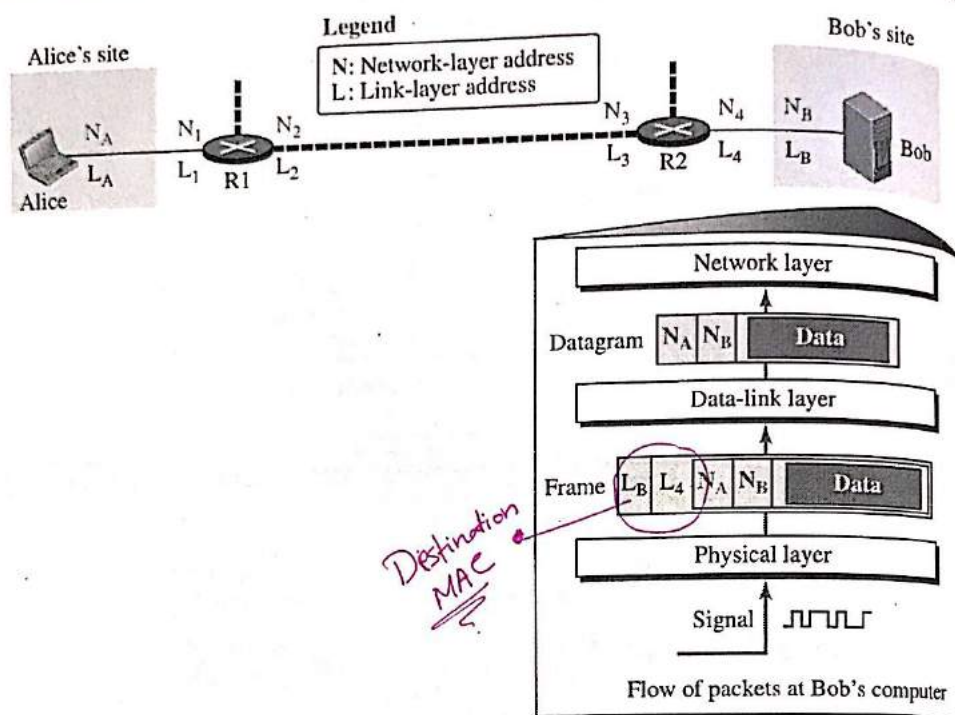
Changes in Addresses

This example shows that the source and destination network-layer addresses, N_A and N_B , have not been changed during the whole journey. However, all four network-layer addresses of routers R1 and R2 (N_1 , N_2 , N_3 , and N_4) are needed to transfer a datagram from Alice's computer to Bob's computer.

* ہذا میں بدلنا انا اچھا | **Broad Cast** | و توکل
 * یعنی پوری تانبہ انہا میں
 ① security ↓
 ② Network
 ③ پرداز عینا و processing

unicast | و توکل | **Broad Cast**
 * ہذا میں بدلنا انا اچھا | **Broad Cast** | و توکل
 * یعنی پوری تانبہ انہا میں
 ① security ↓
 ② Network
 ③ پرداز عینا و processing

Figure 9.14 Activities at Bob's site



9.3 END-CHAPTER MATERIALS

9.3.1 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

Several books discuss link-layer issues. Among them we recommend [Ham 80], [Zar 01], [Ror 96], [Tan 03], [GW 04], [For 03], [KMK 04], [Sta 04], [Kes 02], [PD 03], [K 02], [Spu 00], [KCK 98], [Sau 98], [Izz 00], [Per 00], and [WV 00].

9.3.2 Key Terms

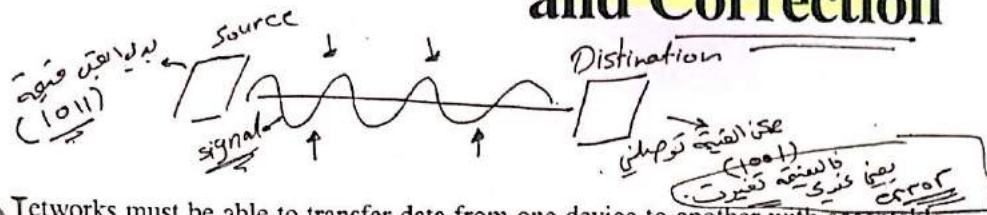
Address Resolution Protocol (ARP)
 data link control (DLC)
 frame
 framing

links
 media access control (MAC)
 nodes

9.3.3 Summary

The Internet is made of many hosts, networks, and connecting devices such as routers. The hosts and connecting devices are referred to as *nodes*; the networks are referred to

Error Detection and Correction



Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted. Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting errors.

Some applications can tolerate a small level of error. For example, random errors in audio or video transmissions may be tolerable, but when we transfer text, we expect a very high level of accuracy.

At the data-link layer, if a frame is corrupted between the two nodes, it needs to be corrected before it continues its journey to other nodes. However, most link-layer protocols simply discard the frame and let the upper-layer protocols handle the retransmission of the frame. Some multimedia applications, however, try to correct the corrupted frame.

This chapter is divided into five sections.

- The first section introduces (types of errors) the concept of redundancy, and distinguishes between error detection and correction.
- The second section discusses (block coding). It shows how error can be detected using block coding and also introduces the concept of Hamming distance.
- The third section discusses cyclic codes. It discusses a subset of cyclic code, CRC, that is very common in the data-link layer. The section shows how CRC can be easily implemented in hardware and represented by polynomials.
- The fourth section discusses checksums. It shows how a checksum is calculated for a set of data words. It also gives some other approaches to traditional checksum.
- The fifth section discusses forward error correction. It shows how Hamming distance can also be used for this purpose. The section also describes cheaper methods to achieve the same goal, such as XORing of packets, interleaving chunks, or compounding high and low resolutions packets.

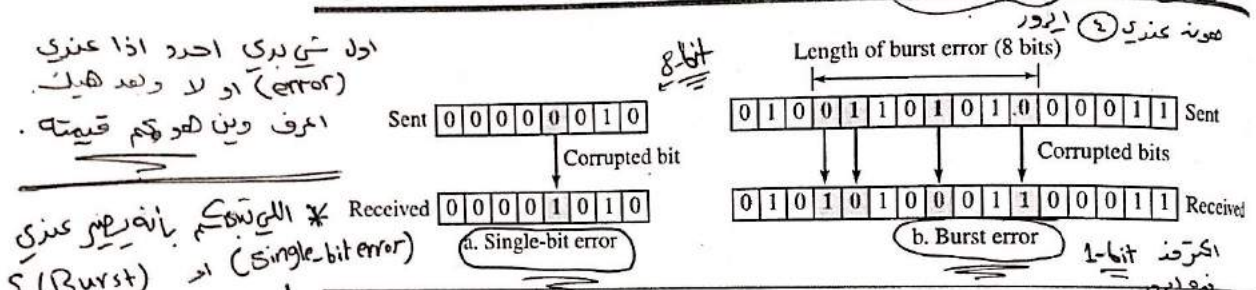
10.1 INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correction.

10.1.1 Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of **interference**. This interference can change the shape of the signal. The term **single-bit error** means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1. The term **burst error** means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1. Figure 10.1 shows the effect of a single-bit and a burst error on a data unit.

Figure 10.1 Single-bit and burst error



اول شي بدري احدد اذا عندي (error) او لا وبعد هيك اعرف وين هو وكم قيمته.

اول شي بدري احدد اذا عندي (error) او لا وبعد هيك اعرف وين هو وكم قيمته.

اذا كان الـ 100 = 1000 bit
 1 Kbps
 جان عدد الـ (bit) العرضية
 $1 \text{ Mbps} \times \frac{1}{100} = \text{error}$
 $= 10000 \text{ bit}$
 لو بينرنا الـ (bit) الـ
 عدد الـ bit الـ 1 Mbps
 فانه عدد error
 $1 \text{ Mbps} \times \frac{1}{100} = 10 \text{ K bit}$

Duration of noise
 المؤثر الكافي لحدوث noise

Transmission speed
 rate
 يعني M او I

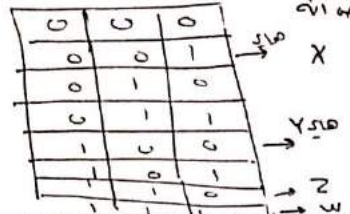
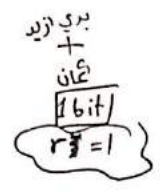
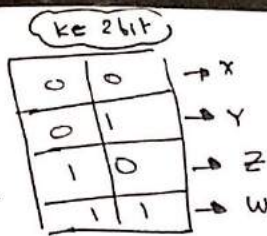
A burst error is more likely to occur than a single-bit error because the duration of the noise signal is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 kbps, a noise of 1/100 second can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

10.1.2 Redundancy

The central concept in detecting or correcting errors is **redundancy**. To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

10.1.3 Detection versus Correction

The correction of errors is more difficult than the detection. In error detection, we are only looking to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of corrupted bits. A single-bit error is the same for us as a burst error. In error correction we need to know the exact number of bits that are corrupted and, more importantly, their location in the message. The number of errors and the size of the message are important factors. If we need to correct a single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two



* لو نفضه انه
عدد ال bits
الاصليه
لزيادة
* $n = k + r$
 $= 2 + 1 = 3$
لا فضل كندى
صلى
مستحسن
Competition
مستحسن

errors in a data unit of the same size, we need to consider 28 (permutation of 8 by 2) possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

Block coding
Convolution coding

10.1.4 Coding

صدا بس افق $X = (001)$ ممكن تكبرها ل $Noise$ ن تكبر ريس
(error) بارل bit ريس (000) دهالوش مستقره ن معناها هيا (error)
ر موار (error) مار bit اشافي (011) برضه هيا مستقره نعرف انه (Error) دهالو

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect errors. The ratio of redundant bits to data bits and the robustness of the process are important factors in any coding scheme.

We can divide coding schemes into two broad categories: **block coding** and **convolution coding**. In this book, we concentrate on block coding; convolution coding is more complex and beyond the scope of this book.

افق ار
[extra bit]
اما اذا صار
ياي ك طرة
متاكدودان
مفهوم
مشكله دهالو
1 نهار اكود ال كندى هيا صوبي كندى

10.2 BLOCK CODING → انه انما صفة ال data ل Block

In block coding, we divide our message into blocks, each of k bits, called **datawords**. We add r redundant bits to each block to make the length $n = k + r$. The resulting n -bit blocks are called **codewords**. How the extra r bits are chosen or calculated is something we will discuss later. For the moment, it is important to know that we have a set of datawords, each of size k , and a set of codewords, each of size of n . With k bits, we can create a combination of 2^k datawords; with n bits, we can create a combination of 2^n codewords. Since $n > k$, the number of possible codewords is larger than the number of possible datawords. The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have $2^n - 2^k$ codewords that are not used. We call these codewords invalid or illegal. The trick in error detection is the existence of these invalid codes, as we discuss next. If the receiver receives an invalid codeword, this indicates that the data was corrupted during transmission.

بعض الرياضه

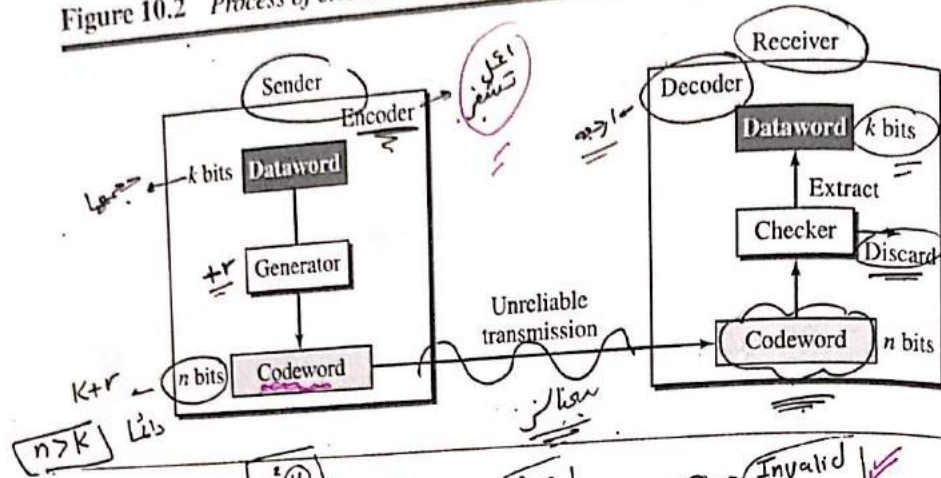
10.2.1 Error Detection :-

How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.
2. The original codeword has changed to an invalid one.

Figure 10.2 shows the role of block coding in error detection. The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding (discussed later). Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected.

Figure 10.2 Process of error detection in block coding



Example 10.1

Let us assume that $k=2$ and $n=3$. Table 10.1 shows the list of datawords and codewords. We will see how to derive a codeword from a dataword.

Table 10.1 A code for error detection in Example 10.1

Dataword	Codeword	Dataword	Codeword
00	000	10	101
01	011	11	110

هذا مثال عملي في الكود
البيتي في طول الكود البيتي
[2, 3] فقط

Assume the sender encodes the dataword 01 and sends it to the receiver. Consider the following cases:

1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.
3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. The corrupted bits have made the error undetectable.

لا توجد
Noise
بالتالي
2-error
بالتالي

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Hamming Distance :-

One of the central concepts in coding for error control is the idea of the Hamming distance. The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words x and y as $d(x, y)$. We may wonder why Hamming distance is important for error detection. The reason is that the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is $d(00000, 01101) = 3$. In other words, if the Hamming

عدد ال (bit) المختلفة
0110
1110
1001
بالتالي
عدد ال اختلافات اذا كان
عدد ال bit كبر يكون
كم الاختلافات كبر فكل
بالتالي (XOR)
بالتالي

distance between the sent and the received codeword is not zero, the codeword has been corrupted during transmission.

The Hamming distance can easily be found if we apply the XOR operation (\oplus) on the two words and count the number of 1s in the result. Note that the Hamming distance is a value greater than or equal to zero.

The Hamming distance between two words is the number of differences between corresponding bits.

Example 10.2

مسافة اعرصية

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance $d(000, 011)$ is 2 because $(000 \oplus 011)$ is 011 (two 1s).
2. The Hamming distance $d(10101, 11110)$ is 3 because $(10101 \oplus 11110)$ is 01011 (three 1s).

Minimum Hamming Distance for Error Detection

000
011
101
110
minimum distance = 2

In a set of codewords, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of codewords. Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s . If our system is to detect up to s errors, the minimum distance between the valid codes must be $(s + 1)$, so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid codewords is $(s + 1)$, the received codeword cannot be erroneously mistaken for another codeword. The error will be detected. We need to clarify a point here: Although a code with $d_{\min} = s + 1$ may be able to detect more than s errors in some special cases, only s or fewer errors are guaranteed to be detected.

Hamming Distance
أكبر صفة 20
صفا في صفة
اختلاف صفة
Error

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{\min} = s + 1$. $s = \text{Max number of error}$

We can look at this criteria geometrically. Let us assume that the sent codeword x is at the center of a circle with radius s . All received codewords that are created by 0 to s errors are points inside the circle or on the perimeter of the circle. All other valid codewords must be outside the circle, as shown in Figure 10.3. This means that d_{\min} must be an integer greater than s or $d_{\min} = s + 1$.

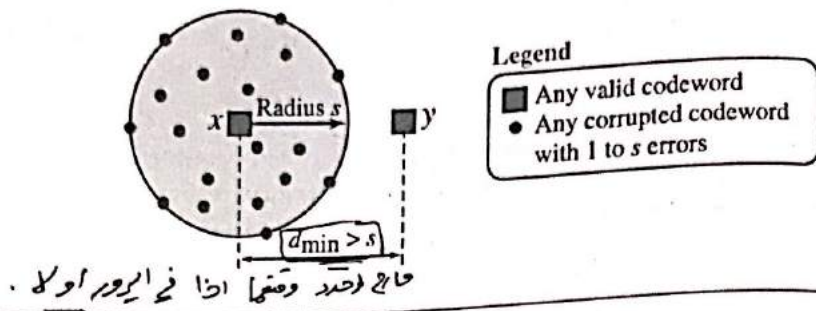
Example 10.3

The minimum Hamming distance for our first code scheme (Table 10.1) is 2. This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Example 10.4

A code scheme has a Hamming distance $d_{\min} = 4$. This code guarantees the detection of up to three errors ($d = s + 1$ or $s = 3$).

Figure 10.3 Geometric concept explaining d_{min} in error detection



Linear Block Codes

Almost all block codes used today belong to a subset of block codes called *linear block codes*. The use of nonlinear block codes for error detection and correction is not widespread because their structure makes theoretical analysis and implementation difficult. We therefore concentrate on linear block codes. The formal definition of linear block codes requires the knowledge of abstract algebra (particularly Galois fields), which is beyond the scope of this book. We therefore give an informal definition. For our purposes, a linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

Example 10.5

The code in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.

Minimum Distance for Linear Block Codes

It is simple to find the minimum Hamming distance for a linear block code. The minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Example 10.6

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$.

Parity-Check Code

Perhaps the most familiar error-detecting code is the **parity-check code**. This code is a linear block code. In this code, a k -bit dataword is changed to an n -bit codeword where $n = k + 1$. The extra bit, called the *parity bit*, is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of 1s, we discuss the even case. The minimum Hamming distance for this category is $d_{min} = 2$, which means that the code is a single-bit error-detecting code. Our first code (Table 10.1) is a parity-check code ($k = 2$ and $n = 3$). The code in Table 10.2 is also a parity-check code with $k = 4$ and $n = 5$.

CHAPTER 13

Wired LANs: Ethernet

* كان عندى (other Technology) غير (Ethernet) لغاى الوجوده المي واكتبه الطور هه Ethernet

After discussing the general issues related to the data-link layer in Chapters 9 to 12, it is time in this chapter to discuss the wired LANs. Although over a few decades many wired LAN protocols existed, only the Ethernet technology survives today. This is the reason that we discuss only this technology and its evolution in this chapter.

This chapter is divided into five sections. ^{Frame} ال (Format) الصيغ اللى بتطلع منه هيا [Frame] اسمها

- ❑ The first section discusses the Ethernet protocol in general. It explains that IEEE Project 802 defines the LLC and MAC sublayers for all LANs including Ethernet. The section also lists the four generations of Ethernet.
- ❑ The second section discusses the Standard Ethernet. Although this generation is rarely seen in practice, most of the characteristics have been inherited by the following three generations. The section first describes some characteristics of the Standard Ethernet. It then discusses the addressing mechanism, which is the same in all Ethernet generations. The section next discusses the access method, CSMA/CD, which we discussed in Chapter 12. The section then reviews the efficiency of the Standard Ethernet. It then shows the encoding and the implementation of this generation. Before closing the section, the changes in this generation that resulted in the move to the next generation are listed.
- ❑ The third section describes the Fast Ethernet, the second generation, which can still be seen in many places. The section first describes the changes in the MAC sublayer. The section then discusses the physical layer and the implementation of this generation.
- ❑ The fourth section discusses the Gigabit Ethernet, with the rate of 1 gigabit per second. The section first describes the MAC sublayer. It then moves to the physical layer and implementation.
- ❑ The fifth section touches on the 10 Gigabit Ethernet. This is a new technology that can be used both for a backbone LAN or as a MAN (metropolitan area network). The section briefly describes the rationale and the implementation.

بعض عندي
تطور
رقة ال
Ethernet
Technology

13.1 ETHERNET PROTOCOL

In Chapter 1, we mentioned that the TCP/IP protocol suite does not define any protocols for the data-link or the physical layer. In other words, TCP/IP accepts any protocols in these two layers that can provide services to the network layer. The data-link layer and the physical layer are actually the territory of the local and wide area networks. This means that when we discuss these two layers, we are talking about networks that are using them. As we see in this and the following two chapters, we can have wired and wireless networks. We discuss wired networks in this chapter and the next and postpone the discussion of wireless networks to Chapter 15.

In Chapter 1, we learned that a local area network (LAN) is a computer network that is designed for a limited geographic area such as a building or a campus. Although a LAN can be used as an isolated network to connect computers in an organization for the sole purpose of sharing resources, most LANs today are also linked to a wide area network (WAN) or the Internet.

In the 1980s and 1990s several different types of LANs were used. All of these LANs used a media-access method to solve the problem of sharing the media. The Ethernet used the CSMA/CD approach. The Token Ring, Token Bus, and FDDI (Fiber Distribution Data Interface) used the token-passing approach. During this period another LAN technology, ATM LAN, which deployed the high speed WAN technology (ATM), appeared in the market.

Almost every LAN except Ethernet has disappeared from the marketplace because Ethernet was able to update itself to meet the needs of the time. Several reasons for this success have been mentioned in the literature, but we believe that the Ethernet protocol was designed so that it could evolve with the demand for higher transmission rates. It is natural that an organization that has used an Ethernet LAN in the past and now needs a higher data rate would update to the new generation instead of switching to another technology, which might cost more. This means that we confine our discussion of wired LANs to the discussion of Ethernet.

access method
الوصول الى
الذي ينظم من عندنا علمية
Sharing

13.1.1 IEEE Project 802 :-

Before we discuss the Ethernet protocol and all its generations, we need to briefly discuss the IEEE standard that we often encounter in text or real life. In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 does not seek to replace any part of the OSI model or TCP/IP protocol suite. Instead, it is a way of specifying functions of the physical layer and the data-link layer of major LAN protocols.

The relationship of the 802 Standard to the TCP/IP protocol suite is shown in Figure 13.1. The IEEE has subdivided the data-link layer into two sublayers: logical link control (LLC) and media access control (MAC). IEEE has also created several physical-layer standards for different LAN protocols.

Logical Link Control (LLC) :-

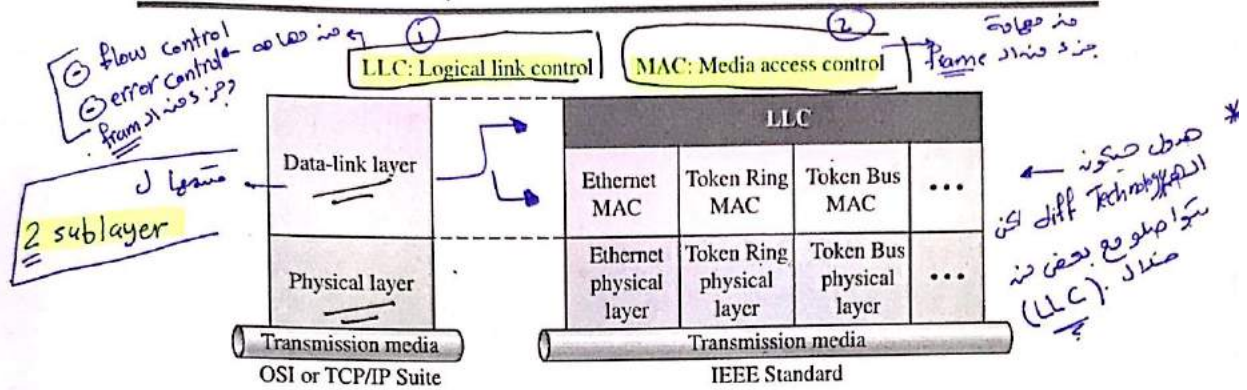
Earlier we discussed data link control. We said that data link control handles framing, flow control, and error control. In IEEE Project 802, flow control, error control,

Project * هاد اد
انه يعني انه يكونه بتدي
[Standard] اجباره
بافتلاف الشركان
الاصنافه لاد Technology
Communication تفصيل
الايكل يكونه قادر على
التواصل بكل اركونه
مع الطرف الاخر.

الاصنافه Protocol
عدد لاد 2
Physical layer
data link

(1)
(2)

Figure 13.1 IEEE standard for LANs



part of the framing duties are collected into one sublayer called the *logical link control* (LLC). Framing is handled in both the LLC sublayer and the MAC sublayer.

The LLC provides a single link-layer control protocol for all IEEE LANs. This means LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent.

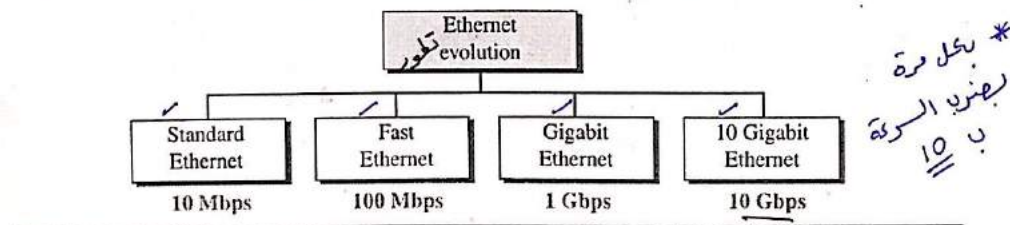
Media Access Control (MAC)

Earlier we discussed multiple access methods including *random access*, *controlled access*, and *channelization*. IEEE Project 802 has created a sublayer called *media access control* that defines the specific access method for each LAN. For example, it defines *CSMA/CD* as the media access method for Ethernet LANs and defines the token-passing method for Token Ring and Token Bus LANs. As we mentioned in the previous section, part of the framing function is also handled by the MAC layer.

13.1.2 [Ethernet Evolution] :-

The Ethernet LAN was developed in the 1970s by Robert Metcalfe and David Boggs. Since then, it has gone through four generations: **Standard Ethernet** (10 Mbps), **Fast Ethernet** (100 Mbps), **Gigabit Ethernet** (1 Gbps), and **10 Gigabit Ethernet** (10 Gbps), as shown in Figure 13.2. We briefly discuss all these generations.

Figure 13.2 Ethernet evolution through four generations



التي بكل مرة يغير السرعة :- كما زيادة الـ BW [من خلال تطور التكنولوجيا]
 * عدد الـ [physical layer] للجهاز السنتايز. فصار عندك تطور
 ما سيقايز صاه الـ BW الـ layer التي آتت انه زياد الـ BW

13.2 STANDARD ETHERNET

We refer to the original Ethernet technology with the data rate of 10 Mbps as the *Standard Ethernet*. Although most implementations have moved to other technologies in the Ethernet evolution, there are some features of the Standard Ethernet that have not changed during the evolution. We discuss this standard version to pave the way for understanding the other three technologies.

13.2.1 Characteristics

Let us first discuss some characteristics of the Standard Ethernet.

Connectionless and Unreliable Service

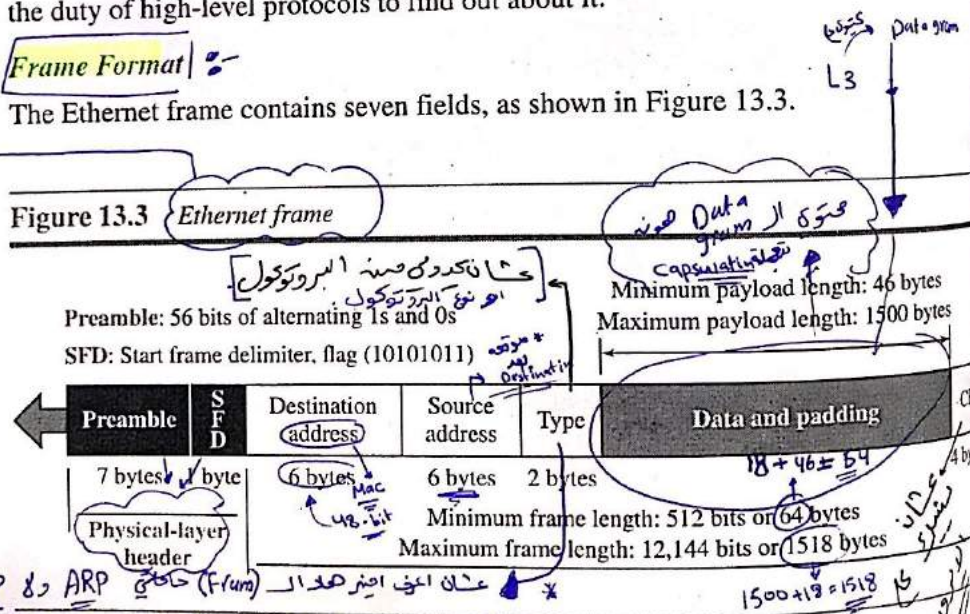
Ethernet provides a connectionless service, which means each frame sent is independent of the previous or next frame. Ethernet has no connection establishment or connection termination phases. The sender sends a frame whenever it has it; the receiver may or may not be ready for it. The sender may overwhelm the receiver with frames, which may result in dropping frames. If a frame drops, the sender will not know about it. Since IP, which is using the service of Ethernet, is also connectionless, it will not know about it either. If the transport layer is also a connectionless protocol, such as UDP, the frame is lost and its salvation may only come from the application layer. However, if the transport layer is TCP, the sender TCP does not receive acknowledgment for its segment and sends it again.

Ethernet is also unreliable like IP and UDP. If a frame is corrupted during transmission and the receiver finds out about the corruption, which has a high level of probability of happening because of the CRC-32, the receiver drops the frame silently. It is the duty of high-level protocols to find out about it.

Frame Format

The Ethernet frame contains seven fields, as shown in Figure 13.3.

Figure 13.3 Ethernet frame



□ **Preamble.** This field contains 7 bytes (56 bits) of alternating 0s and 1s that alert receiving system to the coming frame and enable it to synchronize its clock if it is of synchronization. The pattern provides only an alert and a timing pulse. The 56

pattern allows the stations to miss some bits at the beginning of the frame. The preamble is actually added at the physical layer and is not (formally) part of the frame.

- **Start frame delimiter (SFD).** This field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits are $(11)_2$ and alert the receiver that the next field is the destination address. This field is actually a flag that defines the beginning of the frame. We need to remember that an Ethernet frame is a variable-length frame. It needs a flag to define the beginning of the frame. The SFD field is also added at the physical layer.

عشان نعرف انه حيله ان Real detect

- **Destination address (DA).** This field is six bytes (48 bits) and contains the link-layer address of the destination station or stations to receive the packet. We will discuss addressing shortly. When the receiver sees its own link-layer address, or a multicast address for a group that the receiver is a member of, or a broadcast address, it decapsulates the data from the frame and passes the data to the upper-layer protocol defined by the value of the type field.

- **Source address (SA).** This field is also six bytes and contains the link-layer address of the sender of the packet. We will discuss addressing shortly.

بجد نوع البروتوكول

- **Type.** This field defines the upper-layer protocol whose packet is encapsulated in the frame. This protocol can be IP, ARP, OSPF, and so on. In other words, it serves the same purpose as the protocol field in a datagram and the port number in a segment or user datagram. It is used for multiplexing and demultiplexing.

- **Data.** This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes. We discuss the reason for these minimum and maximum values shortly. If the data coming from the upper layer is more than 1500 bytes, it should be fragmented and encapsulated in more than one frame. If it is less than 46 bytes, it needs to be padded with extra 0s. A padded data frame is delivered to the upper-layer protocol as it is (without removing the padding), which means that it is the responsibility of the upper layer to remove or, in the case of the sender, to add the padding. The upper-layer protocol needs to know the length of its data. For example, a datagram has a field that defines the length of the data.

بعد الاضافة بتغير 64 - ده حايي لان Energy بتاخذ 1518 - اخر الحقل

- **CRC.** The last field contains error detection information, in this case a CRC-32. The CRC is calculated over the addresses, types, and data field. If the receiver calculates the CRC and finds that it is not zero (corruption in transmission), it discards the frame.

بس طول اقل ال (Mix) طول عشان اقل (cost) ده يكون بي memory space

Energy لا sense ال career (sense) Multiple Access / Collision Detection

Frame Length : ال عني - Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame. The minimum length restriction is required for the correct operation of CSMA/CD, as we will see shortly. An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is $64 - 18 = 46$ bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference.

[D MAC] ال ال field ال ال Frame

The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes. The maximum length restriction has two historical reasons. First, memory was very expensive when Ethernet was designed; a maximum length restriction helped to reduce the size of the buffer. Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Minimum frame length: 64 bytes ✓	Minimum data length: 46 bytes ✓
Maximum frame length: 1518 bytes ✓	Maximum data length: 1500 bytes ✓

Handwritten note: 18 bytes

13.2.2 Addressing

Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the station with a link-layer address. The Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes. For example, the following shows an Ethernet MAC address:

4A:30:10:21:10:1A

Transmission of Address Bits

The way the addresses are sent out online is different from the way they are written in hexadecimal notation. The transmission is left to right, byte by byte; however, for each byte, the least significant bit is sent first and the most significant bit is sent last. This means that the bit that defines an address as unicast or multicast arrives first at the receiver. This helps the receiver to immediately know if the packet is unicast or multicast.

Example 13.1

Show how the address 47:20:1B:2E:08:EE is sent out online.

Solution

The address is sent left to right, byte by byte. For each byte, it is sent right to left, bit by bit, as shown below:

Hexadecimal	47	20	1B	2E	08	EE
Binary	01000111	00100000	00011011	00101110	00001000	11101110
Transmitted	11100010	00000100	11011000	01110100	00010000	01110110

Handwritten notes: Most sig, least, DMAC

Unicast, Multicast, and Broadcast Addresses

A source address is always a unicast address—the frame comes from only one station. The destination address, however, can be unicast, multicast, or broadcast. Figure 13-1 shows how to distinguish a unicast address from a multicast address. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.

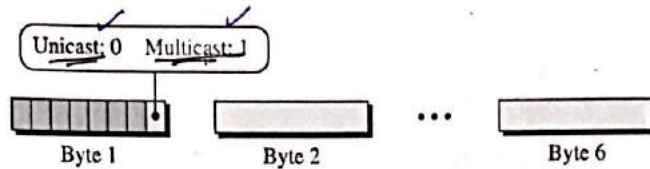
Note that with the way the bits are transmitted, the unicast/multicast bit is the first bit which is transmitted or received. The broadcast address is a special case of a

Handwritten notes:
 11100010
 1110 0010
 1110 0010
 1110 0010
 1110 0010

Handwritten notes:
 * في كل كلمة اول بتة هي بتة DMAC
 اول بتة هي بتة DMAC
 اذا الـ 0 بتة هي unicast
 اذا الـ 1 بتة هي Multicast

Handwritten notes:
 * بتة DMAC
 * بتة DMAC
 * بتة DMAC
 * بتة DMAC

Figure 13.4 Unicast and multicast addresses



multicast address: the recipients are all the stations on the LAN. A broadcast destination address is forty-eight 1s.

Example 13.2

Define the type of the following destination addresses:

- ✓ a. 4A:30:10:21:10:1A
- ✓ b. 47:20:1B:2E:08:EE
- ✓ c. FF:FF:FF:FF:FF:FF

} unicast
 } multicast
 } broadcast

*** Solution**

To find the type of the address, we need to look at the second hexadecimal digit from the left. If it is even, the address is unicast. If it is odd, the address is multicast. If all digits are Fs, the address is broadcast. Therefore, we have the following:

- a. This is a unicast address because A in binary is 10(0)(even).
- b. This is a multicast address because 7 in binary is 011(1)(odd).
- c. This is a broadcast address because all digits are Fs in hexadecimal.

Distinguish Between Unicast, Multicast, and Broadcast Transmission

Standard Ethernet uses a coaxial cable (bus topology) or a set of twisted-pair cables with a hub (star topology) as shown in Figure 13.5.

We need to know that transmission in the standard Ethernet is always broadcast, no matter if the intention is unicast, multicast, or broadcast. In the bus topology, when station A sends a frame to station B, all stations will receive it. In the star topology, when station A sends a frame to station B, the hub will receive it. Since the hub is a passive element, it does not check the destination address of the frame; it regenerates the bits (if they have been weakened) and sends them to all stations except station A. In fact, it floods the network with the frame.

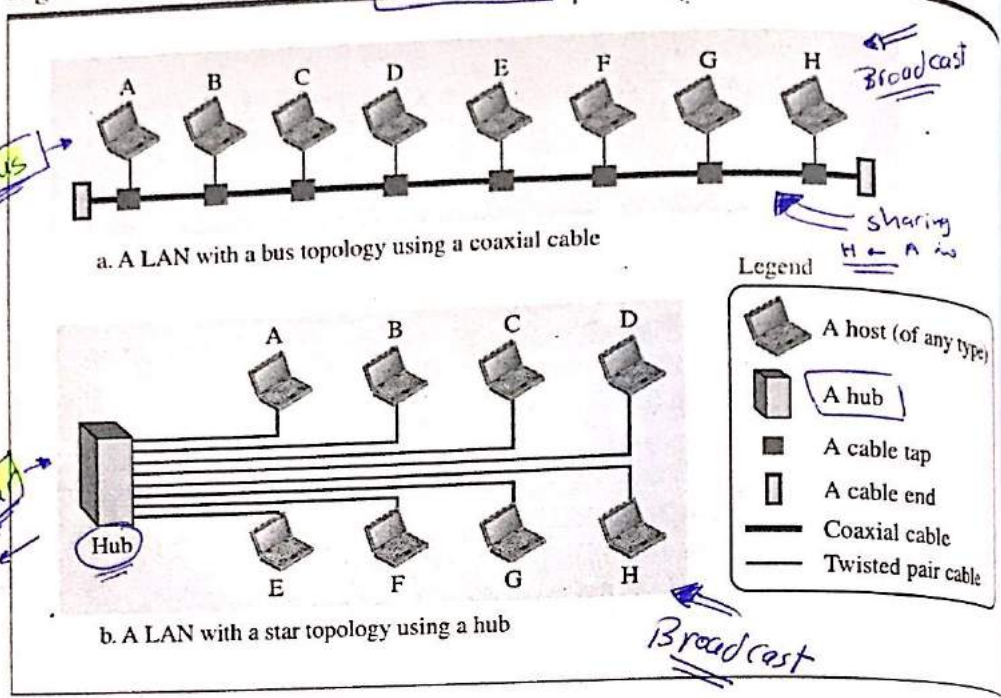
The question is, then, how the actual unicast, multicast, and broadcast transmissions are distinguished from each other. The answer is in the way the frames are kept or dropped.

- In a unicast transmission, all stations will receive the frame, the intended recipient keeps and handles the frame; the rest discard it.
- In a multicast transmission, all stations will receive the frame, the stations that are members of the group keep and handle it; the rest discard it.

Figure 13.5 Implementation of standard Ethernet [10Mbps]

Handwritten notes on the left side of the page:

- switch → 2-layer
- Router → layer 3
- Hub → 1-layer device
- Address



Handwritten notes on the left side of the page:

- Bus
- Star
- Hub
- connected nodes
- passive device

- In a broadcast transmission, all stations (except the sender) will receive the frame and all stations (except the sender) keep and handle it.

13.2.3 Access Method :-

Since the network that uses the standard Ethernet protocol is a broadcast network we need to use an access method to control access to the sharing medium. The standard Ethernet chose CSMA/CD with 1-persistent method, discussed earlier in Chapter 12, Section 1.3. Let us use a scenario to see how this method works for the standard Ethernet protocol.

- Assume station A in Figure 13.5 has a frame to send to station D. Station A should check whether any other station is sending (carrier sense). Station A measures the level of energy on the medium (for a short period of time, normally less than 100 μs). If there is no signal energy on the medium, it means that no station is sending (or the signal has not reached station A). Station A interprets this situation as idle medium. It starts sending its frame. On the other hand, if the signal energy level is not zero, it means that the medium is being used by another station. Station A continuously monitors the medium until it becomes idle for 100 μs. It then starts sending the frame. However, station A needs to keep a copy of the frame in its buffer until it is sure that there is no collision. When station A is sure of this is the subject we discuss next.
- The medium sensing does not stop after station A has started sending the frame. Station A needs to send and receive continuously. Two cases may occur:

- a. Station A has sent 512 bits and no collision is sensed (the energy level did not go above the regular energy level), the station then is sure that the frame will go through and stops sensing the medium. Where does the number 512 bits come from? If we consider the transmission rate of the Ethernet as 10 Mbps, this means that it takes the station $512/(10 \text{ Mbps}) = 51.2 \mu\text{s}$ to send out 512 bits. With the speed of propagation in a cable (2×10^8 meters), the first bit could have gone 10,240 meters (one way) or only 5120 meters (round trip), have collided with a bit from the last station on the cable, and have gone back. In other words, if a collision were to occur, it should occur by the time the sender has sent out 512 bits (worst case) and the first bit has made a round trip of 5120 meters. We should know that if the collision happens in the middle of the cable, not at the end, station A hears the collision earlier and aborts the transmission. We also need to mention another issue. The above assumption is that the length of the cable is 5120 meters. The designer of the standard Ethernet actually put a restriction of 2500 meters because we need to consider the delays encountered throughout the journey. It means that they considered the worst case. The whole idea is that if station A does not sense the collision before sending 512 bits, there must have been no collision, because during this time, the first bit has reached the end of the line and all other stations know that a station is sending and refrain from sending. In other words, the problem occurs when another station (for example, the last station) starts sending before the first bit of station A has reached it. The other station mistakenly thinks that the line is free because the first bit has not yet reached it. The reader should notice that the restriction of 512 bits actually helps the sending station: The sending station is certain that no collision will occur if it is not heard during the first 512 bits, so it can discard the copy of the frame in its buffer.
- b. Station A has sensed a collision before sending 512 bits. This means that one of the previous bits has collided with a bit sent by another station. In this case both stations should refrain from sending and keep the frame in their buffer for resending when the line becomes available. However, to inform other stations that there is a collision in the network, the station sends a 48-bit jam signal. The jam signal is to create enough signal (even if the collision happens after a few bits) to alert other stations about the collision. After sending the jam signal, the stations need to increment the value of K (number of attempts). If after increment $K = 15$, the experience has shown that the network is too busy, the station needs to abort its effort and try again. If $K < 15$, the station can wait a backoff time (T_B in Figure 12.13) and restart the process. As Figure 12.13 shows, the station creates a random number between 0 and $2^K - 1$, which means each time the collision occurs, the range of the random number increases exponentially. After the first collision ($K = 1$) the random number is in the range (0, 1). After the second collision ($K = 2$) it is in the range (0, 1, 2, 3). After the third collision ($K = 3$) it is in the range (0, 1, 2, 3, 4, 5, 6, 7). So after each collision, the probability increases that the backoff time becomes longer. This is due to the fact that if the collision happens even after the third or fourth attempt, it means that the network is really busy; a longer backoff time is needed.

13.2.4 Efficiency of Standard Ethernet

The efficiency of the Ethernet is defined as the ratio of the time used by a station to send data to the time the medium is occupied by this station. The practical efficiency of standard Ethernet has been measured to be

$$\text{Efficiency} = 1 / (1 + 6.4 \times a)$$

in which the parameter "a" is the number of frames that can fit on the medium. It can be calculated as $a = (\text{propagation delay}) / (\text{transmission delay})$ because the transmission delay is the time it takes a frame of average size to be sent out and the propagation delay is the time it takes to reach the end of the medium. Note that as the value of parameter "a" decreases, the efficiency increases. This means that if the length of the media is shorter or the frame size longer, the efficiency increases. In the ideal case, $a = 0$ and the efficiency is 1. We ask to calculate this efficiency in problems at the end of the chapter.

حساب
Calculation
الحساب

Example 13.3

In the Standard Ethernet with the transmission rate of 10 Mbps, we assume that the length of the medium is 2500 m and the size of the frame is 512 bits. The propagation speed of a signal in cable is normally 2×10^8 m/s.

$$\text{Propagation delay} = 2500 / (2 \times 10^8) = 12.5 \mu\text{s}$$

$$\text{Transmission delay} = 512 / (10^7) = 51.2 \mu\text{s}$$

$$\text{Efficiency} = 39\%$$

$$a = 12.5 / 51.2 = 0.24$$

The example shows that $a = 0.24$, which means only 0.24 of a frame occupies the whole medium in this case. The efficiency is 39 percent, which is considered moderate; it means that only 61 percent of the time the medium is occupied but not used by a station.

13.2.5 Implementation

The Standard Ethernet defined several implementations, but only four of them became popular during the 1980s. Table 13.1 shows a summary of Standard Ethernet implementations.

Table 13.1 Summary of Standard Ethernet implementations

Implementation	Medium	Medium Length	Encoding
10Base5	Thick coax	500 m	Manchester
10Base2	Thin coax	185 m	Manchester
10Base-T	2 UTP	100 m	Manchester
10Base-F	2 Fiber	2000 m	Manchester

10Mbps Rate
Base Band
Base Band
Modulation
Analog
Technology
Length

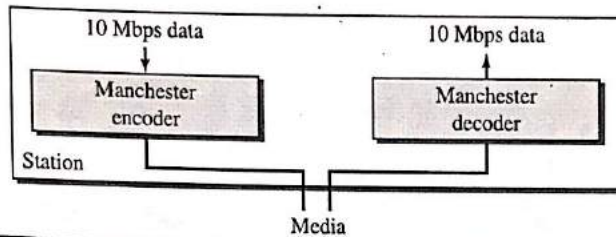
In the nomenclature 10BaseX, the number defines the data rate (10 Mbps), the term Base means baseband (digital) signal, and X approximately defines either the maximum size of the cable in 100 meters (for example 5 for 500 or 2 for 185 meters) or the type of cable, T for unshielded twisted pair cable (UTP) and F for fiber-optic. The standard Ethernet uses a baseband signal, which means that the bits are changed to digital signal and directly sent on the line.

* Thick → ...
Thin → ...

Encoding and Decoding

All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the received signal is interpreted as Manchester and decoded into data. As we saw in Chapter 4, Manchester encoding is self-synchronous, providing a transition at each bit interval. Figure 13.6 shows the encoding scheme for Standard Ethernet.

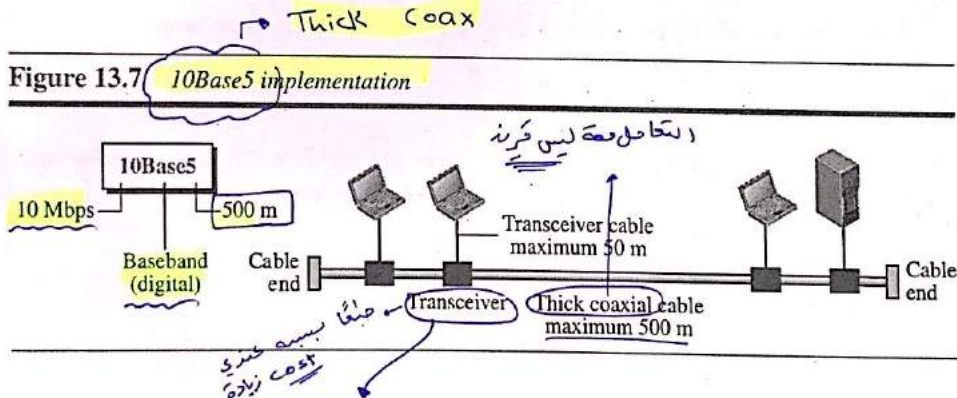
Figure 13.6 Encoding in a Standard Ethernet implementation



10Base5: Thick Ethernet

صافق لينة

The first implementation is called **10Base5**, **(thick) Ethernet**, or **Thicknet**. The nickname derives from the size of the cable, which is roughly the size of a garden hose and too stiff to bend with your hands. 10Base5 was the first Ethernet specification to use a bus topology with an external transceiver (transmitter/receiver) connected via a tap to a thick coaxial cable. Figure 13.7 shows a schematic diagram of a 10Base5 implementation.



The transceiver is responsible for transmitting, receiving, and detecting collisions. The transceiver is connected to the station via a transceiver cable that provides separate paths for sending and receiving. This means that collision can only happen in the coaxial cable.

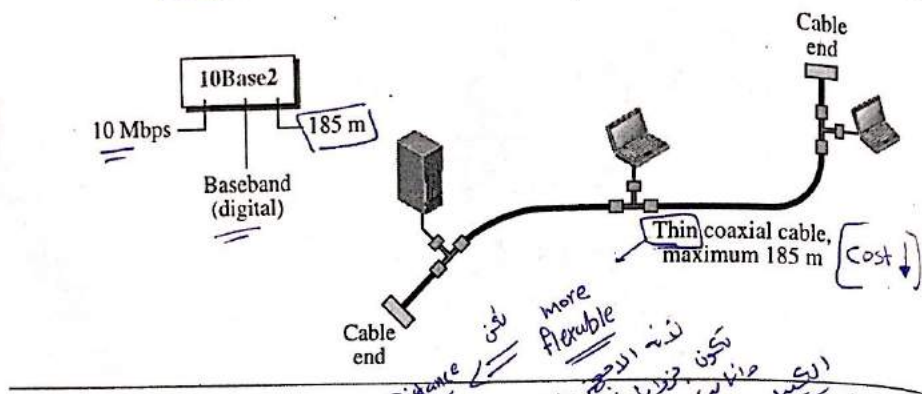
The maximum length of the coaxial cable must not exceed 500 m, otherwise, there is excessive degradation of the signal. If a length of more than 500 m is needed, up to five segments, each a maximum of 500 meters, can be connected using repeaters. Repeater will be discussed in Chapter 17.

*
 طول كل اقل اقل
 Repeater
 كمنصة
 15) Repeater
 منة به

10Base2: Thin Ethernet

The second implementation is called **10Base2, thin Ethernet, or Cheapernet**. 10Base2 also uses a bus topology, but the cable is much thinner and more flexible. The cable can be bent to pass very close to the stations. In this case, the transceiver is normally part of the network interface card (NIC), which is installed inside the station. Figure 13.8 shows the schematic diagram of a 10Base2 implementation.

Figure 13.8 10Base2 implementation

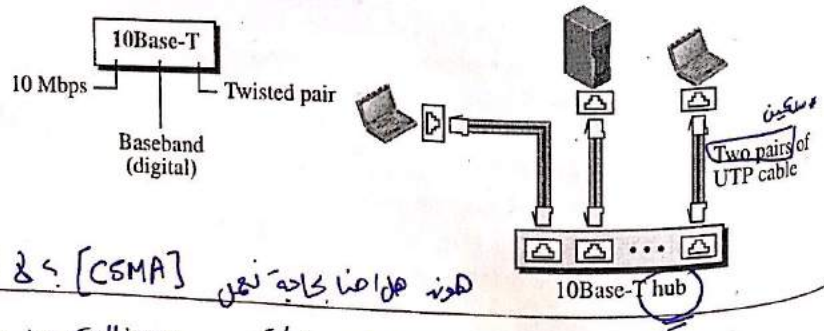


Note that the collision here occurs in the thin coaxial cable. This implementation is more cost effective than 10Base5 because thin coaxial cable is less expensive than thick coaxial and the tee connections are much cheaper than taps. Installation is simple because the thin coaxial cable is very flexible. However, the length of each segment cannot exceed 185 m (close to 200 m) due to the high level of attenuation in thin coaxial cable.

10Base-T: Twisted-Pair Ethernet

The third implementation is called **10Base-T or twisted-pair Ethernet**. 10Base-T uses a physical star topology. The stations are connected to a hub via two pairs of twisted cable, as shown in Figure 13.9.

Figure 13.9 10Base-T implementation



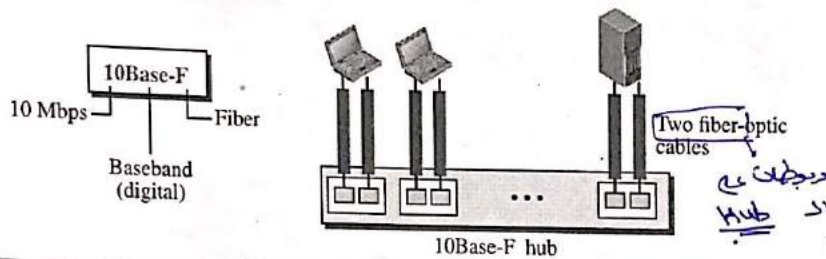
Handwritten notes in Arabic: 'Sharing' (مشاركة), 'CSMA' (CSMA), 'Collision' (اصطدام), 'بما اننا نحتاج لنقل' (because we need to transfer), 'بما اننا نحتاج لنقل' (because we need to transfer), 'بما اننا نحتاج لنقل' (because we need to transfer).

Note that two pairs of twisted cable create two paths (one for sending and one for receiving) between the station and the hub. Any collision here happens in the hub. Compared to 10Base5 or 10Base2, we can see that the hub actually replaces the coaxial cable as far as a collision is concerned. The maximum length of the twisted cable here is defined as 100 m, to minimize the effect of attenuation in the twisted cable.

10Base-F: Fiber Ethernet

Although there are several types of optical fiber 10-Mbps Ethernet, the most common is called **10Base-F**. 10Base-F uses a star topology to connect stations to a hub. The stations are connected to the hub using two fiber-optic cables, as shown in Figure 13.10.

Figure 13.10 **10Base-F implementation**



13.2.6 Changes in the Standard

يعني بتو الاسباب التي ممكنة الطرف انه
Higher

Before we discuss higher-rate Ethernet protocols, we need to discuss the changes that occurred to the 10-Mbps Standard Ethernet. These changes actually opened the road to the evolution of the Ethernet to become compatible with other high-data-rate LANs.

Bridged Ethernet

The first step in the Ethernet evolution was the division of a LAN by **bridges**. Bridges have two effects on an Ethernet LAN: They raise the bandwidth and they separate collision domains. We discuss bridges in Chapter 17.

Raising the Bandwidth

In an unbridged Ethernet network, the total capacity (10 Mbps) is shared among all stations with a frame to send; the stations share the bandwidth of the network. If only one station has frames to send, it benefits from the total capacity (10 Mbps). But if more than one station needs to use the network, the capacity is shared. For example, if two stations have a lot of frames to send, they probably alternate in usage. When one station is sending, the other one refrains from sending. We can say that, in this case, each station on average sends at a rate of 5 Mbps. Figure 13.11 shows the situation.

The bridge, as we will learn in Chapter 17, can help here. A bridge divides the network into two or more networks. Bandwidthwise, each network is independent. For example, in Figure 13.12, a network with 12 stations is divided into two networks, each with 6 stations. Now each network has a capacity of 10 Mbps. The 10-Mbps capacity in each segment is now shared between 6 stations (actually 7 because the bridge acts as a

Figure 13.11 [Sharing bandwidth]

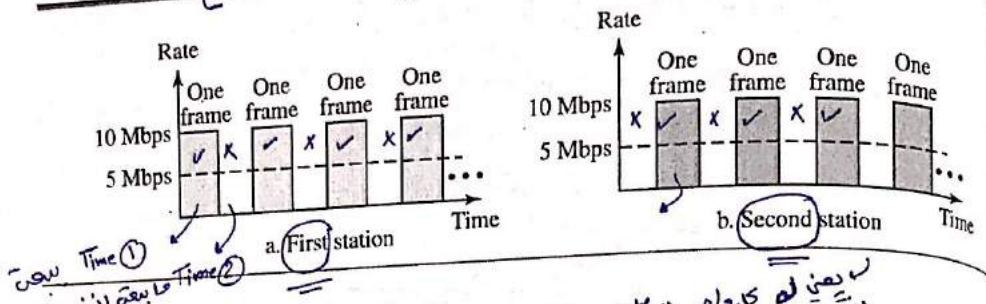
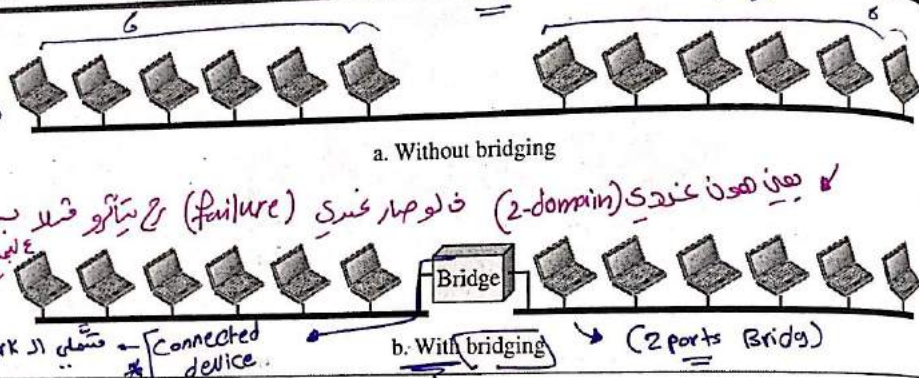


Figure 13.12 A network with and without a bridge



Handwritten notes in Arabic: "منذ صفة بي اى ار", "10M", "12", "بعض صفة بي اى ار".

Handwritten notes in Arabic: "بعض صفة بي اى ار", "Rate", "بعض صفة بي اى ار", "بعض صفة بي اى ار", "بعض صفة بي اى ار".

Handwritten notes in Arabic: "[2 segment] network", "بعض صفة بي اى ار", "Device 12", "بعض صفة بي اى ار", "بعض صفة بي اى ار", "بعض صفة بي اى ار", "بعض صفة بي اى ار".

station in each segment), not 12 stations. In a network with a heavy load, each station theoretically is offered $10/7$ Mbps instead of $10/12$ Mbps.

It is obvious that if we further divide the network, we can gain more bandwidth for each segment. For example, if we use a four-port bridge, each station is now offered $10/4$ Mbps, which is 3 times more than an unbridged network.

Separating Collision Domains

Another advantage of a bridge is the separation of the collision domain. Figure 13.12 shows the collision domains for an unbridged and a bridged network. You can see that the collision domain becomes much smaller and the probability of collision is reduced tremendously. Without bridging, 12 stations contend for access to the medium; with bridging only 3 stations contend for access to the medium.

Switched Ethernet

The idea of a bridged LAN can be extended to a switched LAN. Instead of having two to four networks, why not have N networks, where N is the number of stations on the LAN? In other words, if we can have a multiple-port bridge, why not have an N -port switch? In this way, the bandwidth is shared only between the station and the switch (5 Mbps each). In addition, the collision domain is divided into N domains.

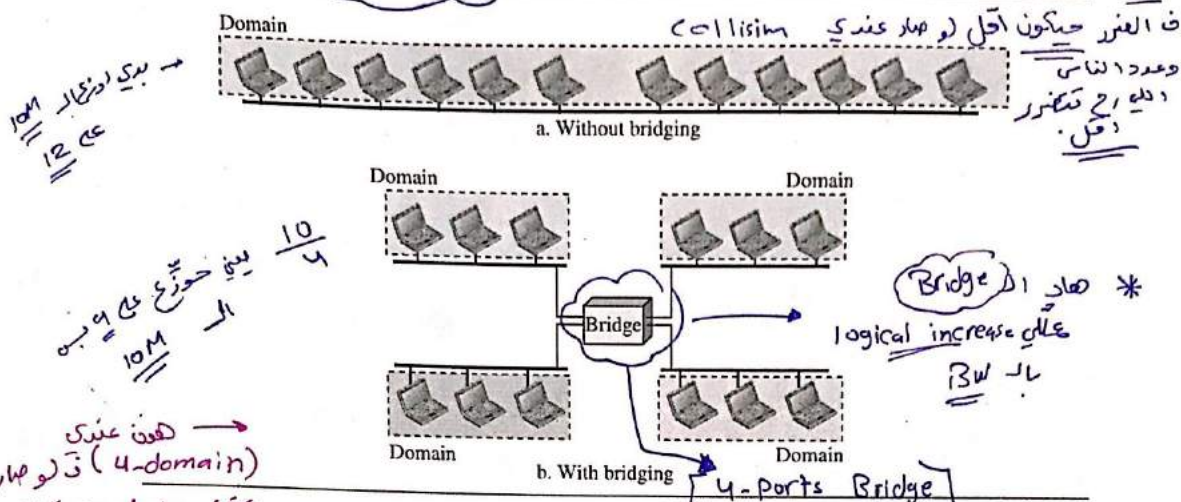
A layer-2 switch is an N -port bridge with additional sophistication that allows fast handling of the packets. Evolution from a bridged Ethernet to a switched Ethernet

Handwritten notes in Arabic: "more collision domain", "more logical BW", "full-duplex switched".

* يعني الـ domain الذي فيه عندي تصادم فيجاء
 * يعني انه Bit لو السيتات تاتي
 على بعضها

* الاحسن يكون عندي اكثر من domain
 يعني في حال هاد عندي ضغط domain ما يتاخر
 ارباع

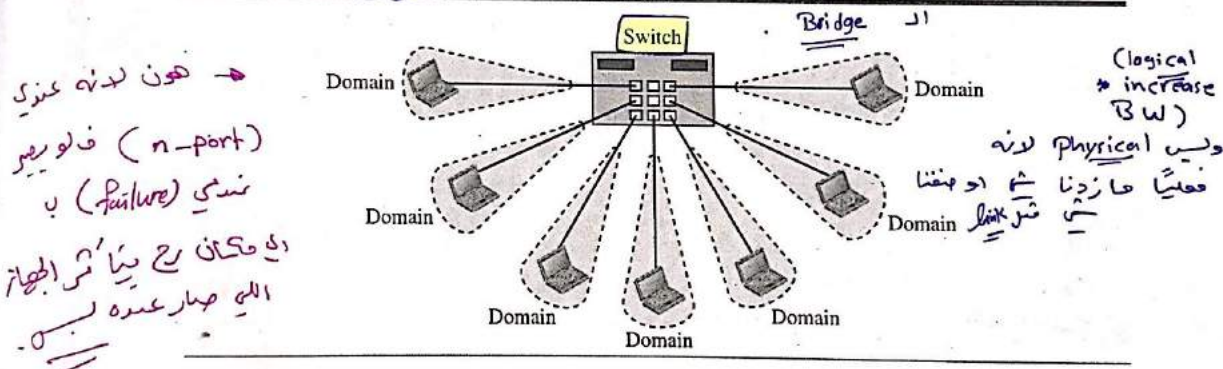
Figure 13.13 Collision domains in an unbridged network and a bridged network



بدي ارباع 10M
 على 12
 يعني حورج على 9 بيه
 الـ 10M
 كهن عندي
 4-domain
 عندي failure
 مع 3

a big step that opened the way to an even faster Ethernet, as we will see. Figure 13.14 shows a switched LAN.

Figure 13.14 Switched Ethernet



هون لونه عندي
 (n-port) فلو بيه
 عندي (failure) ب
 دي ممكن مع بيتا ش الجواز
 الـ هاد عنده ليه

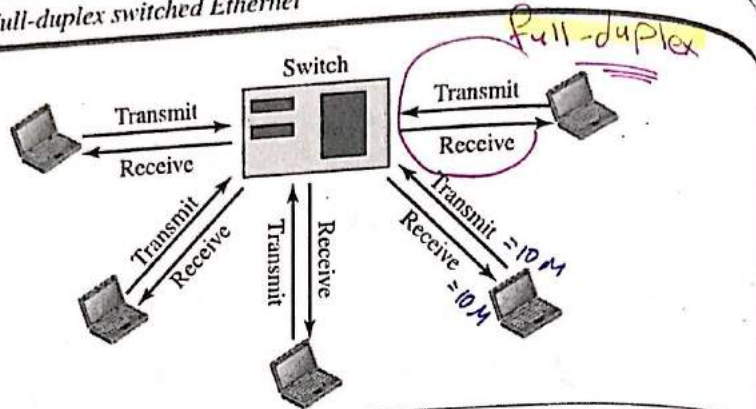
Full-Duplex Ethernet

One of the limitations of 10Base5 and 10Base2 is that communication is half-duplex (10Base-T is always full-duplex); a station can either send or receive, but may not do both at the same time. The next step in the evolution was to move from switched Ethernet to full-duplex switched Ethernet. The full-duplex mode increases the capacity of each domain from 10 to 20 Mbps. Figure 13.15 shows a switched Ethernet in full-duplex mode. Note that instead of using one link between the station and the switch, the configuration uses two links: one to transmit and one to receive.

No Need for CSMA/CD

In full-duplex switched Ethernet, there is no need for the CSMA/CD method. In a full-duplex switched Ethernet, each station is connected to the switch via two separate links.

Figure 13.15 Full-duplex switched Ethernet



Each station or switch can send and receive independently without worrying about collision. Each link is a point-to-point dedicated path between the station and the switch. There is no longer a need for carrier sensing; there is no longer a need for collision detection. The job of the MAC layer becomes much easier. The carrier sensing and collision detection functionalities of the MAC sublayer can be turned off.

MAC Control Layer

Standard Ethernet was designed as a connectionless protocol at the MAC sublayer. There is no explicit flow control or error control to inform the sender that the frame has arrived at the destination without error. When the receiver receives the frame, it does not send any positive or negative acknowledgment.

To provide for flow and error control in full-duplex switched Ethernet, a new sublayer, called the *MAC control*, is added between the LLC sublayer and the MAC sublayer.

- 1) Bridge
 - 2) Switch
 - 3) Full-duplex
- (n-port bridge)

كلمة fast في الـ standard
 في الـ standard
 update

13.3 FAST ETHERNET (100 MBPS) :-

In the 1990s, some LAN technologies with transmission rates higher than 10 Mbps such as FDDI and Fiber Channel, appeared on the market. If the Standard Ethernet wanted to survive, it had to compete with these technologies. Ethernet made a big jump by increasing the transmission rate to 100 Mbps, and the new generation was called the *Fast Ethernet*. The designers of the Fast Ethernet needed to make it compatible with the Standard Ethernet. The MAC sublayer was left unchanged, which meant the frame format and the maximum and minimum size could also remain unchanged. By increasing the transmission rate, features of the Standard Ethernet that depend on the transmission rate, access method, and implementation had to be reconsidered. The goals of Fast Ethernet can be summarized as follows:

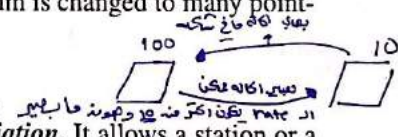
1. Upgrade the data rate to 100 Mbps.
2. Make it compatible with Standard Ethernet.
3. Keep the same 48-bit address.
4. Keep the same frame format.

13.3.1 Access Method :-

We remember that the proper operation of the CSMA/CD depends on the transmission rate, the minimum size of the frame, and the maximum network length. If we want to keep the minimum size of the frame, the maximum length of the network should be changed. In other words, if the minimum frame size is still 512 bits, and it is transmitted 10 times faster, the collision needs to be detected 10 times sooner, which means the maximum length of the network should be 10 times shorter (the propagation speed does not change). So the Fast Ethernet came with two solutions (it can work with either choice):

بعض ال length انه لازم يكون 10 مرات اقصر عند collision

1. The first solution was to totally drop the bus topology and use a passive hub and star topology but make the maximum size of the network 250 meters instead of 2500 meters as in the Standard Ethernet. This approach is kept for compatibility with the Standard Ethernet.
2. The second solution is to use a link-layer switch (discussed later in the chapter) with a buffer to store frames and a full-duplex connection to each host to make the transmission medium private for each host. In this case, there is no need for CSMA/CD because the hosts are not competing with each other. The link-layer switch receives a frame from a source host and stores it in the buffer (queue) waiting for processing. It then checks the destination address and sends the frame out of the corresponding interface. Since the connection to the switch is full-duplex, the destination address can even send a frame to another station at the same time that it is receiving a frame. In other words, the shared medium is changed to many point-to-point media, and there is no need for contention.



Autonegotiation

بعض ال length انه لازم يكون 10 مرات اقصر عند collision

A new feature added to Fast Ethernet is called *autonegotiation*. It allows a station or a hub a range of capabilities. Autonegotiation allows two devices to negotiate the mode or data rate of operation. It was designed particularly to allow incompatible devices to connect to one another. For example, a device with a maximum data rate of 10 Mbps can communicate with a device with a 100 Mbps data rate (but which can work at a lower rate). We can summarize the goal of autonegotiation as follows. It was designed particularly for these purposes:

- To allow incompatible devices to connect to one another. For example, a device with a maximum capacity of 10 Mbps can communicate with a device with a 100 Mbps capacity (but which can work at a lower rate).
- To allow one device to have multiple capabilities.
- To allow a station to check a hub's capabilities.

13.3.2 Physical Layer :-

To be able to handle a 100 Mbps data rate, several changes need to be made at the physical layer.

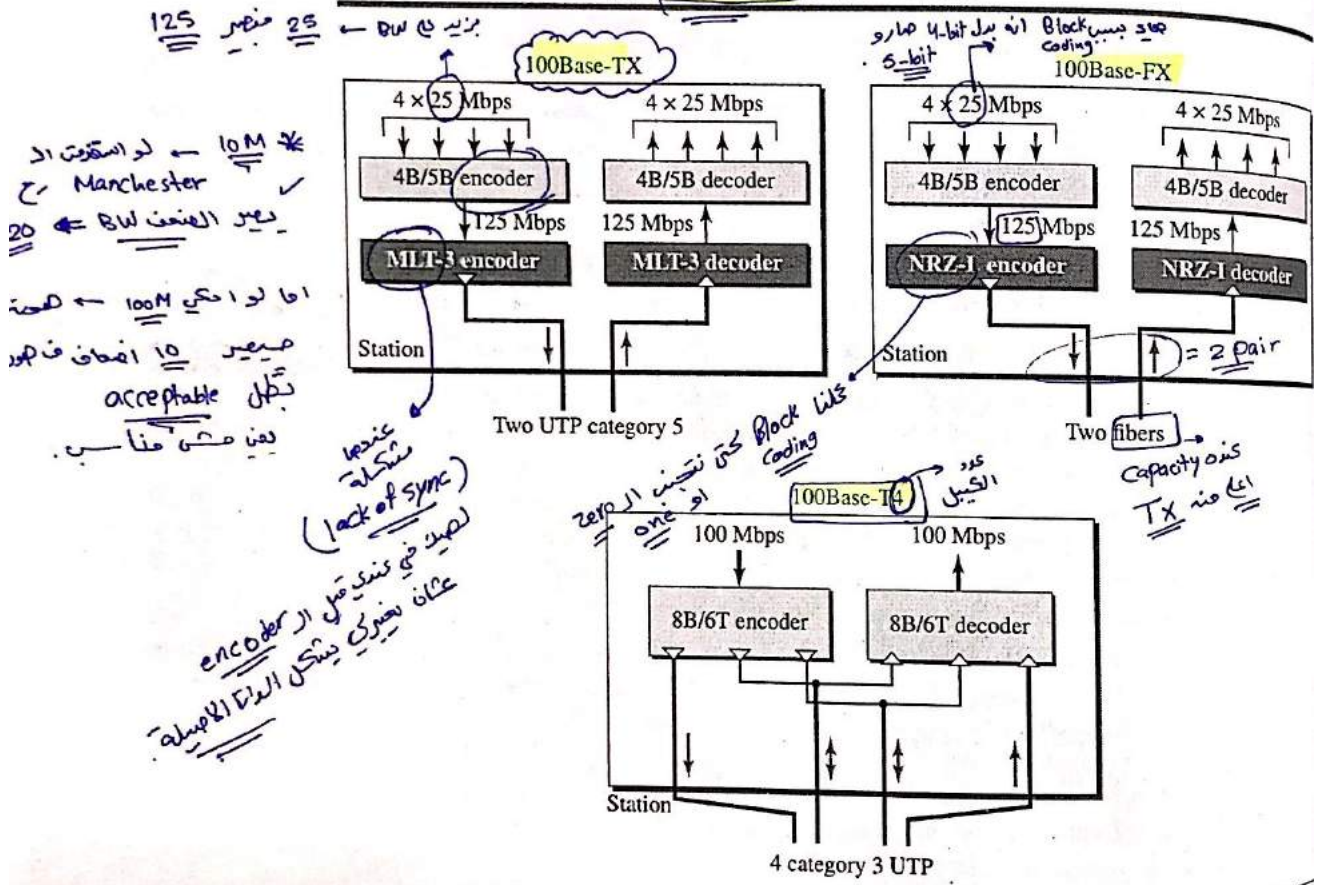
Topology

Fast Ethernet is designed to connect two or more stations. If there are only two stations they can be connected point-to-point. Three or more stations need to be connected in a star topology with a hub or a switch at the center.

Encoding

Manchester encoding needs a 200-Mbaud bandwidth for a data rate of 100 Mbps, which makes it unsuitable for a medium such as twisted-pair cable. For this reason, the Fast Ethernet designers sought some alternative encoding/decoding scheme. However, it was found that one scheme would not perform equally well for all three implementations. Therefore, three different encoding schemes were chosen (see Figure 13.16).

Figure 13.16 Encoding for Fast Ethernet implementation

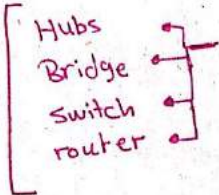


100Base-TX uses two pairs of twisted-pair cable (either category 5 UTP or STP). For this implementation, the MLT-3 scheme was selected since it has good bandwidth performance (see Chapter 4). However, since MLT-3 is not a self-synchronous line coding scheme, 4B/5B block coding is used to provide bit synchronization by preventing

CHAPTER 17

Connecting Devices and Virtual LANs :-

Handwritten notes: "Switching" and "متراب" (connected).



Hosts or LANs do not normally operate in isolation. They are connected to one another or to the Internet. To connect hosts or LANs, we use connecting devices. Connecting devices can operate in different layers of the Internet model. After discussing some connecting devices, we show how they are used to create virtual local area networks (VLANs).

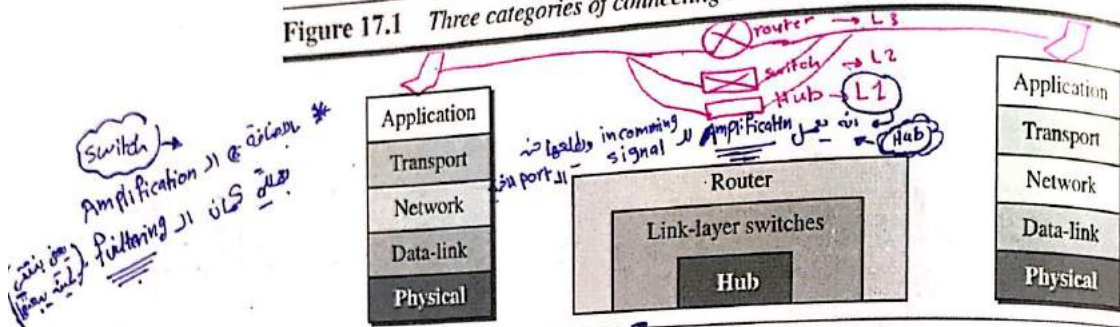
The chapter is divided into two sections.

- The first section discusses connecting devices. It first describes hubs and their features. The section then discusses link-layer switches (or simply *switches*, as they are called), and shows how they can create loops if they connect LANs with broadcast domains.
- The second section discusses virtual LANs or VLANs. The section first shows how membership in a VLAN can be defined. The section then discusses the VLAN configuration. It next shows how switches can communicate in a VLAN. Finally, the section mentions the advantages of a VLAN.

17.1 CONNECTING DEVICES

Hosts and networks do not normally operate in isolation. We use connecting devices to connect hosts together to make a network or to connect networks together to make an internet. Connecting devices can operate in different layers of the Internet model. We discuss three kinds of *connecting devices*: hubs, link-layer switches, and routers. Hubs today operate in the first layer of the Internet model. Link-layer switches operate in the first two layers. Routers operate in the first three layers. (See Figure 17.1.)

Figure 17.1 Three categories of connecting devices



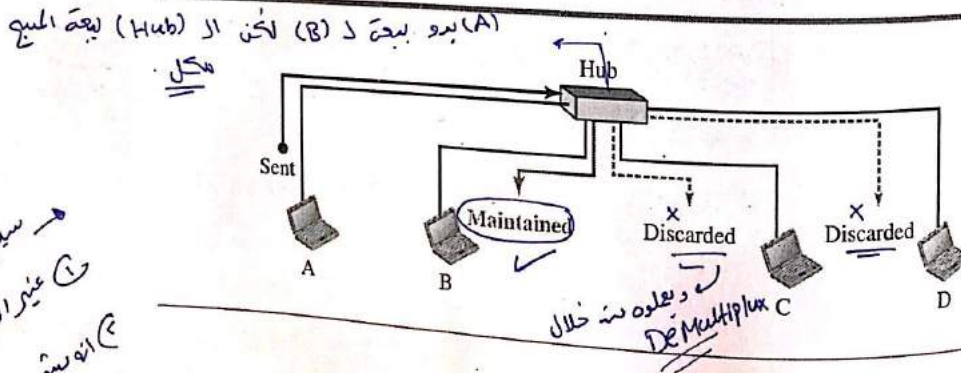
Switch
 * العناية في
 Amplification
 Filtering
 من بين (التي) من بين

Router
 معطى ال router
 بشكل ال Hub
 switch
 معطى ال Hub
 بشكل ال switch
 معطى ال Hub
 بشكل ال switch
 معطى ال Hub
 بشكل ال switch

17.1.1 Hubs

A hub is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A repeater receives a signal and, before it becomes too weak or corrupted, regenerates and retimes the original bit pattern. The repeater then sends the refreshed signal. In the past, when Ethernet LANs were using bus topology, a repeater was used to connect two segments of a LAN to overcome the length restriction of the coaxial cable. Today, however, Ethernet LANs use star topology. In a star topology, a repeater is a multiport device, often called a *hub*, that can be used to serve as the connecting point and at the same time function as a repeater. Figure 17.2 shows that when a packet from station A to station B arrives at the hub, the signal representing the frame is regenerated to remove any possible corrupting noise, but the hub forwards the

Figure 17.2 A hub



security
 Device
 Traffic
 من بين (التي) من بين

packet from all outgoing ports except the one from which the signal was received. In other words, the frame is broadcast. All stations in the LAN receive the frame, but only station B keeps it. The rest of the stations discard it. Figure 17.2 shows the role of a repeater or a hub in a switched LAN.

The figure definitely shows that a hub does not have a filtering capability; it does not have the intelligence to find from which port the frame should be sent out.

A repeater has no filtering capability. *

A hub or a repeater is a physical-layer device. They do not have a link-layer address and they do not check the link-layer address of the received frame. They just regenerate the corrupted bits and send them out from every port.

17.1.2 Link-Layer Switches

A link-layer switch (or switch) operates in both the physical and the data-link layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the link-layer switch can check the MAC addresses (source and destination) contained in the frame.

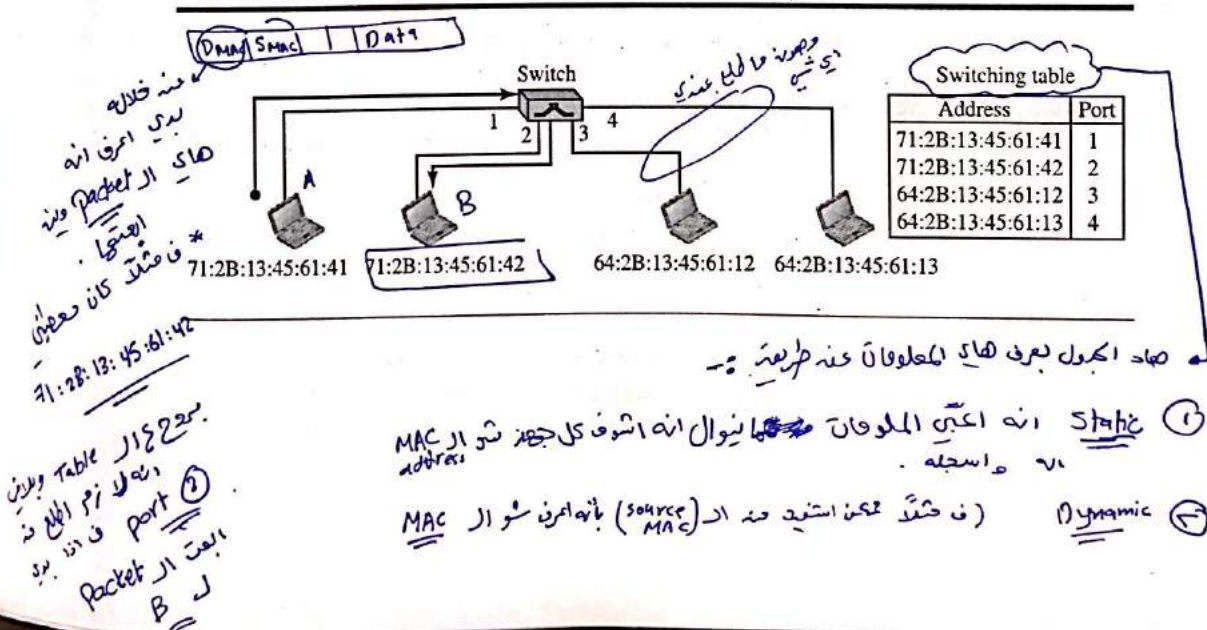
Filtering

One may ask what the difference in functionality is between a link-layer switch and a hub. A link-layer switch has filtering capability. It can check the destination address of a frame and can decide from which outgoing port the frame should be sent.

A link-layer switch has a table used in filtering decisions.

Let us give an example. In Figure 17.3, we have a LAN with four stations that are connected to a link-layer switch. If a frame destined for station 71:2B:13:45:61:42 arrives at port 1, the link-layer switch consults its table to find the departing port. According to its table, frames for 71:2B:13:45:61:42 should be sent out only through port 2; therefore, there is no need for forwarding the frame through other ports.

Figure 17.3 Link-layer switch



A link-layer switch does not change the link-layer (MAC) addresses in a frame.

Transparent Switches

A transparent switch is a switch in which the stations are completely unaware of the switch's existence. If a switch is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent switches must meet three criteria:

- Frames must be forwarded from one station to another.
- The forwarding table is automatically made by learning frame movements in the network.
- Loops in the system must be prevented.

Forwarding

A transparent switch must correctly forward the frames, as discussed in the previous section.

Learning

The earliest switches had switching tables that were static. The system administrator would manually enter each table entry during switch setup. Although the process was simple, it was not practical. If a station was added or deleted, the table had to be modified manually. The same was true if a station's MAC address changed, which is not a rare event. For example, putting in a new network card means a new MAC address.

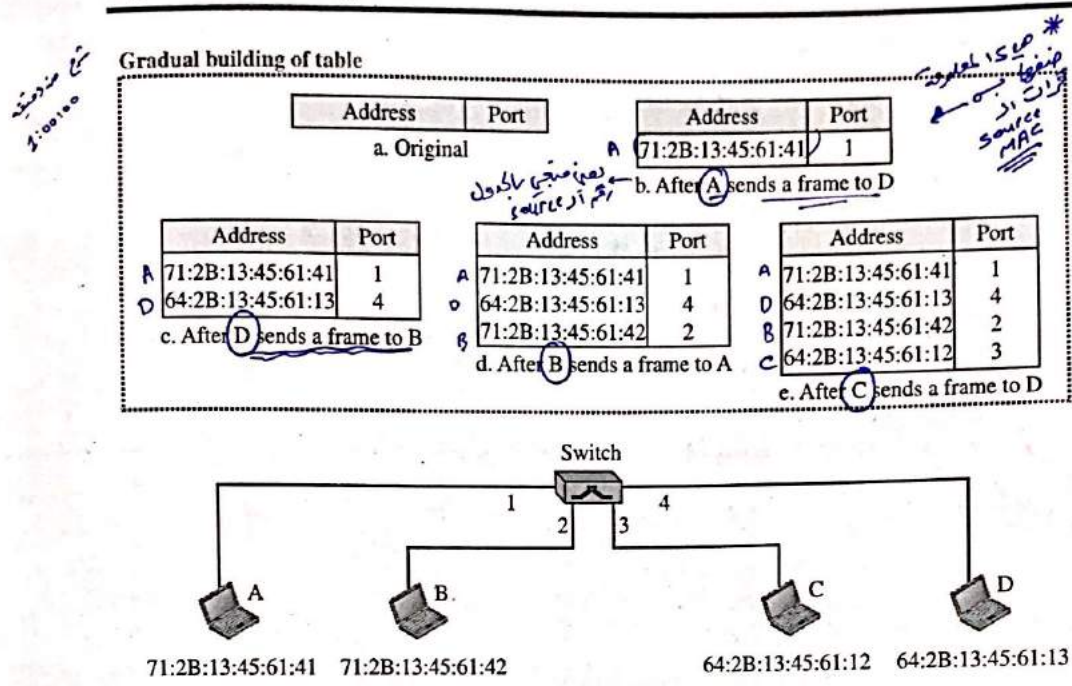
A better solution to the static table is a dynamic table that maps addresses to ports (interfaces) automatically. To make a table dynamic, we need a switch that gradually learns from the frames' movements. To do this, the switch inspects both the destination and the source addresses in each frame that passes through the switch. The destination address is used for the forwarding decision (table lookup); the source address is used for adding entries to the table and for updating purposes. Let us elaborate on this process using Figure 17.4.

1. When station A sends a frame to station D, the switch does not have an entry for either D or A. The frame goes out from all three ports; the frame floods the network. However, by looking at the source address, the switch learns that station A must be connected to port 1. This means that frames destined for A, in the future, must be sent out through port 1. The switch adds this entry to its table. The table has its first entry now.
2. When station D sends a frame to station B, the switch has no entry for B, so it floods the network again. However, it adds one more entry to the table related to station D.
3. The learning process continues until the table has information about every port. However, note that the learning process may take a long time. For example, if a station does not send out a frame (a rare situation), the station will never have an entry in the table.

Loop Problem

Transparent switches work fine as long as there are no redundant switches in the system. Systems administrators, however, like to have redundant switches (more than one switch between a pair of LANs) to make the system more reliable. If a switch fails, another switch takes over until the failed one is repaired or replaced. Redundancy can create loops in the system, which is very undesirable. Loops can be created only when

Figure 17.4 Learning switch



two or more broadcasting LANs (those using hubs, for example) are connected by more than one switch.

Figure 17.5 shows a very simple example of a loop created in a system with two LANs connected by two switches.

1. Station A sends a frame to station D. The tables of both switches are empty. Both forward the frame and update their tables based on the source address A.
2. Now there are two copies of the frame on LAN 2. The copy sent out by the left switch is received by the right switch, which does not have any information about the destination address D; it forwards the frame. The copy sent out by the right switch is received by the left switch and is sent out for lack of information about D. Note that each frame is handled separately because switches, as two nodes on a broadcast network sharing the medium, use an access method such as CSMA/CD. The tables of both switches are updated, but still there is no information for destination D.
3. Now there are two copies of the frame on LAN 1. Step 2 is repeated, and both copies are sent to LAN2.
4. The process continues on and on. Note that switches are also repeaters and regenerate frames. So in each iteration, there are newly generated fresh copies of the frames.

Spanning Tree Algorithm :-

To solve the looping problem, the IEEE specification requires that switches use the spanning tree algorithm to create a loopless topology. In graph theory, a spanning

Figure 17.5 Loop problem in a learning switch

broadcast

ميت الينج اليه (A) كيج

دليل الينج اليه (A) كيج

Table دى ديلا

دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

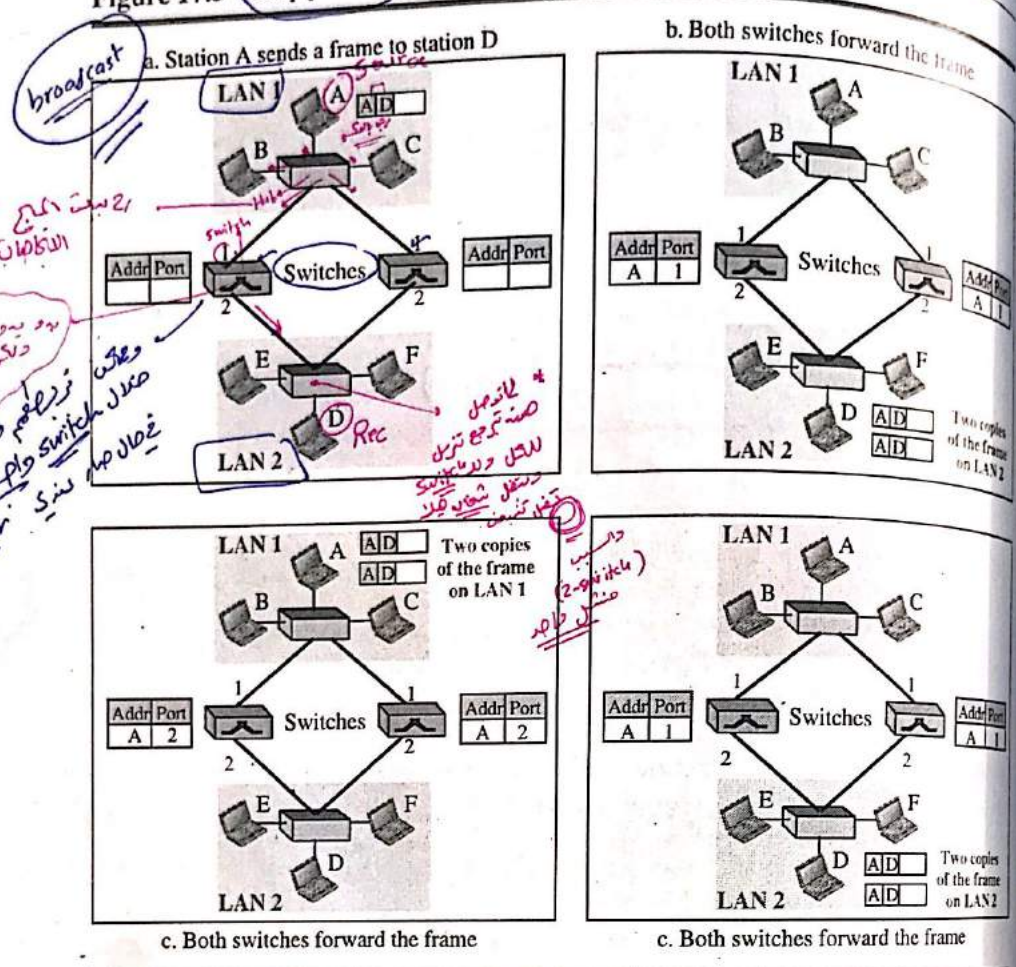
دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

دليل الينج اليه (A) كيج

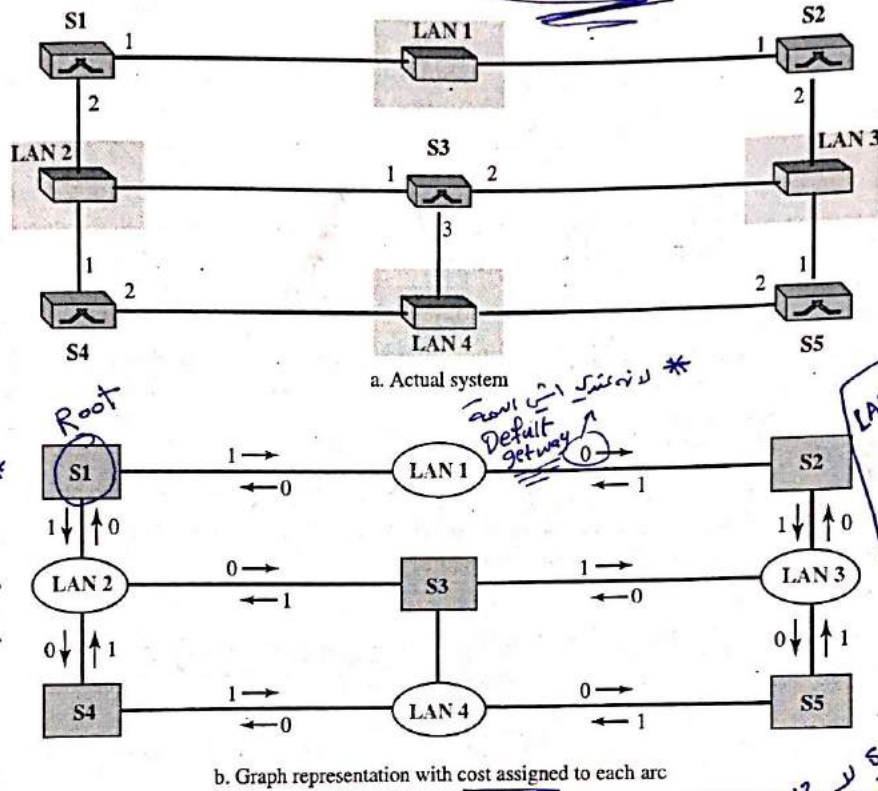


tree is a graph in which there is no loop. In a switched LAN, this means creating a topology in which each LAN can be reached from any other LAN through one path only (no loop). We cannot change the physical topology of the system because of physical connections between cables and switches, but we can create a logical topology that overlays the physical one. Figure 17.6 shows a system with four LANs and five switches. We have shown the physical system and its representation in graph theory. Although some textbooks represent the LANs as nodes and the switches as the connecting arcs, we have shown both LANs and switches as nodes. The connecting arcs show the connection of a LAN to a switch and vice versa. To find the spanning tree, we need to assign a cost (metric) to each arc. The interpretation of the cost is left up to the systems administrator. We have chosen the minimum hops. However, as we will see in Chapter 20, the hop count is normally 1 from a switch to the LAN and 0 in the reverse direction.

The process for finding the spanning tree involves three steps:

- 1. Every switch has a built-in ID (normally the serial number, which is unique). Each switch broadcasts this ID so that all switches know which one has the smallest ID. The switch with the smallest ID is selected as the *root* switch (root of the tree). We

Figure 17.6 A system of connected LANs and its graph representation



* هڪ سڀني جي
 مٿي هڪ سڀني
 اڳي ته مٿي
 اڳي ته مٿي
 اڳي ته مٿي

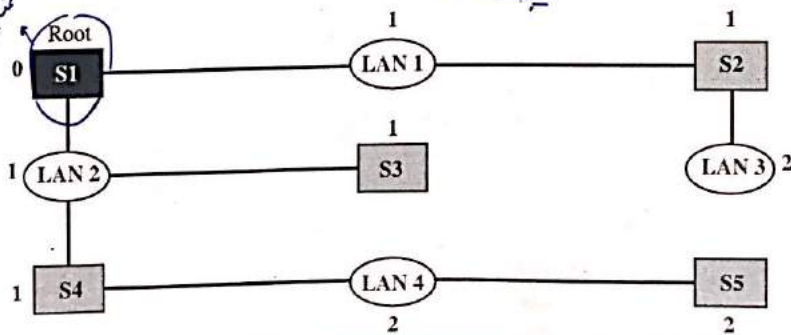
* هڪ سڀني
 2 =
 هڪ سڀني
 0 =

assume that switch S1 has the smallest ID. It is, therefore, selected as the root switch.

- The algorithm tries to find the shortest path (a path with the shortest cost) from the root switch to every other switch or LAN. The shortest path can be found by examining the total cost from the root switch to the destination. Figure 17.7 shows the shortest paths. We have used the Dijkstra algorithm described in Chapter 20.

Figure 17.7 Finding the shortest paths and the spanning tree in a system of switches

* هڪ سڀني
 مٿي
 مٿي
 مٿي

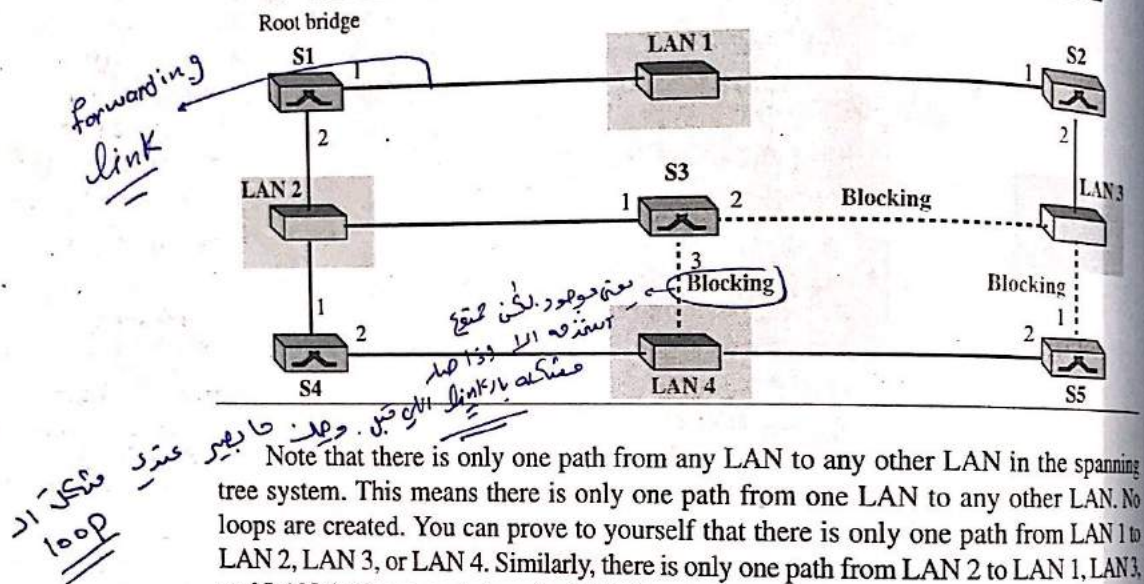


* Traffic
 Dijkstra

- The combination of the shortest paths creates the shortest tree, which is also shown in Figure 17.7.
- Based on the spanning tree, we mark the ports that are part of it, the **forwarding ports**, which forward a frame that the switch receives. We also mark those ports that are not part of the spanning tree, the **blocking ports**, which block the frames received by the switch. Figure 17.8 shows the logical systems of LANs with forwarding ports (solid lines) and blocking ports (broken lines).

Figure 17.8 Forwarding and blocking ports after using spanning tree algorithm

Ports 2 and 3 of bridge S3 are blocking ports (no frame is sent out of these ports). Port 1 of bridge S5 is also a blocking port (no frame is sent out of this port).



Note that there is only one path from any LAN to any other LAN in the spanning tree system. This means there is only one path from one LAN to any other LAN. No loops are created. You can prove to yourself that there is only one path from LAN 1 to LAN 2, LAN 3, or LAN 4. Similarly, there is only one path from LAN 2 to LAN 1, LAN 3, and LAN 4. The same is true for LAN 3 and LAN 4.

We have described the spanning tree algorithm as though it required manual entries. This is not true. Each switch is equipped with a software package that carries out this process dynamically.

Advantages of Switches

A link-layer switch has several advantages over a hub. We discuss only two of them here.

Collision Elimination

As we mentioned in Chapter 13, a link-layer switch eliminates the collision. This means increasing the average bandwidth available to a host in the network. In a switched LAN, there is no need for carrier sensing and collision detection; each host can transmit at any time.

Connecting Heterogeneous Devices

A link-layer switch can connect devices that use different protocols at the physical layer (data rates) and different transmission media. As long as the format of the frame

at the data-link layer does not change, a switch can receive a frame from a device that uses twisted-pair cable and sends data at 10 Mbps and deliver the frame to another device that uses fiber-optic cable and can receive data at 100 Mbps.

17.1.3 Routers :-

We will discuss routers in Part IV of the book when we discuss the network layer. In this section, we mention routers to compare them with a two-layer switch and a hub. A **router** is a three-layer device; it operates in the physical, data-link, and network layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the router checks the physical addresses (source and destination) contained in the packet. As a network-layer device, a router checks the network-layer addresses.

A router is a three-layer (physical, data-link, and network) device.

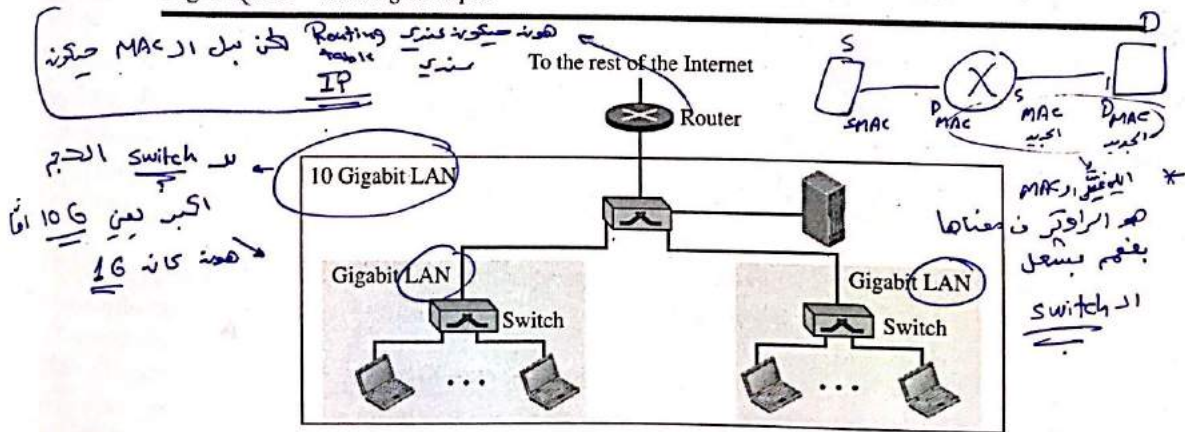
A router can connect networks. In other words, a router is an internetworking device; it connects independent networks to form an internetwork. According to this definition, two networks connected by a router become an internetwork or an internet.

There are three major differences between a router and a repeater or a switch.

- * 1. A router has physical and logical (IP) address for each of its interfaces.
- 2. A router acts only on those packets in which the link-layer destination address matches the address of the interface at which the packet arrives.
- 3. A router changes the link-layer address of the packet (both source and destination) when it forwards the packet.

Let us give an example. In Figure 17.9, assume an organization has two separate buildings with a Gigabit Ethernet LAN installed in each building. The organization uses switches in each LAN. The two LANs can be connected to form a larger LAN using 10 Gigabit Ethernet technology that speeds up the connection to the Ethernet and the connection to the organization server. A router then can connect the whole system to the Internet.

Figure 17.9 Routing example



A router, as we discuss in Chapter 18, will change the MAC addresses it receives because the MAC addresses have only local jurisdictions.

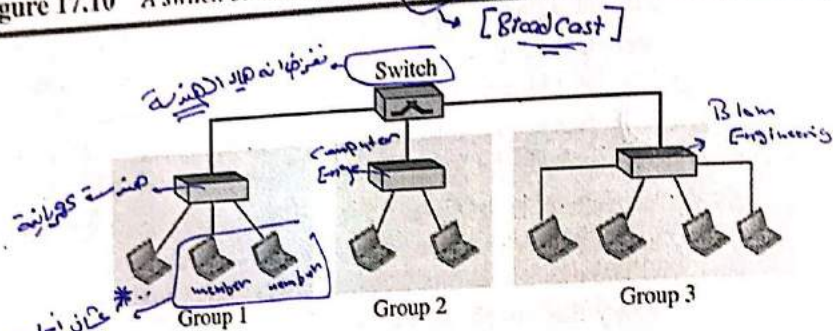
A router changes the link-layer addresses in a packet.

17.2 VIRTUAL LANS

A station is considered part of a LAN if it physically belongs to that LAN. The criterion of membership is geographic. What happens if we need a virtual connection between two stations belonging to two different physical LANs? We can roughly define a **virtual local area network (VLAN)** as a local area network configured by software, not by physical wiring.

Let us use an example to elaborate on this definition. Figure 17.10 shows a switched LAN in an engineering firm in which nine stations are grouped into three LANs that are connected by a switch.

Figure 17.10 A switch connecting three LANs



The first three engineers work together as the first group, the next two engineers work together as the second group, and the last four engineers work together as the third group. The LAN is configured to allow this arrangement.

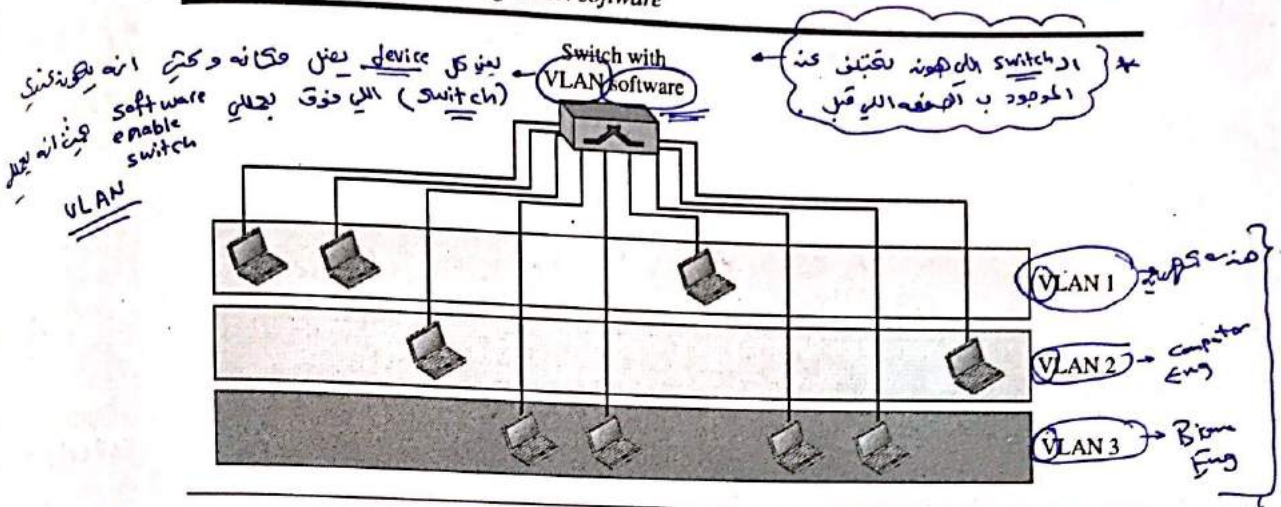
But what would happen if the administrators needed to move two engineers from the first group to the third group, to speed up the project being done by the third group? The LAN configuration would need to be changed. The network technician must rewire. The problem is repeated if, in another week, the two engineers move back to their previous group. In a switched LAN, changes in the work group mean physical changes in the network configuration.

Figure 17.11 shows the same switched LAN divided into VLANs. The whole idea of VLAN technology is to divide a LAN into logical, instead of physical, segments. A LAN can be divided into several logical LANs, called VLANs. Each VLAN is a work group in the organization. If a person moves from one group to another, there is no need to change the physical configuration. The group membership in VLANs is defined by software, not hardware. Any station can be logically moved to another VLAN. All members belonging to a VLAN can receive broadcast messages sent to that particular VLAN. This means that if a station moves from VLAN 1 to

Computer Engineering
 ربطهم ببعض
 في كل ما احتاج مع العمل
 فانه اذ لم يتغير
 Physical link
 فكل هذا السهل اول شي
 cost +
 يمكن ان يتغير
 ما تتغير تستقبل الابعث
 الجديدة.

VLAN فكل انه عمل

Figure 17.11 A switch using VLAN software

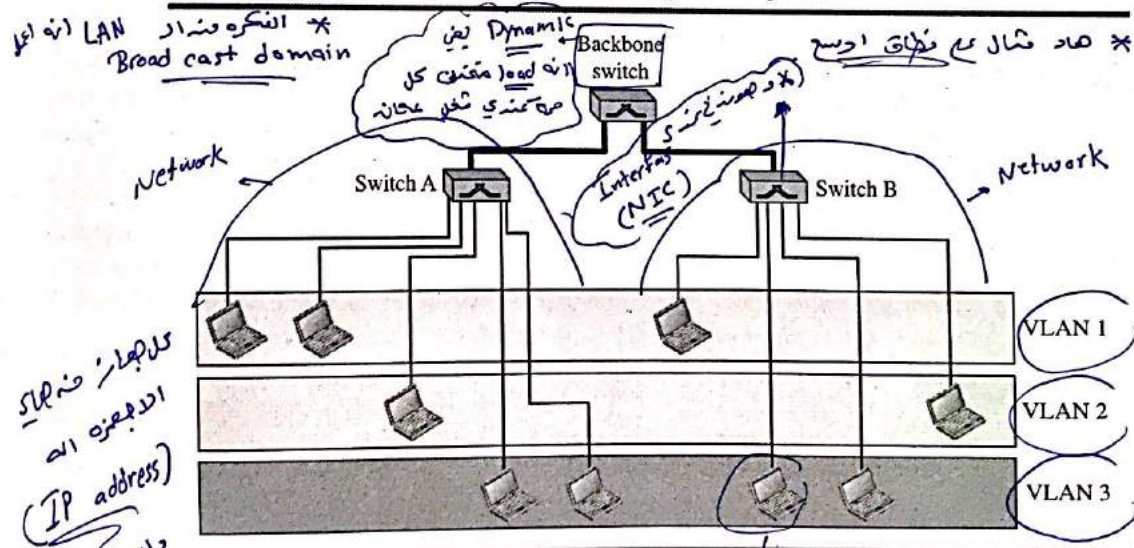


VLAN 2, it receives broadcast messages sent to VLAN 2, but no longer receives broadcast messages sent to VLAN 1.

It is obvious that the problem in our previous example can easily be solved by using VLANs. Moving engineers from one group to another through software is easier than changing the configuration of the physical network.

VLAN technology even allows the grouping of stations connected to different switches in a VLAN. Figure 17.12 shows a backbone local area network with two switches and three VLANs. Stations from switches A and B belong to each VLAN.

Figure 17.12 Two switches in a backbone using VLAN software



This is a good configuration for a company with two separate buildings. Each building can have its own switched LAN connected by a backbone. People in the first

من شك هاد اجهاز مربوطه
 Interface A لو بيد اوده ل (A) ف ب اللي بجلاي
 Interface B
 هاد مثال مع نطاق اوسع
 معشان ييسر هاد الشئ لازم
 يتبادر او [Switching table]
 ويحتاج لانه غير متغير زيانه بالمعموري بحتاج
 large Memory وهدا يتطلب (cost)

building and people in the second building can be in the same work group even though they are connected to different physical LANs.

From these three examples, we can see that a VLAN defines broadcast domains. VLANs group stations belonging to one or more physical LANs into broadcast domains. The stations in a VLAN communicate with one another as though they belonged to a physical segment.

17.2.1 Membership

What characteristic can be used to group stations in a VLAN? Vendors use different characteristics such as interface numbers, port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of two or more of these.

1) Interface Numbers

Some VLAN vendors use switch interface numbers as a membership characteristic. For example, the administrator can define that stations connecting to ports 1, 2, 3, and 7 belong to VLAN 1, stations connecting to ports 4, 10, and 12 belong to VLAN 2, and so on.

2) MAC Addresses

Some VLAN vendors use the 48-bit MAC address as a membership characteristic. For example, the administrator can stipulate that stations having MAC addresses E2:13:42:A1:23:34 and F2:A1:23:BC:D3:41 belong to VLAN 1.

3) IP Addresses

Some VLAN vendors use the 32-bit IP address (see Chapter 18) as a membership characteristic. For example, the administrator can stipulate that stations having IP addresses 181.34.23.67, 181.34.23.72, 181.34.23.98, and 181.34.23.112 belong to VLAN 1.

4) Multicast IP Addresses

Some VLAN vendors use the multicast IP address (see Chapter 21) as a membership characteristic. Multicasting at the IP layer is now translated to multicasting at the data-link layer.

5) Combination

Recently, the software available from some vendors allows all these characteristics to be combined. The administrator can choose one or more characteristics when installing the software. In addition, the software can be reconfigured to change the settings.

17.2.2 Configuration

How are the stations grouped into different VLANs? Stations are configured in one of three ways: manually, semiautomatically, and automatically.

1) Manual Configuration

In a manual configuration, the network administrator uses the VLAN software to manually assign the stations into different VLANs at setup. Later migration from one VLAN to another is also done manually. Note that this is not a physical configuration.

تسم منضلة
 4- انا بكون عندي شانه
 Interface
 عمل ال configuration من خلال
 ال keyboard انه بيضل مثل ما

it is a logical configuration. The term *manually* here means that the administrator types the port numbers, the IP addresses, or other characteristics, using the VLAN software.

Automatic Configuration

In an automatic configuration, the stations are automatically connected or disconnected from a VLAN using criteria defined by the administrator. For example, the administrator can define the project number as the criterion for being a member of a group. When a user changes projects, he or she automatically migrates to a new VLAN.

Semiautomatic Configuration

A semiautomatic configuration is somewhere between a manual configuration and an automatic configuration. Usually, the initializing is done manually, with migrations done automatically.

17.2.3 Communication between Switches

In a multi-switched backbone, each switch must know not only which station belongs to which VLAN, but also the membership of stations connected to other switches. For example, in Figure 17.12, switch A must know the membership status of stations connected to switch B, and switch B must know the same about switch A. Three methods have been devised for this purpose: table maintenance, frame tagging, and time-division multiplexing.

Table Maintenance

In this method, when a station sends a broadcast frame to its group members, the switch creates an entry in a table and records station membership. The switches send their tables to one another periodically for updating.

Frame Tagging

In this method, when a frame is traveling between switches, an extra header is added to the MAC frame to define the destination VLAN. The frame tag is used by the receiving switches to determine the VLANs to be receiving the broadcast message.

Time-Division Multiplexing (TDM)

In this method, the connection (trunk) between switches is divided into time-shared channels (see TDM in Chapter 6). For example, if the total number of VLANs in a backbone is five, each trunk is divided into five channels. The traffic destined for VLAN 1 travels in channel 1, the traffic destined for VLAN 2 travels in channel 2, and so on. The receiving switch determines the destination VLAN by checking the channel from which the frame arrived.

IEEE Standard

In 1996, the IEEE 802.1 subcommittee passed a standard called 802.1Q that defines the format for frame tagging. The standard also defines the format to be used in multi-switched backbones and enables the use of multivendor equipment in VLANs. IEEE 802.1Q has opened the way for further standardization in other issues related to VLANs. Most vendors have already accepted the standard.

انه اذا انا انا انا
 هذا هو القوي القوي
 في VLAN 2
 في بعض الحالات التي
 يكون فيها عدد من
 3

flexibility ^{جاء} 17.2.4 Advantages

There are several advantages to using VLANs.

1) Cost and Time Reduction

VLANs can reduce the migration cost of stations going from one group to another. Physical reconfiguration takes time and is costly. Instead of physically moving one station to another segment or even to another switch, it is much easier and quicker to move it by using software.

2) Creating Virtual Work Groups

VLANs can be used to create virtual work groups. For example, in a campus environment, professors working on the same project can send broadcast messages to one another without the necessity of belonging to the same department. This can reduce traffic if the multicasting capability of IP was previously used.

3) Security

VLANs provide an extra measure of security. People belonging to the same group can send broadcast messages with the guaranteed assurance that users in other groups will not receive these messages.

17.3 END-CHAPTER MATERIALS

17.3.1 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

Several books discuss link-layer issues. Among them we recommend [Ham 80], [Zar 02], [Ror 96], [Tan 03], [GW 04], [For 03], [KMK 04], [Sta 04], [Kes 02], [PD 03], [Kei 02], [Spu 00], [KCK 98], [Sau 98], [Izz 00], [Per 00], and [WV 00].

17.3.2 Key Terms

blocking port	repeater
connecting device	router
filtering	spanning tree
forwarding port	switch
hub	transparent switch
link-layer switch	virtual local area network (VLAN)

17.3.3 Summary

A repeater is a connecting device that operates in the physical layer of the Internet model. A repeater regenerates a signal, connects segments of a LAN, and has no filtering capability. A link-layer switch is a connecting device that operates in the physical and data-link layers of the Internet model. A transparent switch can forward and filter

Introduction to Network Layer :-

The network layer in the TCP/IP protocol suite is responsible for the host-to-host delivery of datagrams. It provides services to the transport layer and receives services from the data-link layer. In this chapter, we introduce the general concepts and issues in the network layer. This chapter also discusses the addressing mechanism used in the network layer, as briefly mentioned in Chapter 2. This chapter prepares the way for discussion of other network-layer issues, which follows in the next four chapters.

The chapter is divided into five sections.

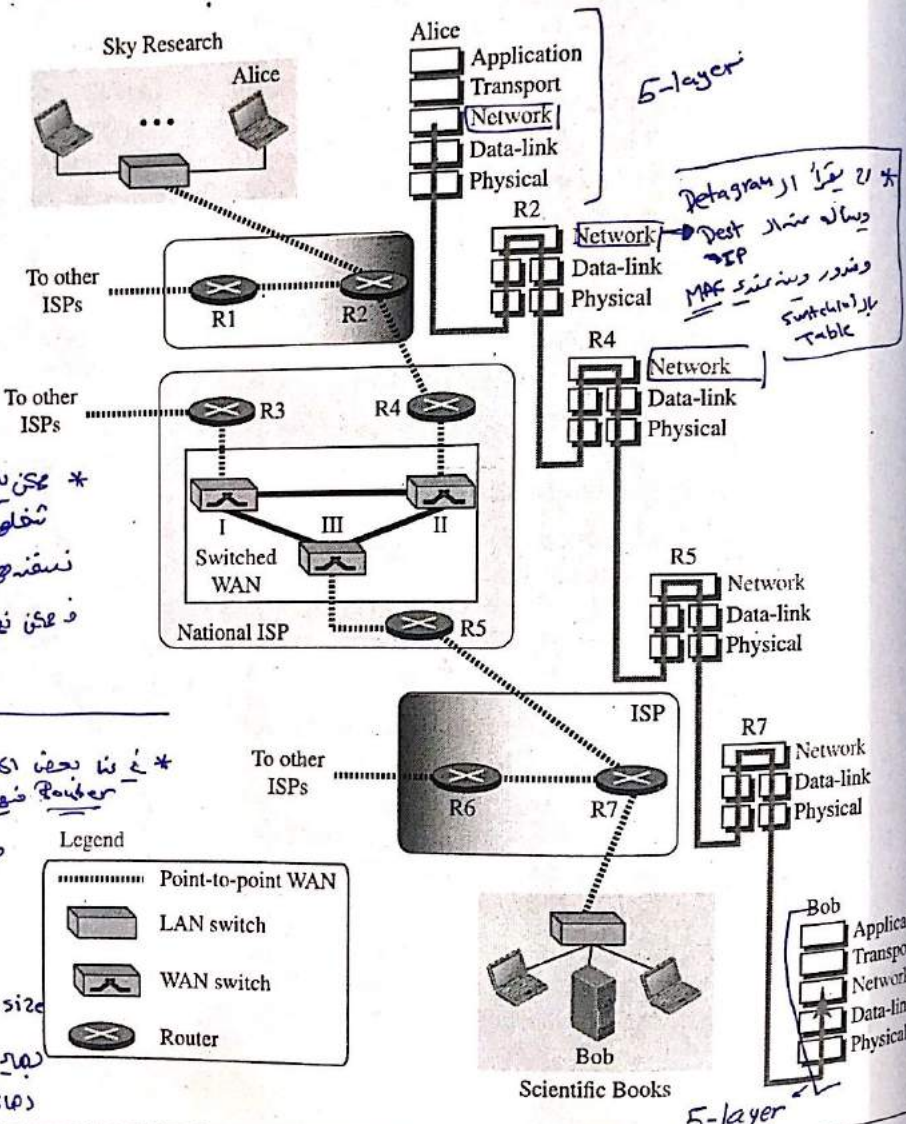
- The first section introduces the network layer by defining the services provided by this layer. It first discusses packetizing. It then describes forwarding and routing and compares the two. The section then briefly explains the other services such as flow, error, and congestion control.
- The second section discusses packet switching which occurs at the network layer. The datagram approach and the virtual-circuit approach of packet switching are described in some detail in this section.
- The third section discusses network-layer performance. It describes different delays that occur in network-layer communication. It also mentions the issue of packet loss. Finally, it discusses the issue of congestion control at the network layer.
- The fourth section discusses ^{*} IPv4 addressing, probably the most important issue in the network layer. It first describes the address space. It then briefly discusses classful addressing, which belongs to the past but is useful in understanding classless addressing. The section then moves to classless addressing and explains several issues related to this topic. It then discusses DHCP, which can be used to dynamically assign addresses in an organization. Finally, the section discusses NAT, which can be used to relieve the shortage of addresses to some extent.
- The fifth section discusses forwarding of network-layer packets. It first shows how forwarding can be done based on the destination address in a packet. It then discusses how forwarding can be done using a label.

End to End - يعني الاذرع كما يعمل اي
تتغير على ال Data gram ايضا

18.1 NETWORK-LAYER SERVICES :-

Before discussing the network layer in the Internet today, let's briefly discuss the network-layer services that, in general, are expected from a network-layer protocol. Figure 18.1 shows the communication between Alice and Bob at the network layer. This is the same scenario we used in Chapters 3 and 9 to show the communication at the physical and the data-link layers, respectively.

Figure 18.1 Communication at the network layer



* عنى يكونه نودى Router بتطلبه
شغلها اكثر من 5 layer زينه لاني
ننقلها لـ configuration
في عنى نقل run نـ APP محينه

* في بنا بعض الاكالات المحدوده اللي لسانه انه الـ
Router ضحا تتعمل بالمسبح. في الـ Data gram
بيعمل في بعض اتجاهايل المقلقه
بالـ Datagram منه هلا الاكالات
انه يكونه حجم الـ Data gram اكبر منه حجم
Maximum size اللي بتقبله الـ الـ الـ الـ
لهذا اكاه الـ الـ الـ الـ الـ الـ الـ الـ
الـ الـ الـ الـ الـ الـ الـ الـ الـ الـ الـ الـ الـ

وهذا الشئ اسمه
Fragmentation
انه نركب بعض الـ الـ الـ الـ
لقدّه استقام
* فالـ الـ الـ الـ الـ الـ الـ الـ الـ
في هانه اجزاء مع الـ الـ الـ الـ الـ الـ الـ الـ

The figure shows that the Internet is made of many networks (or links) connected through the connecting devices. In other words, the Internet is an internetwork, a

combination of LANs and WANs. To better understand the role of the network layer (or the internetwork layer), we need to think about the connecting devices (routers or switches) that connect the LANs and WANs.

As the figure shows, the network layer is involved at the source host, destination host, and all routers in the path (R2, R4, R5, and R7). At the source host (Alice), the network layer accepts a packet from a transport layer, encapsulates the packet in a datagram, and delivers the packet to the data-link layer. At the destination host (Bob), the datagram is decapsulated, and the packet is extracted and delivered to the corresponding transport layer. Although the source and destination hosts are involved in all five layers of the TCP/IP suite, the routers use three layers if they are routing packets only; however, they may need the transport and application layers for control purposes. A router in the path is normally shown with two data-link layers and two physical layers, because it receives a packet from one network and delivers it to another network.

Service ^{اداء}
18.1.1 Packetizing

✓ Layer 2 → fram
 ✓ Layer 3 → Data gram

من Source
 Encapsulation
 الى Dest
 Decapsulation

The first duty of the network layer is definitely **packetizing**: encapsulating the payload (data received from upper layer) in a network-layer packet at the source and decapsulating the payload from the network-layer packet at the destination. In other words, one duty of the network layer is to carry a payload from the source to the destination without changing it or using it. The network layer is doing the service of a carrier such as the postal office, which is responsible for delivery of packages from a sender to a receiver without changing or using the contents.

payload | header

The source host receives the payload from an upper-layer protocol, adds a header that contains the source and destination addresses and some other information that is required by the network-layer protocol (as discussed later) and delivers the packet to the data-link layer. The source is not allowed to change the content of the payload unless it is too large for delivery and needs to be fragmented.

تحت عنوان
 Payload

The destination host receives the network-layer packet from its data-link layer, decapsulates the packet, and delivers the payload to the corresponding upper-layer protocol. If the packet is fragmented at the source or at routers along the path, the network layer is responsible for waiting until all fragments arrive, reassembling them, and delivering them to the upper-layer protocol.

م
 صمغ
 تغير

The routers in the path are not allowed to decapsulate the packets they received unless the packets need to be fragmented. The routers are not allowed to change source and destination addresses either. They just inspect the addresses for the purpose of forwarding the packet to the next network on the path. However, if a packet is fragmented, the header needs to be copied to all fragments and some changes are needed, as we discuss in detail later.

switching ^{تحويل}
18.1.2 Routing and Forwarding

Other duties of the network layer, which are as important as the first, are routing and forwarding, which are directly related to each other.

عنوان
 اذ
 بروتوكول

* يعني نستقبل منه مكان ونرسله لكان
 (connection) (يعني اعمل)

Routing

The network layer is responsible for routing the packet from its source to the destination. A physical network is a combination of networks (LANs and WANs) and routers.

انه يستعمل
 ال Switching
 عنوان على جدول
 Process (جاء ال Packet
 table

* جدول
 ال Table
 ال Protocol

↓
 جدول
 ال node
 بعضهم
 ال بعض

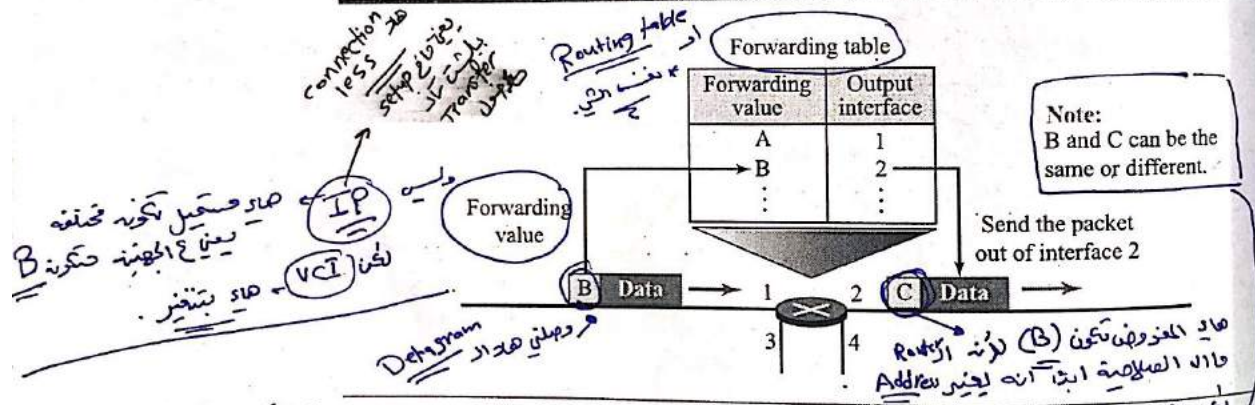
Routing
 Routing table
 Network X
 Network Y
 Routing table
 من خلال ال Router
 من خلال عرف انه (Y) من (X) في طرفها من port 4

Routing Protocol - كيف عرف هذه المعلومات :- عند تعريف Routing table
 that connect them. This means that there is more than one route from the source to the destination. The network layer is responsible for finding the best one among these possible routes. The network layer needs to have some specific strategies for defining the best route. In the Internet today, this is done by running some routing protocols to help the routers coordinate their knowledge about the neighborhood and to come up with consistent tables to be used when a packet arrives. The routing protocols, which we discuss in Chapters 20 and 21, should be run before any communication occurs.

يعني كيف استخدم هذه ال (Table)
Forwarding

Forwarding is applying strategies and running some routing protocols to create the decision-making tables for each router. Forwarding can be defined as the action applied by each router when a packet arrives at one of its interfaces. The decision-making table a router normally uses for applying this action is sometimes called the forwarding table and sometimes the routing table. When a router receives a packet from one of its attached networks, it needs to forward the packet to another attached network (in unicast routing) or to some attached networks (in multicast routing). To make this decision, the router uses a piece of information in the packet header, which can be the destination address or a label, to find the corresponding output interface number in the forwarding table. Figure 18.2 shows the idea of the forwarding process in a router.

Figure 18.2 Forwarding process



IP
 VCI
 VC S
 Network layer
 connection less
 error control
 Header

18.1.3 Other Services

Let us briefly discuss other services expected from the network layer.

Error Control

In Chapter 10, we discussed error detection and correction. Although error control also can be implemented in the network layer, the designers of the network layer in the Internet ignored this issue for the data being carried by the network layer. One reason for this decision is the fact that the packet in the network layer may be fragmented at each router, which makes error checking at this layer inefficient.

3

The designers of the network layer, however, have added a checksum field to the datagram to control any corruption in the header, but not in the whole datagram. This checksum may prevent any changes or corruptions in the header of the datagram.

We need to mention that although the network layer in the Internet does not directly provide error control, the Internet uses an auxiliary protocol ICMP, that provides some kind of error control if the datagram is discarded or has some unknown information in the header. We discuss ICMP in Chapter 19.

Flow Control :-

Flow control regulates the amount of data a source can send without overwhelming the receiver. If the upper layer at the source computer produces data faster than the upper layer at the destination computer can consume it, the receiver will be overwhelmed with data. To control the flow of data, the receiver needs to send some feedback to the sender to inform the latter that it is overwhelmed with data.

The network layer in the Internet, however, does not directly provide any flow control. The datagrams are sent by the sender when they are ready, without any attention to the readiness of the receiver.

A few reasons for the lack of flow control in the design of the network layer can be mentioned. (First) since there is no error control in this layer, the job of the network layer at the receiver is so simple that it may rarely be overwhelmed. (Second) the upper layers that use the service of the network layer can implement buffers to receive data from the network layer as they are ready and do not have to consume the data as fast as it is received. (Third) flow control is provided for most of the upper-layer protocols that use the services of the network layer, so another level of flow control makes the network layer more complicated and the whole system less efficient.

يعني انه الشبكات لا توفر قابلية للتحكم في سرعة البيانات

Network → Connection less
 Setup / Trade down
 يعني كذا packets بينت
 Path
 فنتج

Congestion Control :-

Another issue in a network-layer protocol is congestion control. Congestion in the network layer is a situation in which too many datagrams are present in an area of the Internet. Congestion may occur if the number of datagrams sent by source computers is beyond the capacity of the network or routers. In this situation, some routers may drop some of the datagrams. However, as more datagrams are dropped, the situation may become worse because, due to the error control mechanism at the upper layers, the sender may send duplicates of the lost packets. If the congestion continues, sometimes a situation may reach a point where the system collapses and no datagrams are delivered. We discuss congestion control at the network layer later in the chapter although it is not implemented in the Internet.

تأخر في سرعة
 flow + Error control
 فنتج ما سرعة
 Congestion

Quality of Service :-

As the Internet has allowed new applications such as multimedia communication (in particular real-time communication of audio and video), the quality of service (QoS) of the communication has become more and more important. The Internet has thrived by providing better quality of service to support these applications. However, to keep the network layer untouched, these provisions are mostly implemented in the upper layer. We discuss this issue in Chapter 30 after we have discussed multimedia.

UDP → service (لا) Faster Delivery

انه لا اعني
 فنتج كذا كذا
 مع السلاية او
 انما الى كذا

في حال استخدمت TCP
 Kind of Delay
 عني

بصير اى اى انما كذا ل
 Throughput ↑
 Delay ↓
 Packet loss ↓

*
رابطه
IP sec

Security
فصل 32
Internet
البريد الإلكتروني



Another issue related to communication at the network layer is security. Security was not a concern when the Internet was originally designed because it was used by a small number of users at universities for research activities; other people had no access to the Internet. The network layer was designed with no security provisions. Today, however, security is a big concern. To provide security for a connectionless network layer, we need to have another virtual level that changes the connectionless service to a connection-oriented service. This virtual layer, called IPSec, is discussed in Chapter 32.

18.2 PACKET SWITCHING

From the discussion of routing and forwarding in the previous section, we infer that a kind of switching occurs at the network layer. A router, in fact, is a switch that creates a connection between an input port and an output port (or a set of output ports), just as an electrical switch connects the input to the output to let electricity flow.

فصل 18
Packet

Although in data communication switching techniques are divided into two broad categories, circuit switching and packet switching, only packet switching is used at the network layer because the unit of data at this layer is a packet. Circuit switching is mostly used at the physical layer; the electrical switch mentioned earlier is a kind of circuit switch. We discussed circuit switching in Chapter 8; we discuss packet switching in this chapter.

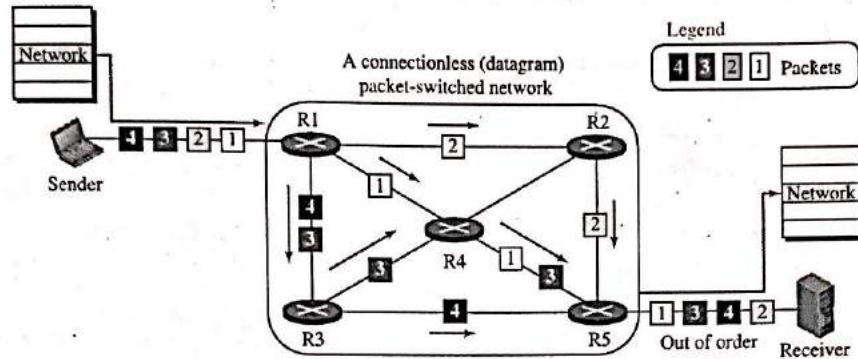
At the network layer, a message from the upper layer is divided into manageable packets and each packet is sent through the network. The source of the message sends the packets one by one; the destination of the message receives the packets one by one. The destination waits for all packets belonging to the same message to arrive before delivering the message to the upper layer. The connecting devices in a packet-switched network still need to decide how to route the packets to the final destination. Today, a packet-switched network can use two different approaches to route the packets: the datagram approach and the virtual circuit approach. We discuss both approaches in the next section.

18.2.1 Datagram Approach: Connectionless Service

When the Internet started, to make it simple, the network layer was designed to provide a connectionless service in which the network-layer protocol treats each packet independently, with each packet having no relationship to any other packet. The idea was that the network layer is only responsible for delivery of packets from the source to the destination. In this approach, the packets in a message may or may not travel the same path to their destination. Figure 18.3 shows the idea.

When the network layer provides a connectionless service, each packet traveling in the Internet is an independent entity; there is no relationship between packets belonging to the same message. The switches in this type of network are called routers. A packet belonging to a message may be followed by a packet belonging to the same message or to a different message. A packet may be followed by a packet coming from the same or from a different source.

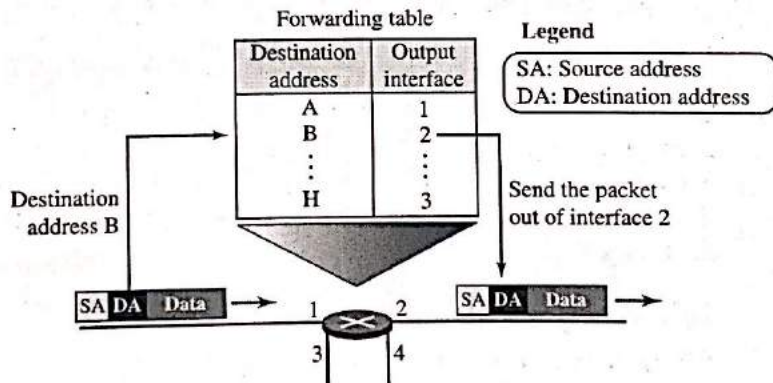
Figure 18.3 A connectionless packet-switched network



Handwritten note: Packet 45
 45 packets
 11/2

Each packet is routed based on the information contained in its header: source and destination addresses. The destination address defines where it should go; the source address defines where it comes from. The router in this case routes the packet based only on the destination address. The source address may be used to send an error message to the source if the packet is discarded. Figure 18.4 shows the forwarding process in a router in this case. We have used symbolic addresses such as A and B.

Figure 18.4 Forwarding process in a router when used in a connectionless network



In the datagram approach, the forwarding decision is based on the destination address of the packet.

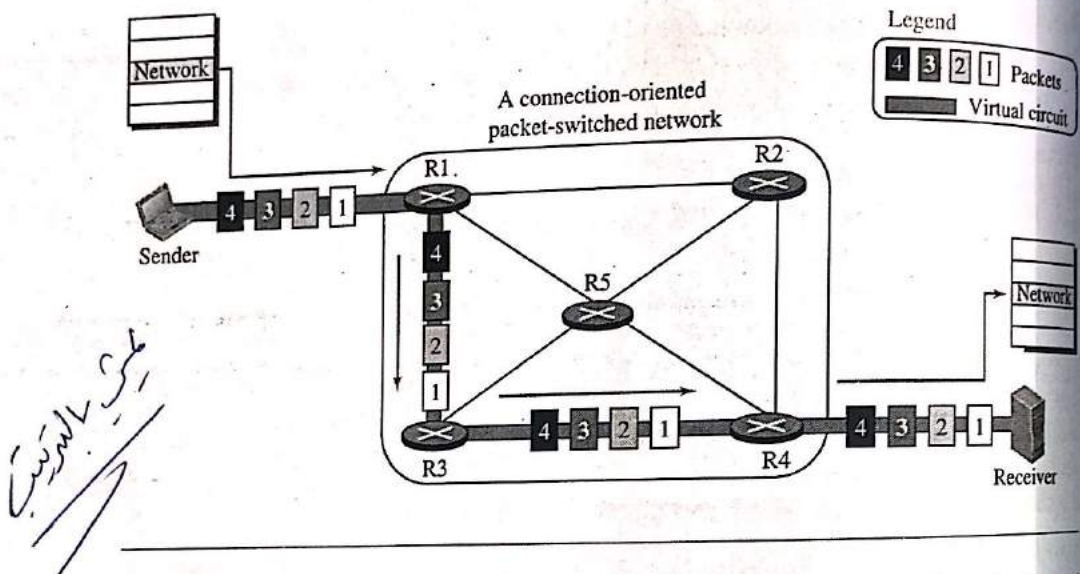
18.2.2 Virtual-Circuit Approach: Connection-Oriented Service

In a connection-oriented service (also called *virtual-circuit approach*), there is a relationship between all packets belonging to a message. Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams. After connection setup, the datagrams can all follow the same path. In this type of service, not

source و destination
 flow label
 3 flow label

only must the packet contain the source and destination addresses, it must also contain a flow-label, a virtual circuit identifier that defines the virtual path the packet should follow. Shortly, we will show how this flow label is determined, but for the moment, we assume that the packet carries this label. Although it looks as though the use of the label may make the source and destination addresses unnecessary during the data transfer phase, parts of the Internet at the network layer still keep these addresses. One reason is that part of the packet path may still be using the connectionless service. Another reason is that the protocol at the network layer is designed with these addresses, and it may take a while before they can be changed. Figure 18.5 shows the concept of connection-oriented service.

Figure 18.5 A virtual-circuit packet-switched network



مسار الترتيب

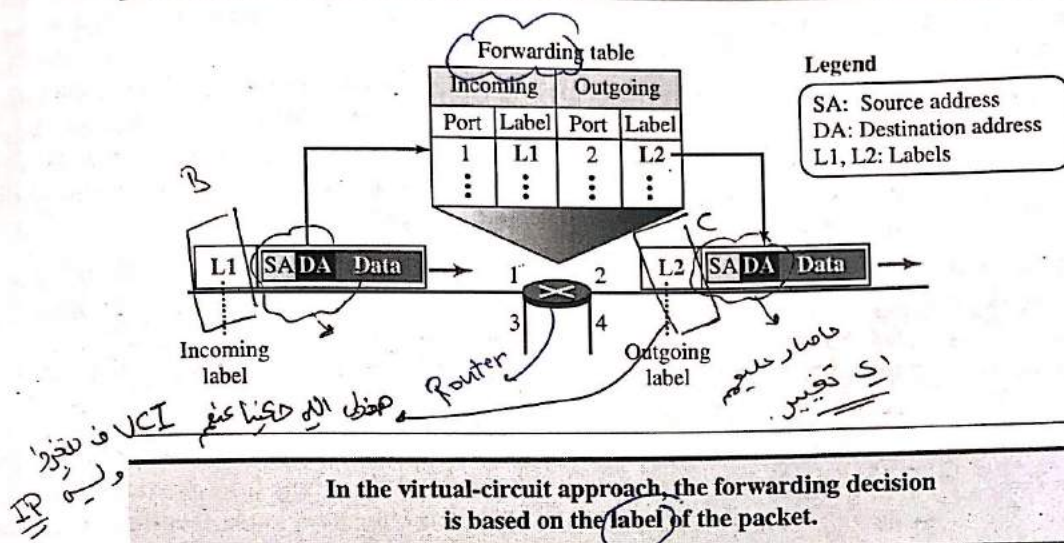
Each packet is forwarded based on the label in the packet. To follow the idea of connection-oriented design to be used in the Internet, we assume that the packet has a label when it reaches the router. Figure 18.6 shows the idea. In this case, the forwarding decision is based on the value of the label, or *virtual circuit identifier*, as it is sometimes called.

To create a connection-oriented service, a three-phase process is used: setup, data transfer, and teardown. In the setup phase, the source and destination addresses of the sender and receiver are used to make table entries for the connection-oriented service. In the teardown phase, the source and destination inform the router to delete the corresponding entries. Data transfer occurs between these two phases.

Setup Phase

In the setup phase, a router creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to destination B. Two auxiliary packets need to be exchanged between the sender and the receiver: the request packet and the acknowledgment packet.

Figure 18.6 Forwarding process in a router when used in a virtual-circuit network

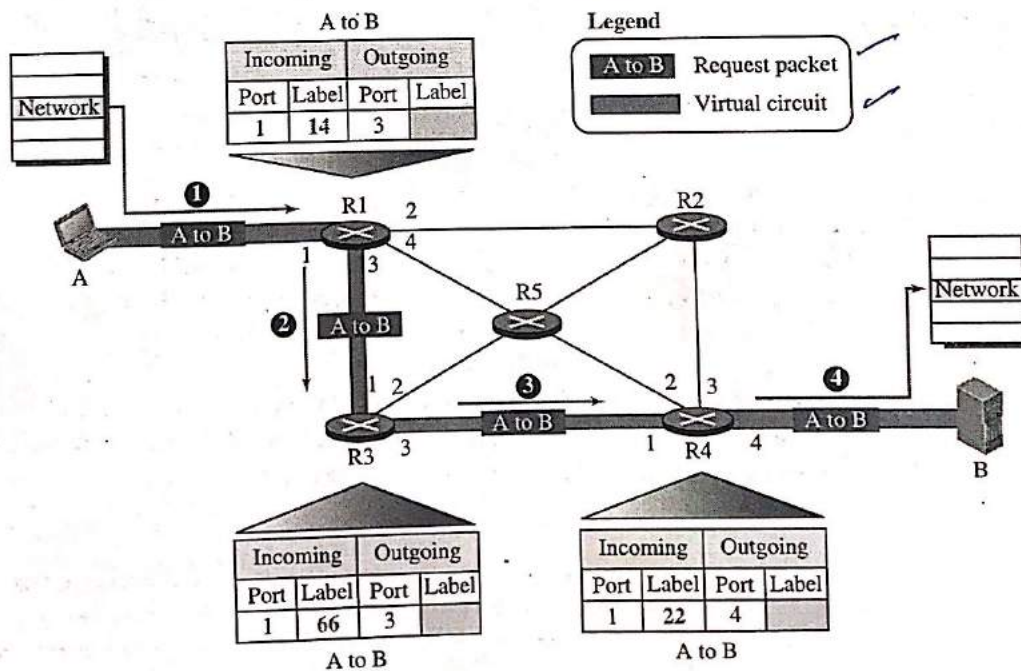


In the virtual-circuit approach, the forwarding decision is based on the label of the packet.

Request packet

A request packet is sent from the source to the destination. This auxiliary packet carries the source and destination addresses. Figure 18.7 shows the process.

Figure 18.7 Sending request packet in a virtual-circuit network

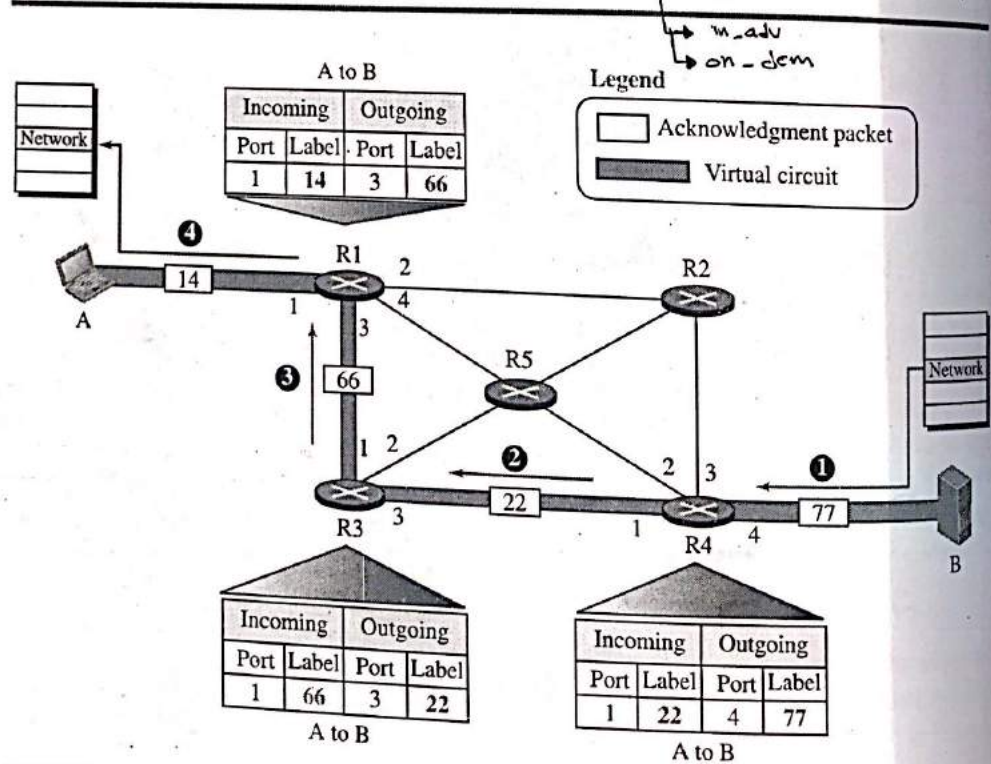


1. Source A sends a request packet to router R1.
2. Router R1 receives the request packet. It knows that a packet going from A to B goes out through port 3. How the router has obtained this information is a point covered later. For the moment, assume that it knows the output port. The router creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The router assigns the incoming port (1) and chooses an available incoming label (14) and the outgoing port (3). It does not yet know the outgoing label, which will be found during the acknowledgment step. The router then forwards the packet through port 3 to router R3.
3. Router R3 receives the setup request packet. The same events happen here as at router R1; three columns of the table are completed: in this case, incoming port (1), incoming label (66), and outgoing port (3).
4. Router R4 receives the setup request packet. Again, three columns are completed: incoming port (1), incoming label (22), and outgoing port (4).
5. Destination B receives the setup packet, and if it is ready to receive packets from A, it assigns a label to the incoming packets that come from A, in this case 77, as shown in Figure 18.8. This label lets the destination know that the packets come from A, and not from other sources.

Acknowledgment Packet

A special packet, called the acknowledgment packet, completes the entries in the switching tables. Figure 18.8 shows the process.

Figure 18.8 Sending acknowledgments in a virtual-circuit network

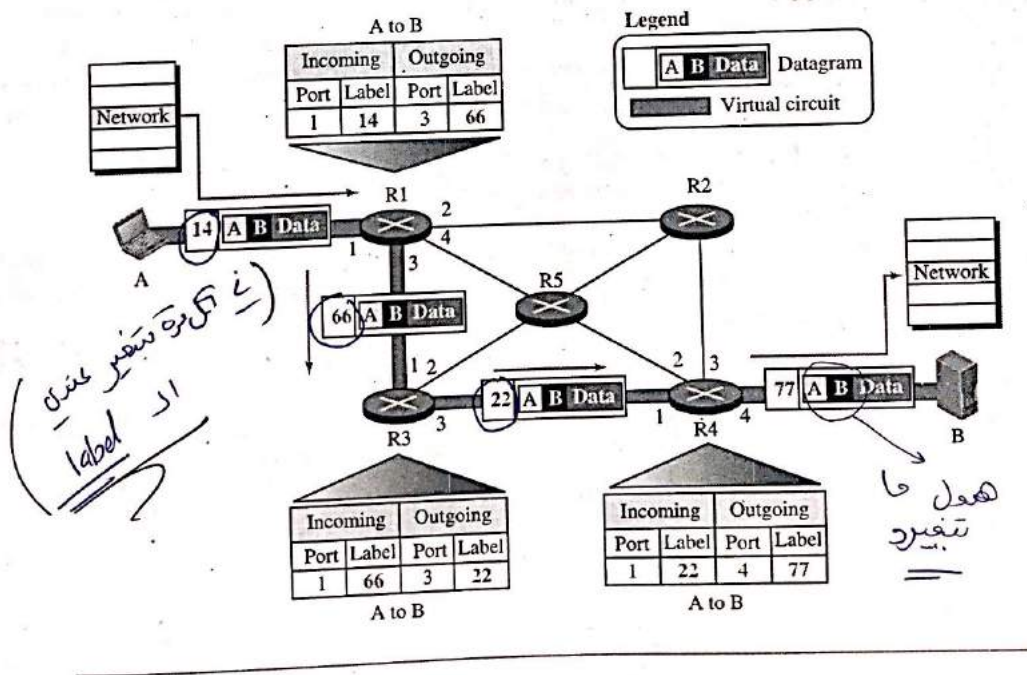


1. The destination sends an acknowledgment to router R4. The acknowledgment carries the global source and destination addresses so the router knows which entry in the table is to be completed. The packet also carries label 77, chosen by the destination as the incoming label for packets from A. Router R4 uses this label to complete the outgoing label column for this entry. Note that 77 is the incoming label for destination B, but the outgoing label for router R4.
2. Router R4 sends an acknowledgment to router R3 that contains its incoming label in the table, chosen in the setup phase. Router R3 uses this as the outgoing label in the table.
3. Router R3 sends an acknowledgment to router R1 that contains its incoming label in the table, chosen in the setup phase. Router R1 uses this as the outgoing label in the table.
4. Finally router R1 sends an acknowledgment to source A that contains its incoming label in the table, chosen in the setup phase.
5. The source uses this as the outgoing label for the data packets to be sent to destination B.

Data-Transfer Phase

The second phase is called the data-transfer phase. After all routers have created their forwarding table for a specific virtual circuit, then the network-layer packets belonging to one message can be sent one after another. In Figure 18.9, we show the flow of a single packet, but the process is the same for 1, 2, or 100 packets. The source computer uses the label 14, which it has received from router R1 in the setup phase

Figure 18.9 Flow of one packet in an established virtual circuit



phase. Router R1 forwards the packet to router R3, but changes the label to 66. Router R3 forwards the packet to router R4, but changes the label to 22. Finally, router R4 delivers the packet to its final destination with the label 77. All the packets in the message follow the same sequence of labels, and the packets arrive in order at the destination.

Teardown Phase

In the teardown phase, source A, after sending all packets to B, sends a special packet called a teardown packet. Destination B responds with a confirmation packet. All routers delete the corresponding entries from their tables.

18.3 NETWORK-LAYER PERFORMANCE

The upper-layer protocols that use the service of the network layer expect to receive an ideal service, but the network layer is not perfect. The performance of a network can be measured in terms of delay, throughput, and packet loss. Congestion control is an issue that can improve the performance.

18.3.1 Delay

عدد ال bit الذي يمر خلال نقطة معينة خلال فترة زمنية

All of us expect instantaneous response from a network, but a packet, from its source to its destination, encounters delays. The delays in a network can be divided into four types: transmission delay, propagation delay, processing delay, and queuing delay. Let us first discuss each of these delay types and then show how to calculate a packet delay from the source to the destination.

الوقت الذي تستغرقه الآلة Machine

Transmission Delay

A source host or a router cannot send a packet instantaneously. A sender needs to put the bits in a packet on the line one by one. If the first bit of the packet is put on the line at time t_1 and the last bit is put on the line at time t_2 , transmission delay of the packet is $(t_2 - t_1)$. Definitely, the transmission delay is longer for a longer packet and shorter if the sender can transmit faster. In other words, the transmission delay is

$$Delay_{tr} = (\text{Packet length}) / (\text{Transmission rate}).$$

For example, in a Fast Ethernet LAN (see Chapter 13) with the transmission rate of 100 million bits per second and a packet of 10,000 bits, it takes $(10,000)/(100,000,000)$ or 100 microseconds for all bits of the packet to be put on the line.

الوقت الذي تستغرقه سلكي لنقل Link

Propagation Delay (node to node)

Propagation delay is the time it takes for a bit to travel from point A to point B in the transmission media. The propagation delay for a packet-switched network depends on the propagation delay of each network (LAN or WAN). The propagation delay depends on the propagation speed of the media, which is 3×10^8 meters/second in a vacuum and normally much less in a wired medium; it also depends on the distance of the link. In other words, propagation delay is

$$Delay_{pr} = (\text{Distance}) / (\text{Propagation speed}).$$

For example, if the distance of a cable link in a point-to-point WAN is 2000 meters and the propagation speed of the bits in the cable is 2×10^8 meters/second, then the propagation delay is 10 microseconds.

الوقت اللازم حان
Processing

Processing Delay

الروتير الذي يحمله ومنه process
يعبر (D)

الوقت (D) يتكرر header قبل ما
تفتح الخلف + Decapsulation

The processing delay is the time required for a router or a destination host to receive a packet from its input port, remove the header, perform an error detection procedure, and deliver the packet to the output port (in the case of a router) or deliver the packet to the upper-layer protocol (in the case of the destination host). The processing delay may be different for each packet, but normally is calculated as an average.

$Delay_{pr} = \text{Time required to process a packet in a router or a destination host}$

الوقت في الروتير
او (D)
الوقت الذي ينتظره
الباكيت في الروتير
Busy

Queuing Delay

Queuing delay can normally happen in a router. As we discuss in the next section, a router has an input queue connected to each of its input ports to store packets waiting to be processed; the router also has an output queue connected to each of its output ports to store packets waiting to be transmitted. The queuing delay for a packet in a router is measured as the time a packet waits in the input queue and output queue of a router. We can compare the situation with a busy airport. Some planes may need to wait to get the landing band (input delay); some planes may need to wait to get the departure band (output delay).

$Delay_{qu} = \text{The time a packet waits in input and output queues in a router}$

Total Delay

Assuming equal delays for the sender, routers, and receiver, the total delay (source-to-destination delay) a packet encounters can be calculated if we know the number of routers, n , in the whole path.

* $Total\ delay = (n + 1) (Delay_{ir} + Delay_{pg} + Delay_{pr}) + (n) (Delay_{qu})$

Note that if we have n routers, we have $(n + 1)$ links. Therefore, we have $(n + 1)$ transmission delays related to n routers and the source, $(n + 1)$ propagation delays related to $(n + 1)$ links, $(n + 1)$ processing delays related to n routers and the destination, and only n queuing delays related to n routers.

صدا المتنازرة

18.3.2 Throughput

Throughput at any point in a network is defined as the number of bits passing through the point in a second, which is actually the transmission rate of data at that point. In a path from source to destination, a packet may pass through several links (networks), each with a different transmission rate. How, then, can we determine the throughput of the whole path? To see the situation, assume that we have three links, each with a different transmission rate, as shown in Figure 18.10.

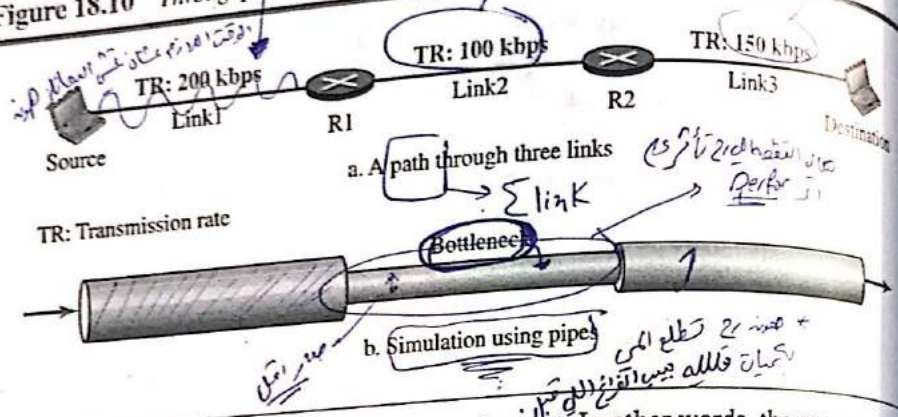
In this figure, the data can flow at the rate of 200 kbps in Link1. However, when the data arrives at router R1, it cannot pass at this rate. Data needs to be queued at the router and sent at 100 kbps. When data arrives at router R2, it could be sent at the rate

Transmission delay
 يقسمه (الراوتر) الى
 اقسام (Trans) و bit
 link @ bit

لانه ما يستقبل الا 100 ف 2
 في Queue في R1

100 / 100 = 1
 100 / 100 = 1
 100 / 100 = 1

Figure 18.10 Throughput in a path with three links in a series

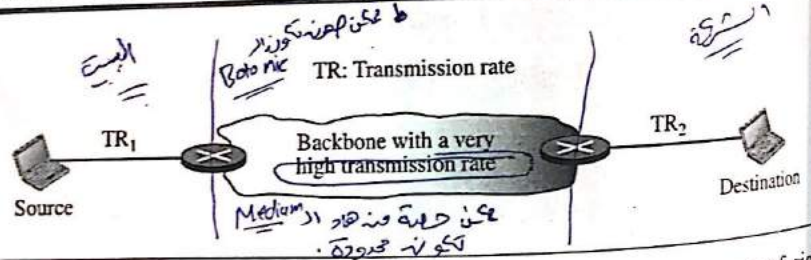


of 150 kbps, but there is not enough data to be sent. In other words, the average rate of the data flow in Link3 is also 100 kbps. We can conclude that the average data rate for this path is 100 kbps, the minimum of the three different data rates. The figure also shows that we can simulate the behavior of each link with pipes of different sizes; the average throughput is determined by the bottleneck, the pipe with the smallest diameter. In general, in a path with n links in series, we have

$$\text{Throughput} = \text{minimum} \{TR_1, TR_2, \dots, TR_n\}.$$

Although the situation in Figure 18.10 shows how to calculate the throughput when the data is passed through several links, the actual situation in the Internet is that the data normally passes through two access networks and the Internet backbone, as shown in Figure 18.11.

Figure 18.11 A path through the Internet backbone



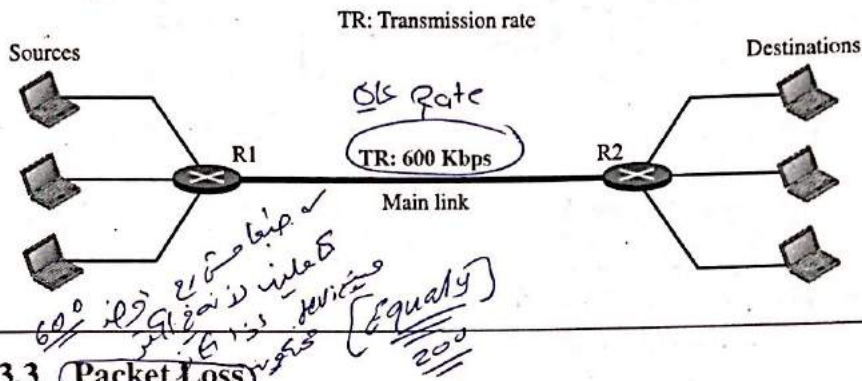
Minimum transmission rate

The Internet backbone has a very high transmission rate, in the range of gigabits per second. This means that the throughput is normally defined as the minimum transmission rate of the two access links that connect the source and destination to the backbone. Figure 18.11 shows this situation, in which the throughput is the minimum of TR_1 and TR_2 . For example, if a server connects to the Internet via a Fast Ethernet LAN with the data rate of 100 Mbps, but a user who wants to download a file connects to the Internet via a dial-up telephone line with the data rate of 40 kbps, the throughput is 40 kbps. The bottleneck is definitely the dial-up line.

We need to mention another situation in which we think about the throughput. The link between two routers is not always dedicated to one flow. A router may collect the

flow from several sources or distribute the flow between several sources. In this case the transmission rate of the link between the two routers is actually shared between the flows and this should be considered when we calculate the throughput. For example, in Figure 18.12 the transmission rate of the main link in the calculation of the throughput is only 200 kbps because the link is shared between three paths.

Figure 18.12 Effect of throughput in shared links



18.3.3 Packet Loss

Another issue that severely affects the performance of communication is the number of packets lost during transmission. When a router receives a packet while processing another packet, the received packet needs to be stored in the input buffer waiting for its turn. A router, however, has an input buffer with a limited size. A time may come when the buffer is full and the next packet needs to be dropped. The effect of packet loss on the Internet network layer is that the packet needs to be resent, which in turn may create overflow and cause more packet loss. A lot of theoretical studies have been done in queuing theory to prevent the overflow of queues and prevent packet loss.

18.3.4 Congestion Control

Congestion control is a mechanism for improving performance. In Chapter 23, we will discuss congestion at the transport layer. Although congestion at the network layer is not explicitly addressed in the Internet model, the study of congestion at this layer may help us to better understand the cause of congestion at the transport layer and find possible remedies to be used at the network layer. Congestion at the network layer is related to two issues, throughput and delay, which we discussed in the previous section. Figure 18.13 shows these two performance measures as functions of load.

When the load is much less than the capacity of the network, the delay is at a minimum. This minimum delay is composed of propagation delay and processing delay, both of which are negligible. However, when the load reaches the network capacity, the delay increases sharply because we now need to add the queuing delay to the total delay. Note that the delay becomes infinite when the load is greater than the capacity.

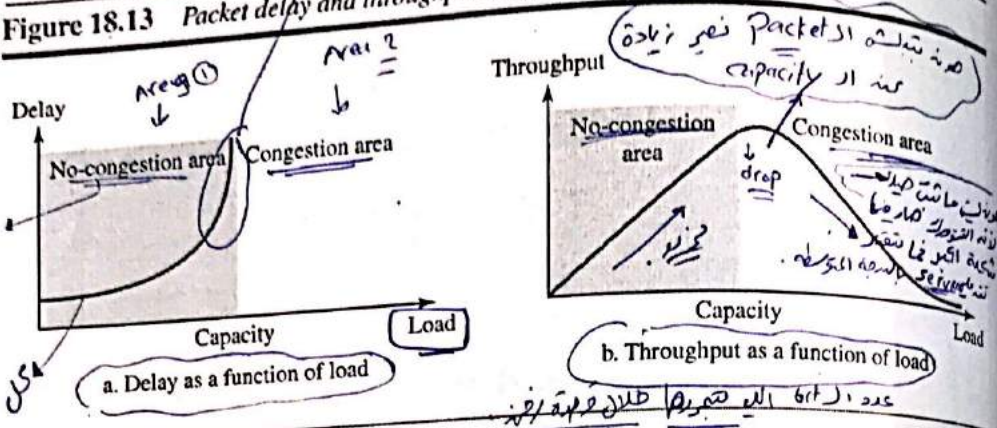
When the load is below the capacity of the network, the throughput increases proportionally with the load. We expect the throughput to remain constant after the load reaches the capacity, but instead the throughput declines sharply. The reason is the discarding of packets by the routers. When the load exceeds the capacity, the queues

packet

1
2
3

loss

Figure 18.13 Packet delay and throughput as functions of load



Area التي لها ما هو
Congestion

كل ما زاد ال
load يزيد ال delay
لكن ال delay (prop + trans)
لا يزداد بل ال delay
الذي هو ال delay
الذي هو ال delay
الذي هو ال delay

become full and the routers have to discard some packets. Discarding packets does not reduce the number of packets in the network because the sources retransmit the packets, using time-out mechanisms, when the packets do not reach the destinations.

Congestion Control

Congestion control refers to techniques and mechanisms that can either prevent congestion before it happens or remove congestion after it has happened. In general, we can divide congestion control mechanisms into two broad categories: **open-loop congestion control (prevention)** and **closed-loop congestion control (removal)**.

Open-Loop Congestion Control

In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination. We give a brief list of policies that can prevent congestion.

Retransmission Policy

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is **lost or corrupted**, the packet needs to be **retransmitted**. Retransmission in general may **increase congestion** in the network. However, a **good retransmission policy can prevent congestion**. The retransmission policy and the retransmission timers must be designed to **optimize efficiency** and at the same time **prevent congestion**.

Window Policy

The type of window at the sender may also affect congestion. The **Selective Repeat window** is better than the **Go-Back-N window** for congestion control. In the **Go-Back-N window**, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The **Selective Repeat window**, on the other hand, tries to send the specific packets that have been lost or corrupted.

Acknowledgment Policy

The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not **acknowledge every packet** it receives, it may **slow down the sender** and help prevent congestion. Several approaches are used in this case. A receiver may **send an acknowledgment** only if it has a packet to be sent or a **special timer expires**. A receiver may decide to acknowledge only **N packets** at a time.

علاوة على ال congestion
انه ارفع ال congestion
جديدة

بنت مجموعة من ال packet
فتمت وطلبت مجموعة ال packet
ال sender
ال receiver
ال sender
ال receiver
ال sender
ال receiver

ال timer
ال timer
ال timer

لنفس ال وقت ال message
5s
20s
ال receiver
ال sender
ال sender
ال receiver

We need to know that the acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.

الذي من مهمات الشبكة
تذكر في مودمنا اننا نرسل
معلومات اقل من
load
Discarding
الطبع

Discarding Policy A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

Admission Policy An admission policy, which is a quality-of-service mechanism (discussed in Chapter 30), can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Virtual circuit switch
set up
Busy

Closed-Loop Congestion Control Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols. We describe a few of them here.

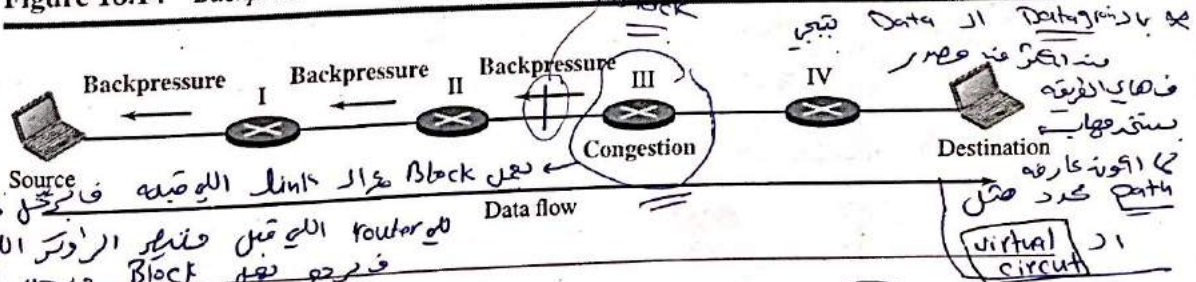
كثير من الازدحام
دائما يرد حلول الازدحام

Busy راجي طلب
Circuits
لا يمكن ان يكون
ممتلئ (we can't receive)
Delay

Backpressure The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream node or nodes, and so on. Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is coming. Figure 18.14 shows the idea of backpressure.

بدا اذ كان
ارسله للوجه
العامه
لنفسه
مسألة
ازدحام
node by node
اذا ما اوجد الازدحام

Figure 18.14 Backpressure method for alleviating congestion



كل ال bit
Congestion
router الذي قبل
فمنه router الذي قبل
Block
في وجه

Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested in its input buffer and informs node I to slow down. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I informs the source of data to slow down. This, in time, alleviates the congestion. Note that the pressure on node III is moved backward to the source to remove the congestion.

It is important to stress that this type of congestion control can only be implemented in virtual-circuit. The technique cannot be implemented in a datagram network, in which a node (router) does not have the slightest knowledge of the upstream router.

68*68
170

We need to know that the acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.

الذي من مهمات روتر
تذكري في روترنا اننا نحتاج
مضيقان اقل في
load سنبتن
Discarding في المطح

Discarding Policy A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

Admission Policy An admission policy, which is a quality-of-service mechanism (discussed in Chapter 30), can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Virtual circuit will stop up (waiting) Busy

كثير من الازدحام
دانا بي نحتاج اطلاع

Closed-Loop Congestion Control Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols. We describe a few of them here.

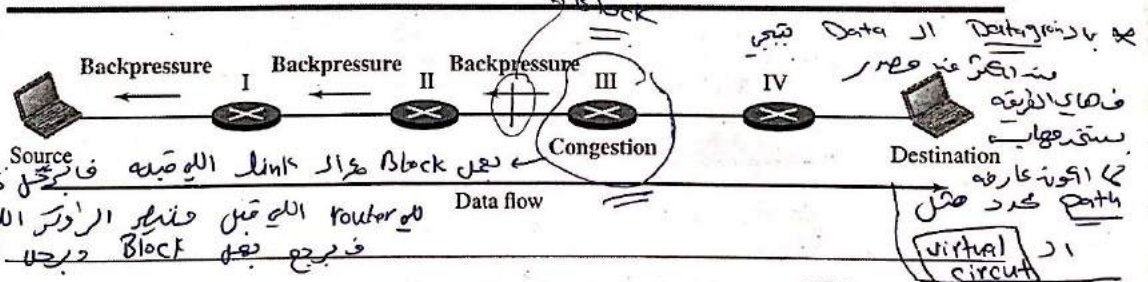
reuses راجعي طلب ل
حقيقه فاذا سفت ايه فا اعندي وفره جاد
Circuits Delay (we can't receive)

بيك احاول
ارحلته لادقاه
الحال

Backpressure The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream node or nodes, and so on. Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is coming. Figure 18.14 shows the idea of backpressure.

نفس برشكل
مسارها از
congestion node by node
اذا ما ادخل للروتين

Figure 18.14 Backpressure method for alleviating congestion



congestion
كل اار bit
له روتر الي قبله
فترجع Block ويرجع
للروتر المقدمه ب

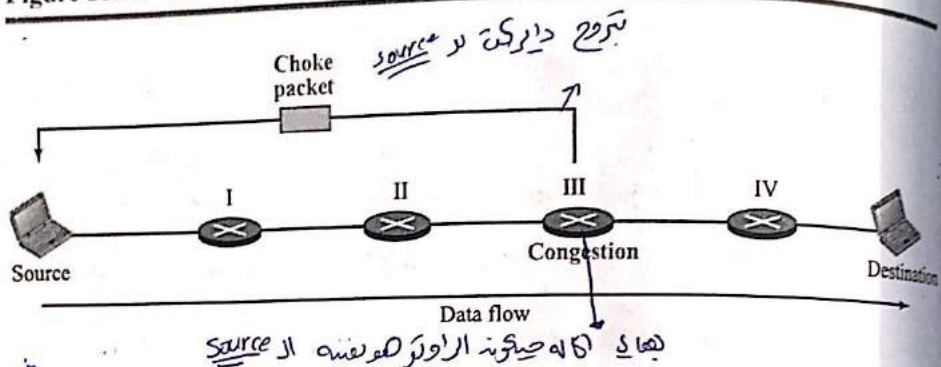
Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I informs the source of data to slow down. This, in time, alleviates the congestion. Note that the pressure on node III is moved backward to the source to remove the congestion.

It is important to stress that this type of congestion control can only be implemented in virtual-circuit. The technique cannot be implemented in a datagram network, in which a node (router) does not have the slightest knowledge of the upstream router.

virtual circuit

Choke Packet A choke packet is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke-packet methods. In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke-packet method, the warning is from the router, which has encountered congestion, directly to the source station. The intermediate nodes through which the packet has traveled are not warned. We will see an example of this type of control in ICMP (discussed in Chapter 19). When a router in the Internet is overwhelmed with IP datagrams, it may discard some of them, but it informs the source host, using a source quench ICMP message. The warning message goes directly to the source station; the intermediate routers do not take any action. Figure 18.15 shows the idea of a choke packet.

Figure 18.15 Choke packet



بکریج داریکن به source
 بھیادی که صورت گرفته از اونتر کوه رفته از source

Implicit Signaling In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is congestion somewhere in the network from other symptoms. For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down. We saw this type of signaling when we discuss TCP congestion control in Chapter 24.

بھیادی که صورت گرفته از اونتر کوه رفته از source
 بکریج داریکن به source

Explicit Signaling The node that experiences congestion can explicitly send a signal to the source or destination. The explicit-signaling method, however, is different from the choke-packet method. In the choke-packet method, a separate packet is used for this purpose; in the explicit-signaling method, the signal is included in the packets that carry data. Explicit signaling can occur in either the forward or the backward direction. This type of congestion control can be seen in an ATM network, discussed in Chapter 14.

بکریج داریکن به source
 بھیادی که صورت گرفته از اونتر کوه رفته از source

18.4 IPV4 ADDRESSES * IP address → Connection

The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address. An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. The IP address is the address of the connection, not the

address to connection

IP
↓
32-bit

MAC
↓
48-bit

host or the router, because if the device is moved to another network, the IP address may be changed.

IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses. IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

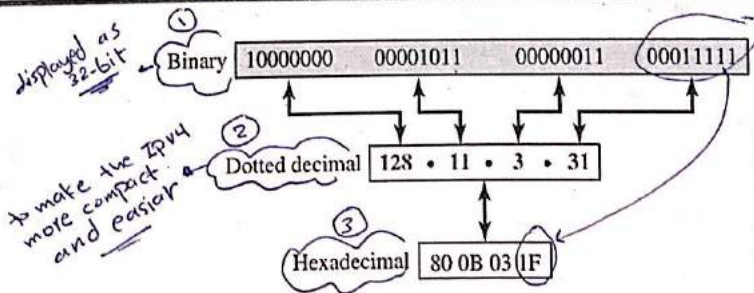
18.4.1 Address Space :-

A protocol like IPv4 that defines addresses has an address space. An address space is the total number of addresses used by the protocol. If a protocol uses b bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1). IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion). If there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation

There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16). In binary notation, an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces are usually inserted between each octet (8 bits). Each octet is often referred to as a byte. To make the IPv4 address more compact and easier to read, it is usually written in decimal form with a decimal point (dot) separating the bytes. This format is referred to as dotted-decimal notation. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255. We sometimes see an IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming. Figure 18.16 shows an IP address in the three discussed notations.

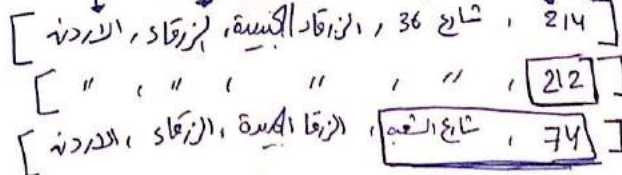
Figure 18.16 Three different notations in IPv4 addressing



Hierarchy in Addressing :-

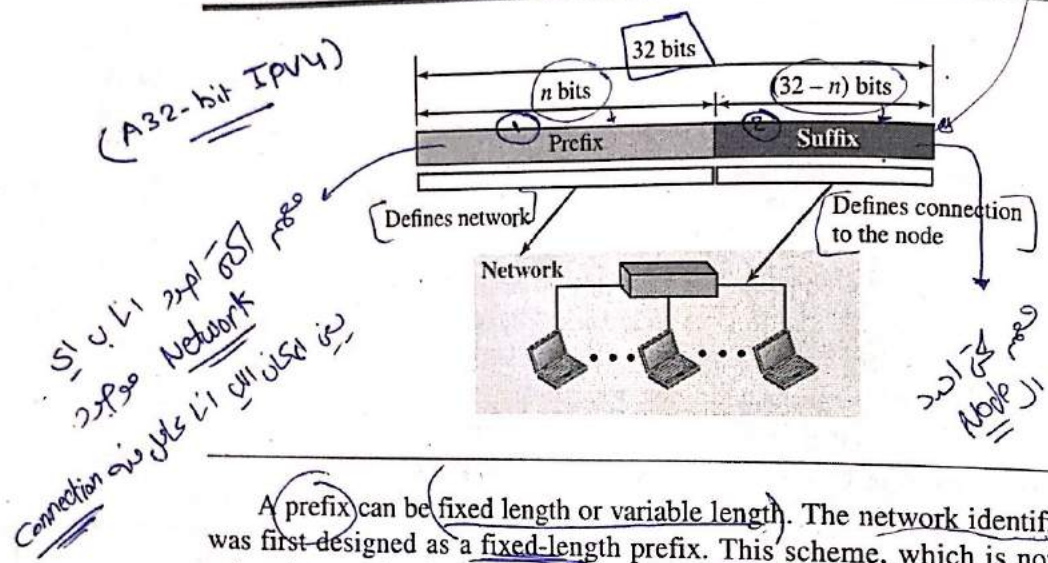
In any communication network that involves delivery, such as a telephone network or a postal network, the addressing system is hierarchical. In a postal network, the postal address (mailing address) includes the country, state, city, street, house number, and the

درد ال Hierarchy ال
يستعمل على ال Delivery
دستقل كمنه البروتوكول ال
يستعمل ال ال



* 962, 589 033 33
 name of the mail-recipient. Similarly, a telephone number is divided into the country code, area code, local exchange, and the connection.
 A 32-bit IPv4 address is also hierarchical, but divided only into two parts. The first part of the address, called the prefix defines the network; the second part of the address, called the suffix defines the node (connection of a device to the Internet). Figure 18.17 shows the prefix and suffix of a 32-bit IPv4 address. The prefix length is n bits and the suffix length is $(32 - n)$ bits.

Figure 18.17 Hierarchy in addressing



A prefix can be fixed length or variable length. The network identifier in the IPv4 was first designed as a fixed-length prefix. This scheme, which is now obsolete, is referred to as classful addressing. The new scheme, which is referred to as classless addressing, uses a variable-length network prefix. First, we briefly discuss classful addressing; then we concentrate on classless addressing.

18.4.2 Classful Addressing

When the Internet started, an IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one ($n = 8$, $n = 16$, and $n = 24$). The whole address space was divided into five classes (class A, B, C, D, and E), as shown in Figure 18.18. This scheme is referred to as classful addressing. Although classful addressing belongs to the past, it helps us to understand classless addressing, discussed later.

In class A, the network length is 8 bits, but since the first bit, which is 0, defines the class, we can have only seven bits as the network identifier. This means there are only $2^7 = 128$ networks in the world that can have a class A address.

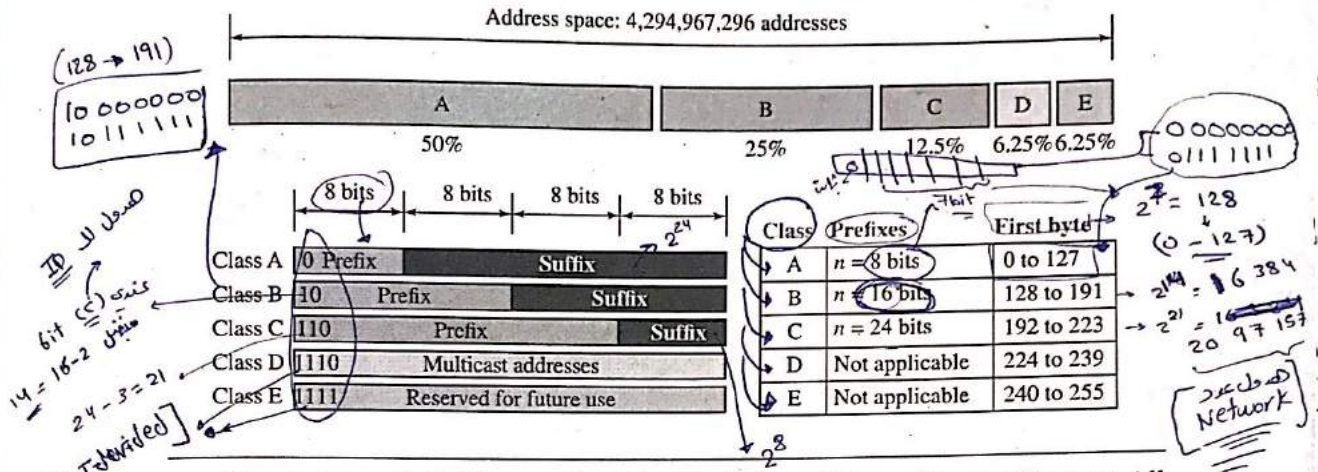
In class B, the network length is 16 bits, but since the first two bits, which are $(10)_2$, define the class, we can have only 14 bits as the network identifier. This means there are only $2^{14} = 16,384$ networks in the world that can have a class B address.

All addresses that start with $(110)_2$ belong to class C. In class C, the network length is 24 bits, but since three bits define the class, we can have only 21 bits as the network identifier. This means there are $2^{21} = 2,097,152$ networks in the world that can have a class C address.

Size of 5
 both small and large
 = 3. fixed length
 size of 5
 size of 5

Network size (address space) *
 Class size (مجموعه عناوين)
 Node عدد

Figure 18.18 Occupation of the address space in classful addressing



Class D is not divided into prefix and suffix. It is used for multicast addresses. All addresses that start with 1111 in binary belong to class E. As in Class D, Class E is not divided into prefix and suffix and is used as reserve.

Address Depletion

The reason that classful addressing has become obsolete is address depletion. Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals that needed to be connected to the Internet. To understand the problem, let us think about class A. This class can be assigned to only 128 organizations in the world, but each organization needs to have a single network (seen by the rest of the world) with 16,777,216 nodes (computers in this single network). Since there may be only a few organizations that are this large, most of the addresses in this class were wasted (unused). Class B addresses were designed for midsize organizations, but many of the addresses in this class also remained unused. Class C addresses have a completely different flaw in design. The number of addresses that can be used in each network (256) was so small that most companies were not comfortable using a block in this address class. Class E addresses were almost never used, wasting the whole class.

Subnetting and Supernetting

To alleviate address depletion, two strategies were proposed and, to some extent, implemented: subnetting and supernetting. In subnetting, a class A or class B block is divided into several subnets. Each subnet has a larger prefix length than the original network. For example, if a network in class A is divided into four subnets, each subnet has a prefix of $n_{sub} = 10$. At the same time, if all of the addresses in a network are not used, subnetting allows the addresses to be divided among several organizations. This idea did not work because most large organizations were not happy about dividing the block and giving some of the unused addresses to smaller organizations.

While subnetting was devised to divide a large block into smaller ones, supernetting was devised to combine several class C blocks into a larger block to be attractive to

Address Depletion =
 reason that classful addressing has become obsolete is address depletion.

Subnetting and Supernetting
 To alleviate address depletion, two strategies were proposed and, to some extent, implemented: subnetting and supernetting.

Handwritten calculations:
 $2^8 = 128$
 $2^{16} = 65536$
 $2^{24} = 16777216$
 Note: Class D is not divided into prefix and suffix.

Handwritten notes on Address Depletion:
 The reason that classful addressing has become obsolete is address depletion. Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals that needed to be connected to the Internet.

Handwritten notes on Subnetting and Supernetting:
 To alleviate address depletion, two strategies were proposed and, to some extent, implemented: subnetting and supernetting.

Handwritten notes at the bottom left:
 Subnetting: فمثلا احنا بقدر نستعمل صناديق الـ class C
 Supernetting: وعلشان انزلهم عندنا بعض صناديق الـ class C
 Note: 962 (2)

Handwritten box: **subnetting**

organizations that need more than the 256 addresses available in a class C block. This idea did not work either because it makes the routing of packets more difficult.

* Advantage of Classful Addressing

Although classful addressing had several problems and became obsolete, it had one advantage: Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately. In other words, the prefix length in classful addressing is inherent in the address; no extra information is needed to extract the prefix and the suffix.

18.4.3 Classless Addressing

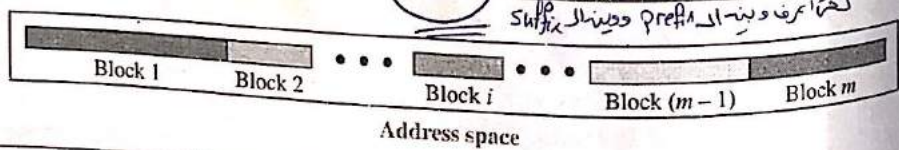
Subnetting and supernetting in classful addressing did not really solve the address depletion problem. With the growth of the Internet, it was clear that a larger address space was needed as a long-term solution. The larger address space, however, requires that the length of IP addresses also be increased, which means the format of the IP packets needs to be changed. Although the long-range solution has already been devised and is called IPv6 (discussed later), a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization. The short-term solution still uses IPv4 addresses, but it is called classless addressing. In other words, the class privilege was removed from the distribution to compensate for the address depletion.

There was another motivation for classless addressing. During the 1990s, Internet Service Providers (ISPs) came into prominence. An ISP is an organization that provides Internet access for individuals, small businesses, and midsize organizations that do not want to create an Internet site and become involved in providing Internet services (such as electronic mail) for their employees. An ISP can provide these services. An ISP is granted a large range of addresses and then subdivides the addresses (in groups of 1, 2, 4, 8, 16, and so on), giving a range of addresses to a household or a small business. The customers are connected via a dial-up modem, DSL, or cable modem to the ISP. However, each customer needs some IPv4 addresses.

In 1996, the Internet authorities announced a new architecture called classless addressing. In classless addressing, variable-length blocks are used that belong to no classes. We can have a block of 1 address, 2 addresses, 4 addresses, 128 addresses, and so on.

In classless addressing, the whole address space is divided into variable length blocks. The prefix in an address defines the block (network); the suffix defines the node (device). Theoretically, we can have a block of $2^0, 2^1, 2^2, \dots, 2^{32}$ addresses. One of the restrictions, as we discuss later, is that the number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses. Figure 18.19 shows the division of the whole address space into nonoverlapping blocks.

Figure 18.19 Variable-length blocks in classless addressing



بیسٹریج کے تحت ایس پی
 (ISP) کے ذریعے
 سہولتیں فراہم کرتے ہیں

* Network space کا ایک حصہ

16 M addresses

یہی ذریعہ ہے کہ ISP
 مختلف حصوں میں
 orange اور zain اور
 ورنہ umniac

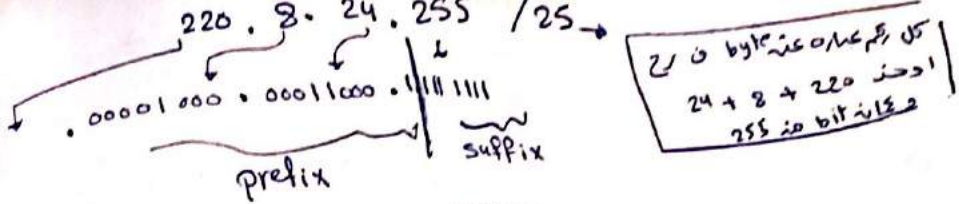
بیسٹریج کے تحت (Users)

ISP کے ذریعے

سہولتیں فراہم کرتے ہیں

Internet service

سہولتیں فراہم کرتے ہیں

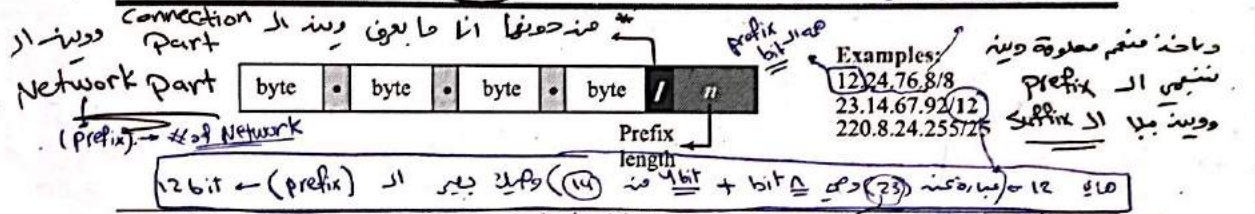
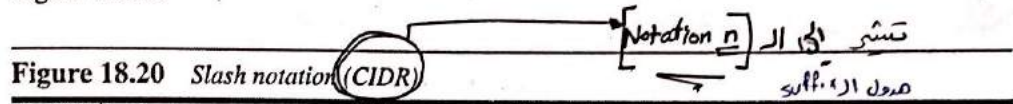


Unlike classful addressing, the prefix length in classless addressing is variable. We can have a prefix length that ranges from 0 to 32. The size of the network is inversely proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.

We need to emphasize that the idea of classless addressing can be easily applied to classful addressing. An address in class A can be thought of as a classless address in which the prefix length is 8. An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

Prefix Length: Slash Notation = -

The first question that we need to answer in classless addressing is how to find the prefix length if an address is given. Since the prefix length is not inherent in the address, we need to separately give the length of the prefix. In this case, the prefix length, n , is added to the address, separated by a slash. The notation is informally referred to as slash notation and formally as classless interdomain routing or CIDR (pronounced cider) strategy. An address in classless addressing can then be represented as shown in Figure 18.20.



In other words, an address in classless addressing does not, per se, define the block or network to which the address belongs; we need to give the prefix length also.

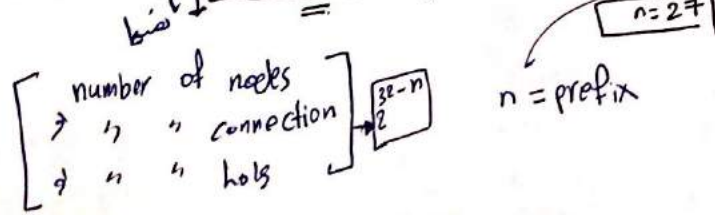
Extracting Information from an Address

Given any address in the block, we normally like to know three pieces of information about the block to which the address belongs: the number of addresses, the first address in the block, and the last address. Since the value of prefix length, n , is given, we can easily find these three pieces of information, as shown in Figure 18.21.

1. The number of addresses in the block is found as $N = 2^{32-n}$.
2. To find the first address, we keep the n leftmost bits and set the $(32 - n)$ rightmost bits all to 0s.
3. To find the last address, we keep the n leftmost bits and set the $(32 - n)$ rightmost bits all to 1s.

Example 18.1

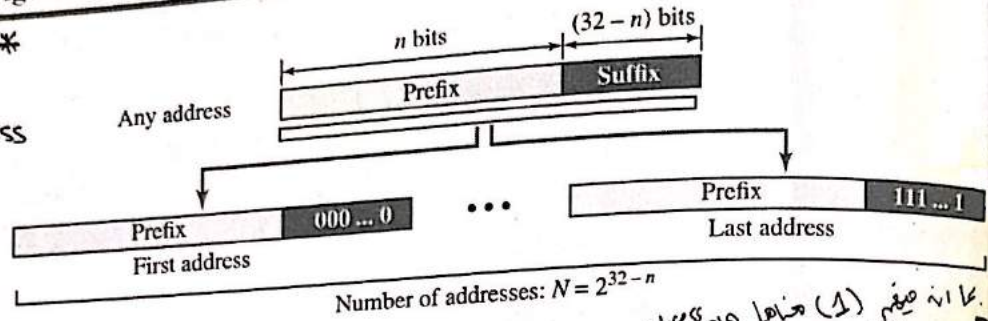
A classless address is given as 167.199.170.82/27. We can find the above three pieces of information as follows. The number of addresses in the network is $2^{32-27} = 2^5 = 32$ addresses.



address
 First address

Figure 18.21 Information extraction in classless addressing

- * من خلال اي address عرف :-
- 1) Number of address
 - 2) First address
 - 3) Last address.



The first address can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

Address: 167.199.170.82/27	10100111	11000111	10101010	01010010
First address: 167.199.170.64/27	10100111	11000111	10101010	01000000

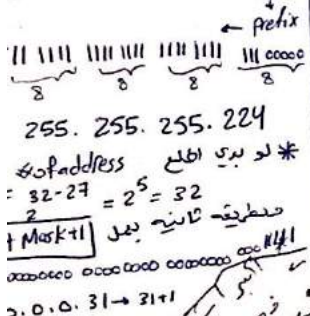
The last address can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

Address: 167.199.170.82/27	10100111	11000111	10101010	01011111
Last address: 167.199.170.95/27	10100111	11000111	10101010	01011111

[Network Id]

Address Mask

* Network mask =
 Set the prefix bits to 1's
 and the suffix bits to 0's
 ex :: 167.149.170.82/27



Another way to find the first and last addresses in the block is to use the address mask. The address mask is a 32-bit number in which the n leftmost bits are set to 1s and the rest of the bits ($32 - n$) are set to 0s. A computer can easily find the address mask because it is the complement of $(2^{32-n} - 1)$. The reason for defining a mask in this way is that it can be used by a computer program to extract the information in a block, using the three bit-wise operations NOT, AND, and OR.

1. The number of addresses in the block $N = \text{NOT}(\text{mask}) + 1$.
2. The first address in the block = (Any address in the block) AND (mask).
3. The last address in the block = (Any address in the block) OR [(NOT (mask))].

Example 18.2

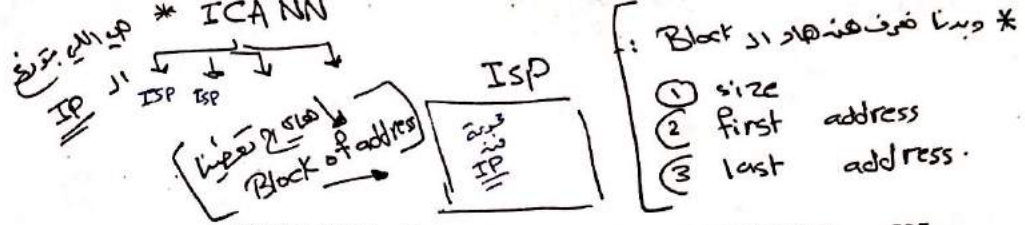
We repeat Example 18.1 using the mask. The mask in dotted-decimal notation is 255.255.255.224. The AND, OR, and NOT operations can be applied to individual bytes using calculators and applets at the book website.

Number of addresses in the block: $N = \text{NOT}(\text{mask}) + 1 = 0.0.0.31 + 1 = 32$ addresses
 First address: * First = (address) AND (mask) = 167.199.170.64
 Last address: Last = (address) OR (NOT mask) = 167.199.170.95

255.255.255.224
 Network address = First address

any given address

IP Block of address



CHAPTER 18 INTRODUCTION TO NETWORK LAYER 535

* هاد address موجود عنده نتورك *

Example 18.3

In classless addressing, an address cannot per se define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks. Some of them are shown below with the value of the prefix associated with that block.

Prefix length: 16	Block:	230.8.0.0	to	230.8.255.255
Prefix length: 20	Block:	230.8.(6.0)	to	230.8.(31)255
Prefix length: 26	Block:	230.8.24.0	to	230.8.24.63
Prefix length: 27	Block:	230.8.24.32	to	230.8.24.63
Prefix length: 29	Block:	230.8.24.56	to	230.8.24.63
Prefix length: 31	Block:	230.8.24.56	to	230.8.24.57

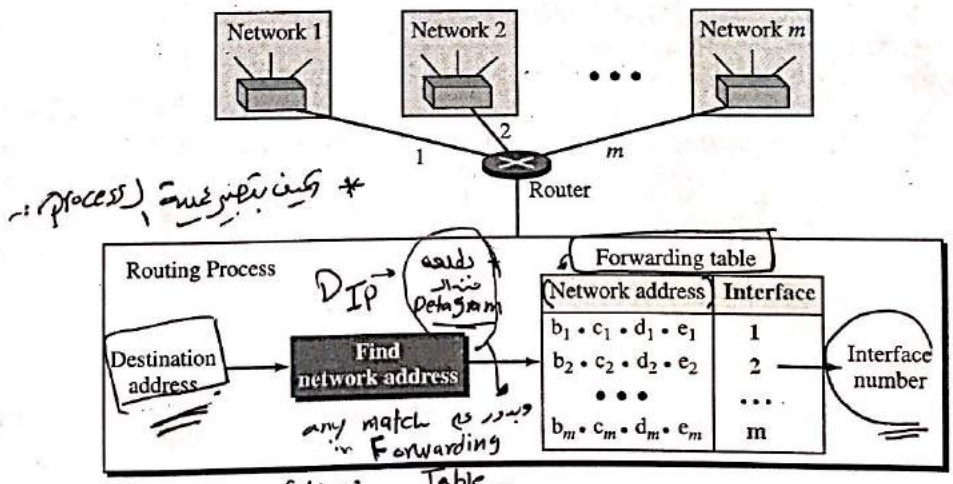
درونه مشن صفین
Slash notation
بدي افرق
Network connection

عشان اعرف لوه صبه اي Class
بجول ان 255
BINARY
ن سلايه 31
عبارت منه
Class D

Network Address :-

The above examples show that, given any address, we can find all information about the block. The first address, the network address, is particularly important because it is used in routing a packet to its destination network. For the moment, let us assume that an internet is made of m networks and a router with m interfaces. When a packet arrives at the router from any source host, the router needs to know to which network the packet should be sent: from which interface the packet should be sent out. When the packet arrives at the network, it reaches its destination host using another strategy that we discuss later. Figure 18.22 shows the idea. After the network address has been

Figure 18.22 Network address



* كين بغيره (process) *

found, the router consults its forwarding table to find the corresponding interface from which the packet should be sent out. The network address is actually the identifier of the network; each network is identified by its network address.

next node
Id
Mac address
عنه طلاقه
blood cast

Block Allocation

The next issue in classless addressing is block allocation. How are the blocks allocated? The ultimate responsibility of block allocation is given to a global authority called the Internet Corporation for Assigned Names and Numbers (ICANN). However, ICANN does not normally allocate addresses to individual Internet users. It assigns a large block of addresses to an ISP (or a larger organization that is considered an ISP in this case). For the proper operation of the CIDR, two restrictions need to be applied to the allocated block.

1. The number of requested addresses, N , needs to be a power of 2. The reason is that $N = 2^{32-n}$ or $n = 32 - \log_2 N$. If N is not a power of 2, we cannot have an integer value for n .
2. The requested block needs to be allocated where there is an adequate number of contiguous addresses available in the address space. However, there is a restriction on choosing the first address in the block. The first address needs to be divisible by the number of addresses in the block. The reason is that the first address needs to be the prefix followed by $(32 - n)$ number of 0s. The decimal value of the first address is then

first address = (prefix in decimal) $\times 2^{32-n}$ = (prefix in decimal) $\times N$.

Example 18.4

An ISP has requested a block of 1000 addresses. Since 1000 is not a power of 2, 1024 addresses are granted. The prefix length is calculated as $n = 32 - \log_2(1024) = 22$. An available block, 18.14.120/22, is granted to the ISP. It can be seen that the first address in decimal is 302,910,464, which is divisible by 1024.

Subnetting

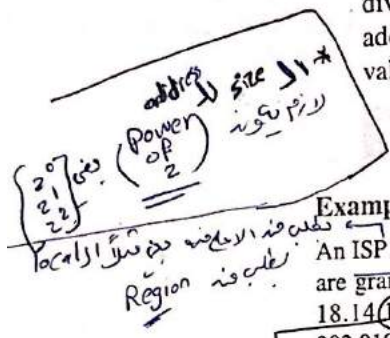
More levels of hierarchy can be created using subnetting. An organization (or an ISP) that is granted a range of addresses may divide the range into several subranges and assign each subrange to a subnetwork (or subnet). Note that nothing stops the organization from creating more levels. A subnetwork can be divided into several sub-subnetworks. A sub-subnetwork can be divided into several sub-sub-subnetworks, and so on.

Designing Subnets

The subnetworks in a network should be carefully designed to enable the routing of packets. We assume the total number of addresses granted to the organization is N , the prefix length is n , the assigned number of addresses to each subnetwork is N_{sub} , and the prefix length for each subnetwork is n_{sub} . Then the following steps need to be carefully followed to guarantee the proper operation of the subnetworks.

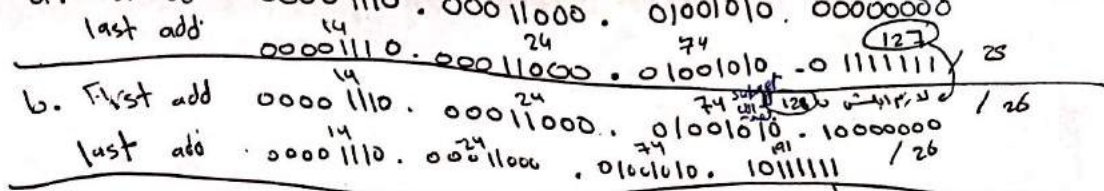
- The number of addresses in each subnetwork should be a power of 2.
- The prefix length for each subnetwork should be found using the following formula:

$n_{sub} = 32 - \log_2 N_{sub}$



Handwritten notes in Arabic: 'مجموعة كذا' (this group), 'هذه تسمى ال dot' (this is called the dot), 'بمدي 16 bit' (with 16 bits), 'في بعض' (in some), 'البيانات لتستمر اذا عاد' (the data continues if it returns), 'Block'.

Handwritten notes in Arabic: 'دقة خارج يقدر يرضح له' (precision outside can be satisfied), '1024 addresses' (1024 addresses), 'n = 32 - log2(1024) = 22' (n = 32 - log2(1024) = 22), 'ال address المرص' (the reserved address), 'بالسال' (in decimal).



The starting address in each subnetwork should be divisible by the number of addresses in that subnetwork. This can be achieved if we first assign addresses to larger subnetworks.

Finding Information about Each Subnetwork

After designing the subnetworks, the information about each subnetwork, such as first and last address, can be found using the process we described to find the information about each network in the Internet.

Example 18.5

An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have 3 subblocks of addresses to use in its three subnets: one subblock of 10 addresses, one subblock of 60 addresses, and one subblock of 120 addresses. Design the subblocks.

Solution

There are $2^{32-24} = 256$ addresses in this block. The first address is 14.24.74.0/24; the last address is 14.24.74.255/24. To satisfy the third requirement, we assign addresses to subblocks, starting with the largest and ending with the smallest one.

- a. The number of addresses in the largest subblock, which requires 120 addresses, is not a power of 2. We allocate 128 addresses. The subnet mask for this subnet can be found as $n_1 = 32 - \log_2 128 = 25$. The first address in this block is 14.24.74.0/25; the last address is 14.24.74.127/25.
- b. The number of addresses in the second largest subblock, which requires 60 addresses, is not a power of 2 either. We allocate 64 addresses. The subnet mask for this subnet can be found as $n_2 = 32 - \log_2 64 = 26$. The first address in this block is 14.24.74.128/26; the last address is 14.24.74.191/26.
- c. The number of addresses in the smallest subblock, which requires 10 addresses, is not a power of 2 either. We allocate 16 addresses. The subnet mask for this subnet can be found as $n_3 = 32 - \log_2 16 = 28$. The first address in this block is 14.24.74.192/28; the last address is 14.24.74.207/28.

If we add all addresses in the previous subblocks, the result is 208 addresses, which means 48 addresses are left in reserve. The first address in this range is 14.24.74.208. The last address is 14.24.74.255. We don't know about the prefix length yet. Figure 18.23 shows the configuration of blocks. We have shown the first address in each block.

Address Aggregation

One of the advantages of the CIDR strategy is address aggregation (sometimes called address summarization or route summarization). When blocks of addresses are combined to create a larger block, routing can be done based on the prefix of the larger block. ICANN assigns a large block of addresses to an ISP. Each ISP in turn divides its assigned block into smaller subblocks and grants the subblocks to its customers.

Example 18.6

Figure 18.24 shows how four small blocks of addresses are assigned to four organizations by an ISP. The ISP combines these four blocks into one single block and advertises the larger block to the rest of the world. Any packet destined for this larger block should be sent to this ISP. It is the responsibility of the ISP to forward the packet to the appropriate organization. This is similar to

15.05
14.24.74.0/24

to find New Prefix length

128 + 63
192 + 17

first address = Network address

1cc 5

first + Net Id + last



you need to find the number of sub-Net required
 $\log_2(N_{sub})$
128

11001111
14.24.74.208
14.24.74.255
prefix
لا نه لسا صحت خارج ستور ال
sub-Net block
المعنى
المعنى

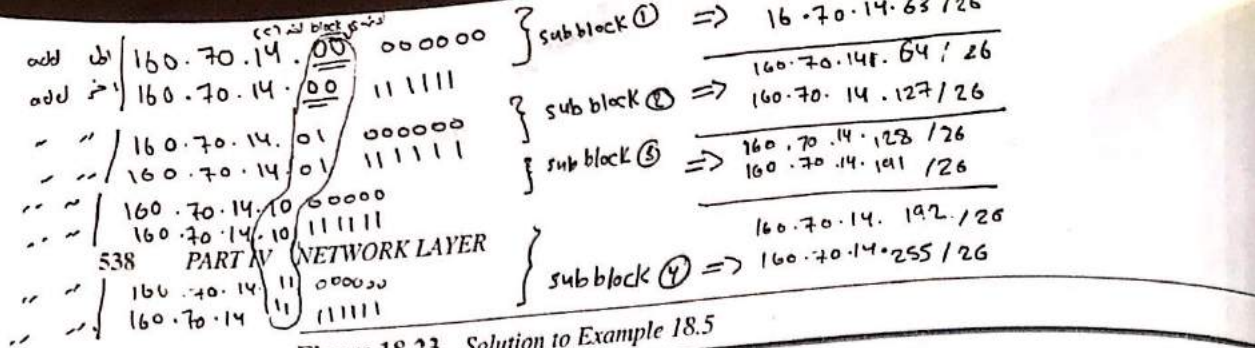


Figure 18.23 Solution to Example 18.5

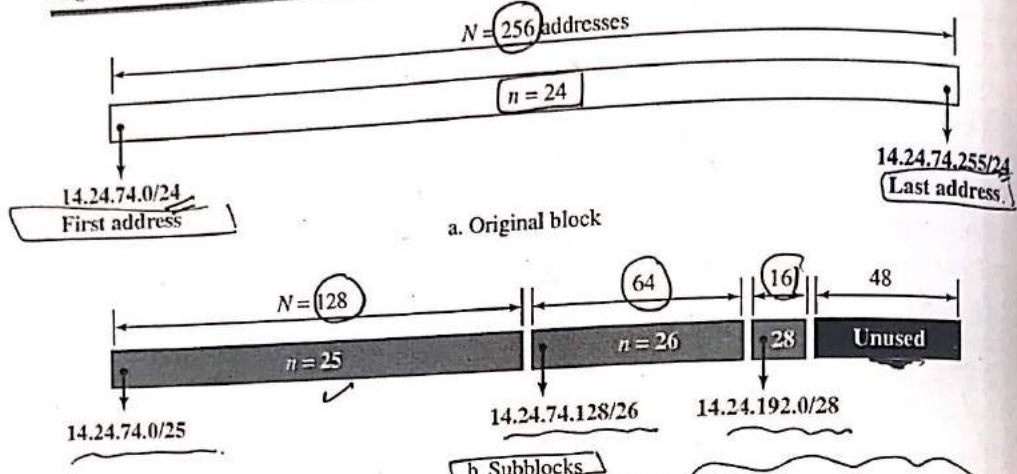
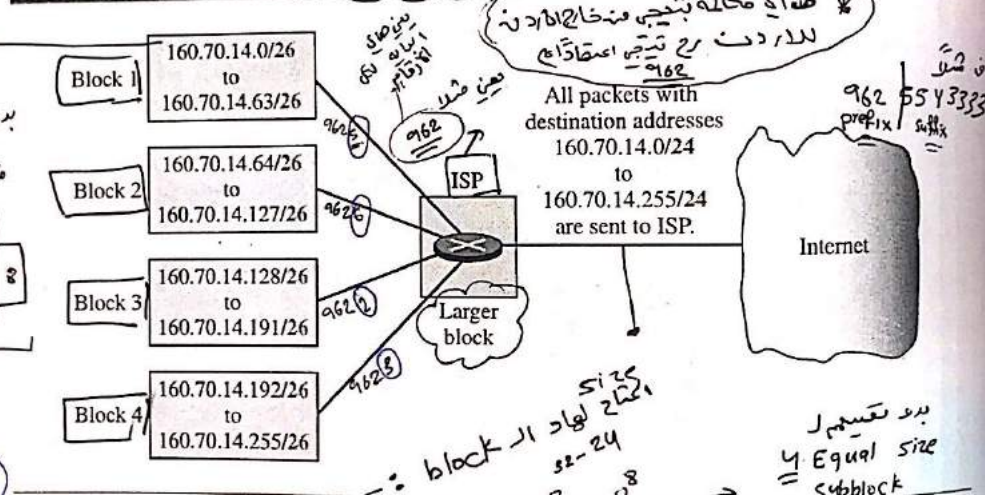
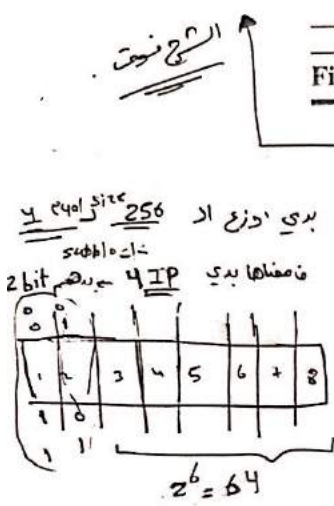


Figure 18.24 Example of address aggregation



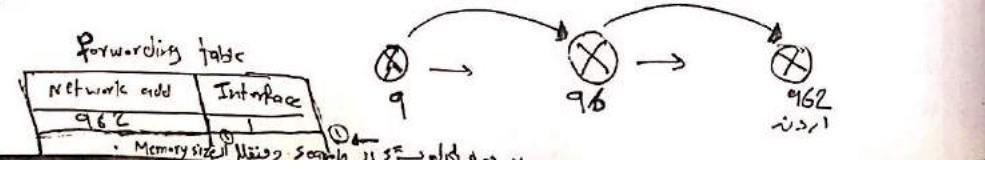
Handwritten notes in Arabic and English:

- بري اوزع ال 256 (Distribute 256)
- نصنعاها برى 4 IP (We made them 4 IP)
- 2 bit (2 bit)
- $2^6 = 64$
- يستعمل ال Delivery (Used Delivery)
- Hyarchical (Hierarchical)
- تقسيم ال address (Division of address)
- * فمثلا كما برى اعطهم (For example, as we gave them)
- forwarding table (forwarding table)
- اكتبه مراح اعط كل (Write the stages, give each)
- سكان ال لاون (Residents of the area)
- فيس بروج جفا ال (Fees for the project)
- Network address = 962
- دربعا مثلا ال (For example, the interface)
- (1) رايه ا بيته (1) his house
- ع مكانه كافي و... (Enough place and...)

routing we can find in a postal network. All packages coming from outside a country are sent first to the capital and then distributed to the corresponding destination.

Special Addresses - Before finishing the topic of addresses in IPv4, we need to mention five special addresses that are used for special purposes: *this-host* address, *limited-broadcast* address, *loopback* address, *private* addresses, and *multicast* addresses.

This-host Address - The only address in the block 0.0.0.0/32 is called the *this-host* address. It is used whenever a host needs to send an IP datagram but it does not know its own address to use as the source address. We will see an example of this case in the next section.



Limited-broadcast Address

The only address in the block 255.255.255.255/32 is called the limited-broadcast address. It is used whenever a router or a host needs to send a datagram to all devices in a network. The routers in the network, however, block the packet having this address as the destination; the packet cannot travel outside the network.

Loopback Address

The block 127.0.0.0/8 is called the loopback address. A packet with one of the addresses in this block as the destination address never leaves the host; it will remain in the host. Any address in the block is used to test a piece of software in the machine. For example, we can write a client and a server program in which one of the addresses in the block is used as the server address. We can test the programs using the same host to see if they work before running them on different computers.

Private Addresses

Four blocks are assigned as private addresses: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, and 169.254.0.0/16. We will see the applications of these addresses when we discuss NAT later in the chapter.

Multicast Addresses

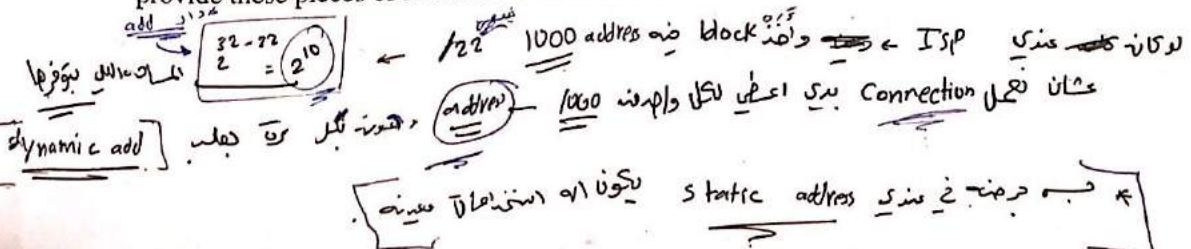
The block 224.0.0.0/4 is reserved for multicast addresses. We discuss these addresses later in the chapter.

18.4.4 Dynamic Host Configuration Protocol (DHCP)

We have seen that a large organization or an ISP can receive a block of addresses directly from ICANN and a small organization can receive a block of addresses from an ISP. After a block of addresses are assigned to an organization, the network administration can manually assign addresses to the individual hosts or routers. However, address assignment in an organization can be done automatically using the Dynamic Host Configuration Protocol (DHCP). DHCP is an application-layer program, using the client-server paradigm, that actually helps TCP/IP at the network layer.

DHCP has found such widespread use in the Internet that it is often called a plug-and-play protocol. In can be used in many situations. A network manager can configure DHCP to assign permanent IP addresses to the host and routers. DHCP can also be configured to provide temporary, on demand, IP addresses to hosts. The second capability can provide a temporary IP address to a traveller to connect her laptop to the Internet while she is staying in the hotel. It also allows an ISP with 1000 granted addresses to provide services to 4000 households, assuming not more than one-fourth of customers use the Internet at the same time.

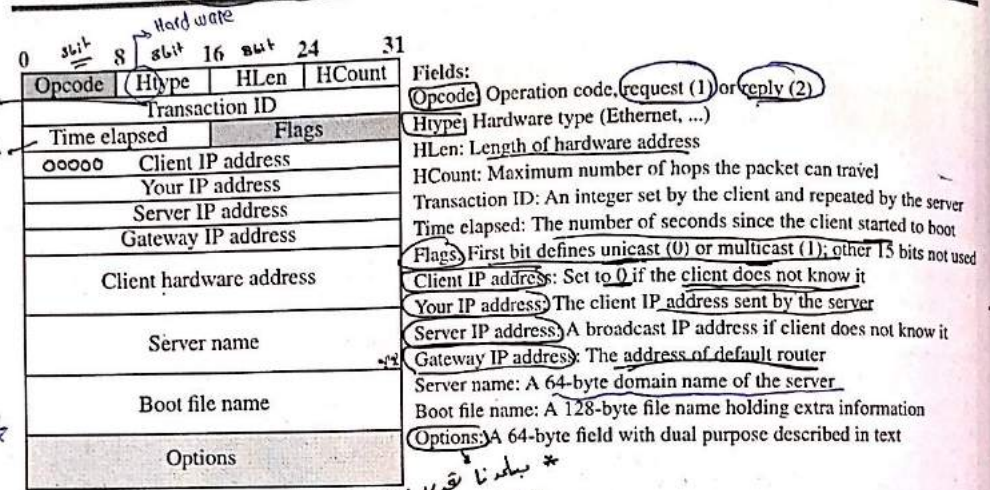
In addition to its IP address, a computer also needs to know the network prefix (or address mask). Most computers also need two other pieces of information, such as the address of a default router to be able to communicate with other networks and the address of a name server to be able to use names instead of addresses, as we will see in Chapter 26. In other words, four pieces of information are normally needed: the computer address, the prefix, the address of a router, and the IP address of a name server. DHCP can be used to provide these pieces of information to the host.



DHCP Message Format

DHCP is a client-server protocol in which the client sends a request message and the server returns a response message. Before we discuss the operation of DHCP, let us show the general format of the DHCP message in Figure 18.25. Most of the fields are explained in the figure, but we need to discuss the option field, which plays a very important role in DHCP.

Figure 18.25 DHCP message format



رقم هاء النوع / رقم النوع
الوقت الذي استغرقه عمال قبل هذا الرد

برك بعض معلومات عن
IP address

prefix length (IP address)
subnet mask
له مكانة معرف معلومات
عنوان block الك
تاسع

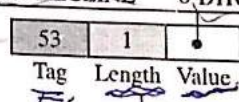
gateway IP address
Domain name server (DNS)

دقيقة بوقت
Domain Name
IP address ويرجى
قبل ما يري ان رسال
هذا تلفونه بحيث
اسمه ويحل
Call
فما اجب عن ال
يرجى ال

The 64-byte option field has a dual purpose. It can carry either additional information or some specific vendor information. The server uses a number, called a **magic cookie**, in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie. If present, the next 60 bytes are options. An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-length value field. There are several tag fields that are mostly used by vendors. If the tag field is 53, the value field defines one of the 8 message types shown in Figure 18.26. We show how these message types are used by DHCP.

Figure 18.26 Option format

- 1 DHCPDISCOVER
- 2 DHCPOFFER
- 3 DHCPREQUEST
- 4 DHCPDECLINE
- 5 DHCPACK
- 6 DHCPNACK
- 7 DHCPRELEASE
- 8 DHCPINFORM

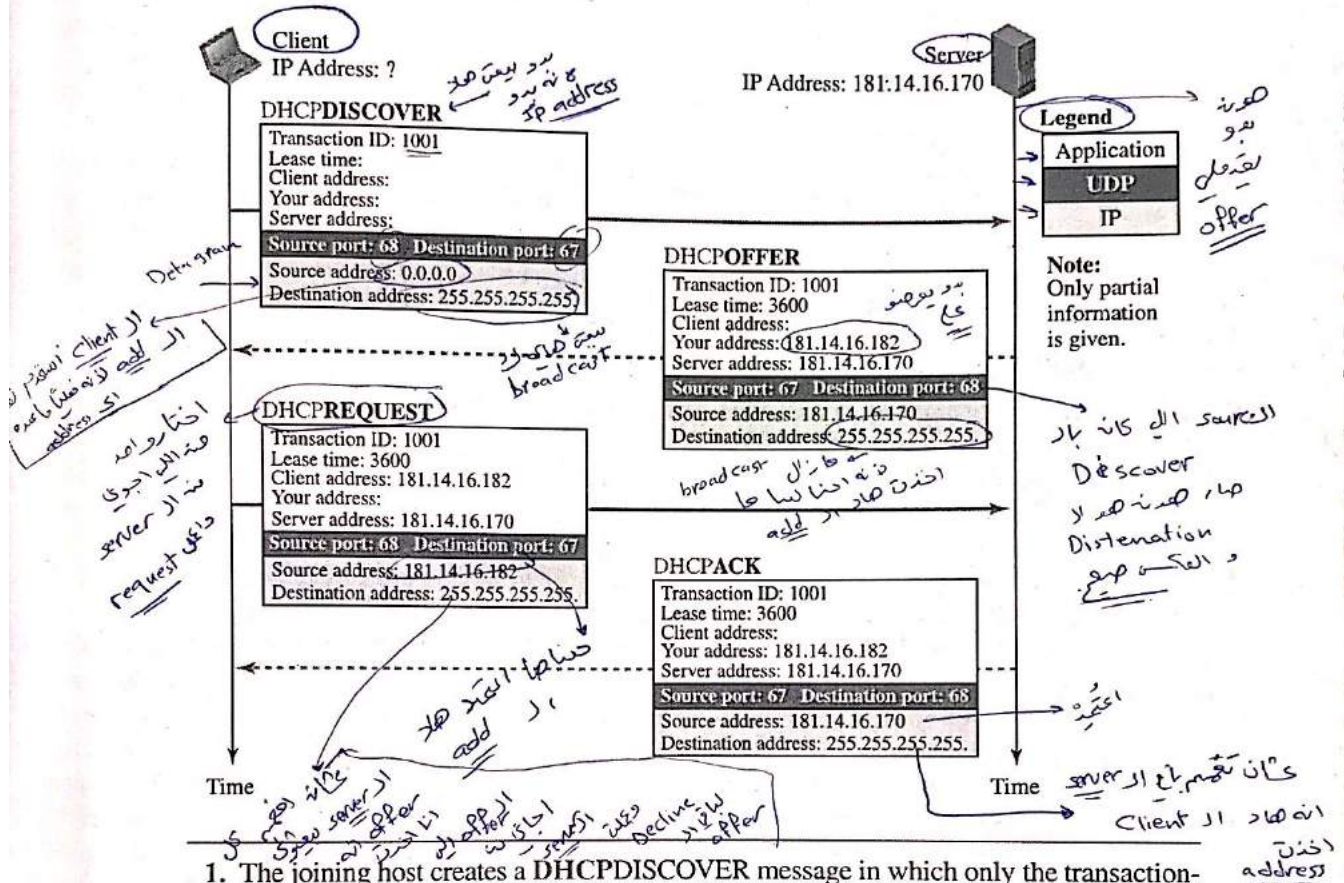


DHCP Operation

Figure 18.27 shows a simple scenario.

احنا ال client
نا بدي نتواصل مع ال server
IP address

Figure 18.27 Operation of DHCP



1. The joining host creates a DHCPDISCOVER message in which only the transaction-ID field is set to a random number. No other field can be set because the host has no knowledge with which to do so. This message is encapsulated in a UDP user datagram with the source port set to 68 and the destination port set to 67. We will discuss the reason for using two well-known port numbers later. The user datagram is encapsulated in an IP datagram with the source address set to 0.0.0.0 ("this host") and the destination address set to 255.255.255.255 (broadcast address). The reason is that the joining host knows neither its own address nor the server address.
2. The DHCP server or servers (if more than one) responds with a DHCPOFFER message in which the your address field defines the offered IP address for the joining host and the server address field includes the IP address of the server. The message also includes the lease time for which the host can keep the IP address. This message is encapsulated in a user datagram with the same port numbers, but in the reverse order. The user datagram in turn is encapsulated in a datagram with the server address as the source IP address, but the destination address is a broadcast address, in which the server allows other DHCP servers to receive the offer and give a better offer if they can.

3. The joining host receives one or more offers and selects the best of them. The joining host then sends a **DHCPREQUEST** message to the server that has given the best offer. The fields with known value are set. The message is encapsulated in a user datagram with port numbers as the first message. The user datagram is encapsulated in an IP datagram with the source address set to the new client address, but the destination address still is set to the broadcast address to let the other servers know that their offer was not accepted.
4. Finally, the selected server responds with a **DHCPACK** message to the client if the offered IP address is valid. If the server cannot keep its offer (for example, if the address is offered to another host in between), the server sends a **DHCPNACK** message and the client needs to repeat the process. This message is also broadcast to let other servers know that the request is accepted or rejected.

Two Well-Known Ports

We said that the DHCP uses two well-known ports (68 and 67) instead of one well-known and one ephemeral. The reason for choosing the well-known port 68 instead of an ephemeral port for the client is that the response from the server to the client is broadcast. Remember that an IP datagram with the limited broadcast message is delivered to every host on the network. Now assume that a DHCP client and a DAYTIME client, for example, are both waiting to receive a response from their corresponding server and both have accidentally used the same temporary port number (56017, for example). Both hosts receive the response message from the DHCP server and deliver the message to their clients. The DHCP client processes the message; the DAYTIME client is totally confused with a strange message received. Using a well-known port number prevents this problem from happening. The response message from the DHCP server is not delivered to the DAYTIME client, which is running on the port number 56017, not 68. The temporary port numbers are selected from a different range than the well-known port numbers.

The curious reader may ask what happens if two DHCP clients are running at the same time. This can happen after a power failure and power restoration. In this case the messages can be distinguished by the value of the transaction ID, which separates each response from the other.

Using FTP

The server does not send all of the information that a client may need for joining the network. In the **DHCPACK** message, the server defines the pathname of a file in which the client can find complete information such as the address of the DNS server. The client can then use a file transfer protocol to obtain the rest of the needed information.

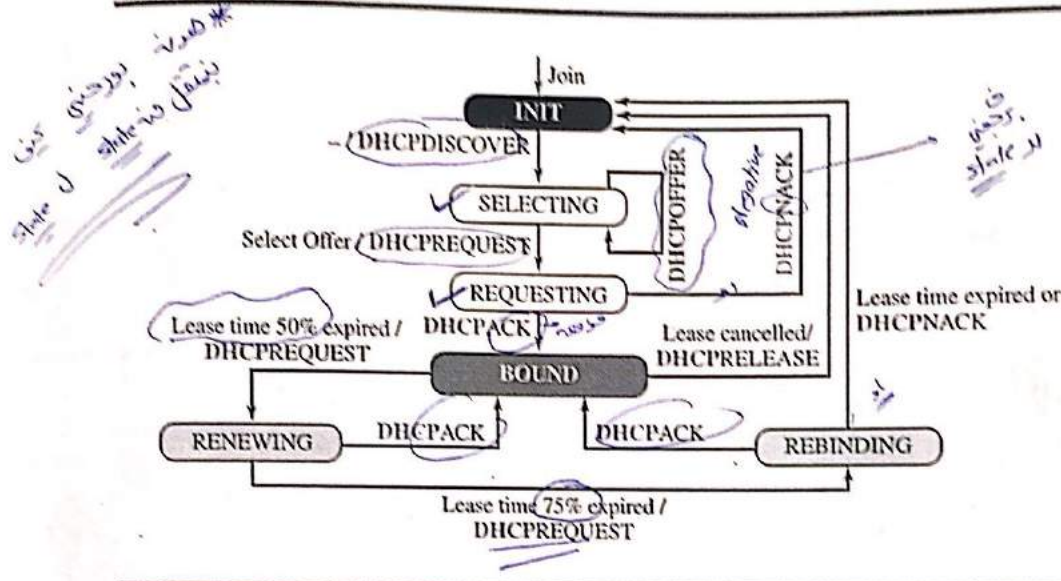
Error Control

DHCP uses the service of UDP, which is not reliable. To provide error control, DHCP uses two strategies. First, DHCP requires that UDP use the checksum. As we will see in Chapter 24, the use of the checksum in UDP is optional. Second, the DHCP client uses timers and a retransmission policy if it does not receive the DHCP reply to a request. However, to prevent a traffic jam when several hosts need to retransmit a request (for example, after a power failure), DHCP forces the client to use a random number to set its timers.

Transition States

The previous scenarios we discussed for the operation of the DHCP were very simple. To provide dynamic address allocation, the DHCP client acts as a state machine that performs transitions from one state to another depending on the messages it receives or sends. Figure 18.28 shows the transition diagram with the main states.

Figure 18.28 FSM for the DHCP client



When the DHCP client first starts, it is in the **INIT** state (initializing state). The client broadcasts a discover message. When it receives an offer, the client goes to the **SELECTING** state. While it is there, it may receive more offers. After it selects an offer, it sends a request message and goes to the **REQUESTING** state. If an ACK arrives while the client is in this state, it goes to the **BOUND** state and uses the IP address. When the lease is 50 percent expired, the client tries to renew it by moving to the **RENEWING** state. If the server renews the lease, the client moves to the **BOUND** state again. If the lease is not renewed and the lease time is 75 percent expired, the client moves to the **REBINDING** state. If the server agrees with the lease (ACK message arrives), the client moves to the **BOUND** state and continues using the IP address; otherwise, the client moves to the **INIT** state and requests another IP address. Note that the client can use the IP address only when it is in the **BOUND**, **RENEWING**, or **REBINDING** state. The above procedure requires that the client uses three timers: *renewal timer* (set to 50 percent of the lease time), *rebinding timer* (set to 75 percent of the lease time), and *expiration timer* (set to the lease time).

18.4.5 Network Address Resolution (NAT)

The distribution of addresses through ISPs has created a new problem. Assume that an ISP has granted a small range of addresses to a small business or a household. If the business grows or the household needs a larger range, the ISP may not be able to grant the demand because the addresses before and after the range may have already been allocated to other networks. In most situations, however, only a portion of computers in

يعني في كل مكان عند شركة وكانها مجموعة من (IP add) و في الشركة
 وشبكة تواصل يعني انه صار بعدها عدد اكبر من (IP) فكلها من (ISP) بس
 فاعينها يعني ان كل تلك الشركة كلها اكله ثم نستعمل من
 [Private add] مخصص

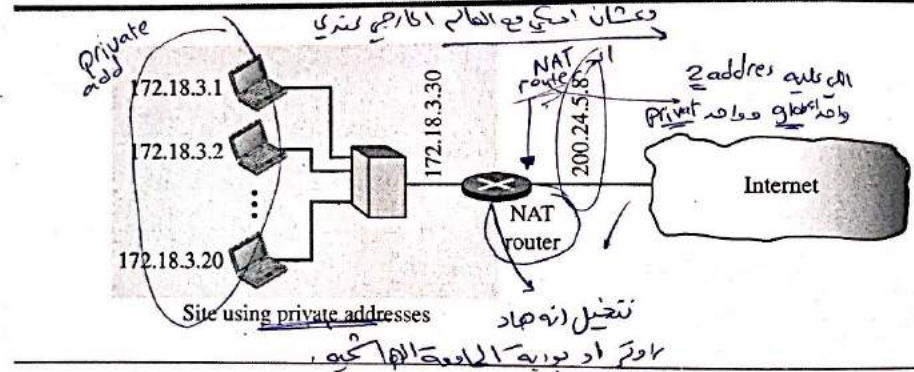
ع حال انه يدي الـ (private add) غير هيد مافيد او هيدتة ولو سكرتيرة العنم بهما فترتة ع حد رح يطلع
 عندهم اهنم الكافه وليس الرقم الـ (private) لغيا في رقم السكرتيرة
 طلب رقم الكافه وبعد هيد بطلب رقم سكرتيرة العنم

a small network need access to the Internet simultaneously. This means that the number of allocated addresses does not have to match the number of computers in the network. For example, assume that in a small business with 20 computers the maximum number of computers that access the Internet simultaneously is only 4. Most of the computers are either doing some task that does not need Internet access or communicating with each other. This small business can use the TCP/IP protocol for both internal and universal communication. The business can use 20 (or 25) addresses from the private block addresses (discussed before) for internal communication; five addresses for universal communication can be assigned by the ISP.

A technology that can provide the mapping between the private and universal addresses, and at the same time support virtual private networks, which we discuss in Chapter 32, is **Network Address Translation (NAT)**. The technology allows a site to use a set of private addresses for internal communication and a set of global Internet addresses (at least one) for communication with the rest of the world. The site must have only one connection to the global Internet through a NAT-capable router that runs NAT software. Figure 18.29 shows a simple implementation of NAT.

من خلاها قلنا حاجة
 هاياد connection
 ل (real IP address)
 بالتاي اتبنا الاسلوب
 الباني الـ هواد
 (Private address)

Figure 18.29 NAT



As the figure shows, the private network uses private addresses. The router that connects the network to the global address uses one private address and one global address. The private network is invisible to the rest of the Internet; the rest of the Internet sees only the NAT router with the address 200.24.5.8.

Address Translation

All of the outgoing packets go through the NAT router, which replaces the source address in the packet with the global NAT address. All incoming packets also pass through the NAT router, which replaces the destination address in the packet (the NAT router global address) with the appropriate private address. Figure 18.30 shows an example of address translation.

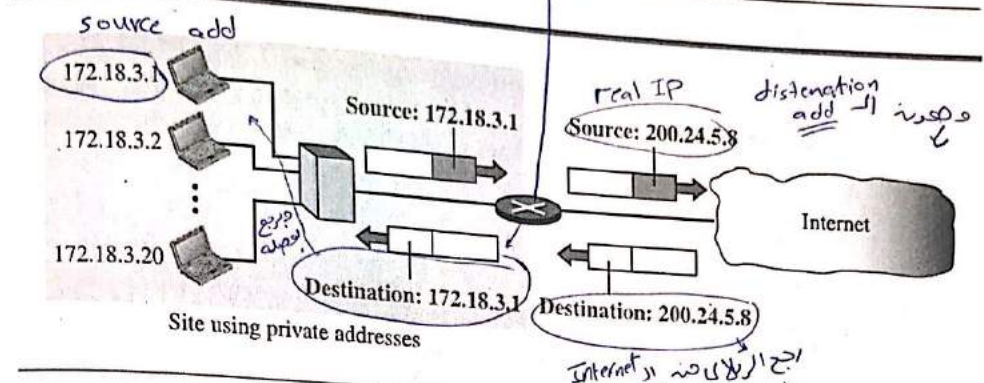
Translation Table

The reader may have noticed that translating the source addresses for an outgoing packet is straightforward. But how does the NAT router know the destination address for a packet coming from the Internet? There may be tens or hundreds of private IP

* يبين الـ NAT
 1 source add
 2 dest add
 Private add

IP diagram
Real IP add

Figure 18.30 Address translation

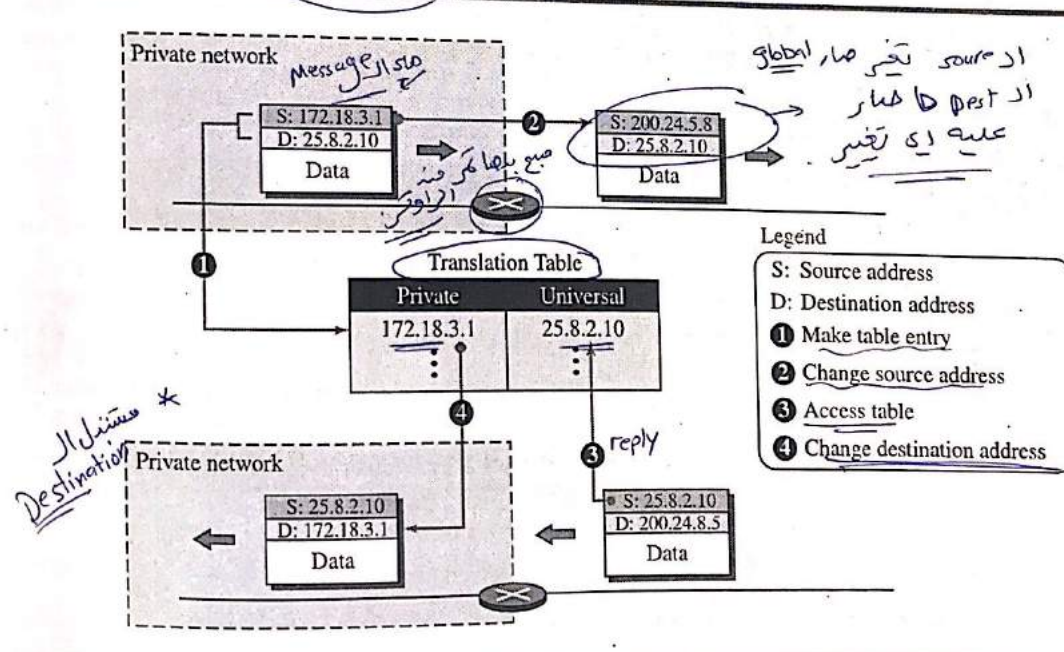


addresses, each belonging to one specific host. The problem is solved if the NAT router has a translation table.

Using One IP Address

In its simplest form, a translation table has only two columns: the private address and the external address (destination address of the packet). When the router translates the source address of the outgoing packet, it also makes note of the destination address—where the packet is going. When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet. Figure 18.31 shows the idea.

Figure 18.31 Translation



In this strategy, communication must always be initiated by the private network. The NAT mechanism described requires that the private network start the communication.

* لما يوصلنا صبح على ال (WhatsApp) ف بننا لفضهم انه لا حد لبيدة يكونه هو باعث مع
 Account من خلال غيري والبا هو دور يكونه في صبحنا
 Account من خلال غيري والبا هو دور يكونه في صبحنا
 (Dynamic Process) ف ابينش فيه انا سي يكونه جهازي
 server هههه (Fetch) PART IV NETWORK LAYER 546

As we will see, NAT is used mostly by ISPs that assign a single address to a customer. The customer, however, may be a member of a private network that has many private addresses. In this case, communication with the Internet is always initiated from the customer site, using a client program such as HTTP, TELNET, or FTP to access the corresponding server program. For example, when e-mail that originates from outside the network site is received by the ISP e-mail server, it is stored in the mailbox of the customer until retrieved with a protocol such as POP.

Using a Pool of IP Addresses

The use of only one global address by the NAT router allows only one private-network host to access a given external host. To remove this restriction, the NAT router can use a pool of global addresses. For example, instead of using only one global address (200.24.5.8), the NAT router can use four addresses (200.24.5.8, 200.24.5.9, 200.24.5.10, and 200.24.5.11). In this case, four private-network hosts can communicate with the same external host at the same time because each pair of addresses defines a separate connection. However, there are still some drawbacks. No more than four connections can be made to the same destination. No private-network host can access two external server programs (e.g., HTTP and TELNET) at the same time. And, likewise, two private-network hosts cannot access the same external server program (e.g., HTTP or TELNET) at the same time.

* To remove this restriction

Using Both IP Addresses and Port Addresses

To allow a many-to-many relationship between private-network hosts and external server programs, we need more information in the translation table. For example, suppose two hosts inside a private network with addresses 172.18.3.1 and 172.18.3.2 need to access the HTTP server on external host 25.8.3.2. If the translation table has five columns, instead of two, that include the source and destination port addresses and the transport-layer protocol, the ambiguity is eliminated. Table 18.1 shows an example of such a table.

اكاه
 * صلا بقدر استيز
 reply
 عند طريقه Parameter
 (port) عن منه ال
 Private address
 ال كيت من خلال امزله
 الرباعي كيت لانه هههه
 (2-private) نفس السبي
 هههه كيت
 حكو مع نفس ال
 2-Private address
 Destination

Table 18.1 Five-column translation table

Private address	Private port	External address	External port	Transport protocol
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
⋮	⋮	⋮	⋮	⋮

Note that when the response from HTTP comes back, the combination of source address (25.8.3.2) and destination port address (1401) defines the private network host to which the response should be directed. Note also that for this translation to work, the ephemeral port addresses (1400 and 1401) must be unique.

18.5 FORWARDING OF IP PACKETS

We discussed the concept of forwarding at the network layer earlier in this chapter. In this section, we extend the concept to include the role of IP addresses in forwarding. As we discussed before, forwarding means to place the packet in its route to its destination.

Since the Internet today is made of a combination of links (networks), forwarding means to deliver the packet to the next hop (which can be the final destination or the intermediate connecting device). Although the IP protocol was originally designed as a connectionless protocol, today the tendency is to change it to a connection-oriented protocol. We discuss both cases.

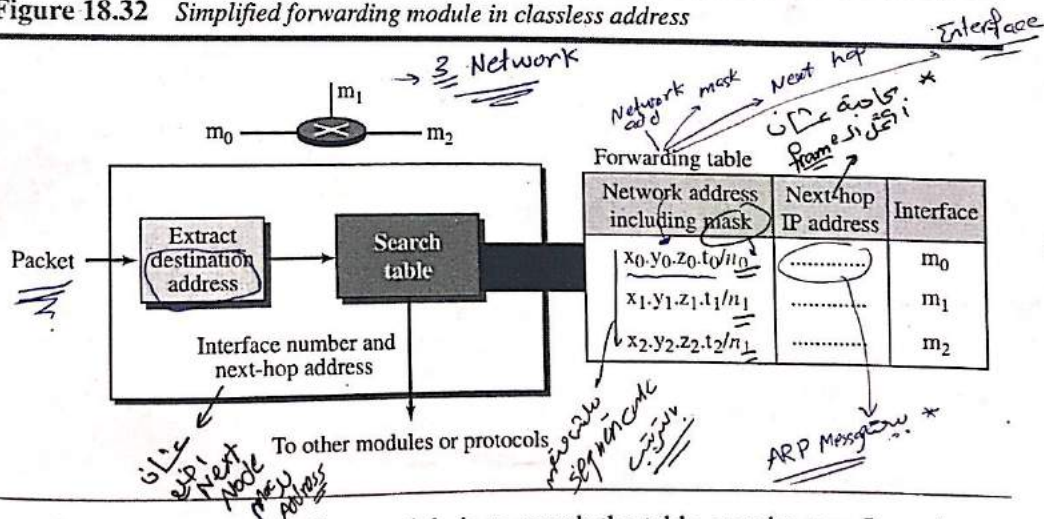
When IP is used as a connectionless protocol, forwarding is based on the destination address of the IP datagram; when the IP is used as a connection-oriented protocol, forwarding is based on the label attached to an IP datagram.

18.5.1 Forwarding Based on Destination Address (IP address)

We first discuss forwarding based on the destination address. This is a traditional approach, which is prevalent today. In this case, forwarding requires a host or a router to have a forwarding table. When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the next hop to deliver the packet to.

In classless addressing, the whole address space is one entity; there are no classes. This means that forwarding requires one row of information for each block involved. The table needs to be searched based on the network address (first address in the block). Unfortunately, the destination address in the packet gives no clue about the network address. To solve the problem, we need to include the mask (n) in the table. In other words, a classless forwarding table needs to include four pieces of information: the mask, the network address, the interface number, and the IP address of the next router (needed to find the link-layer address of the next hop, as we discussed in Chapter 9). However, we often see in the literature that the first two pieces are combined. For example, if n is 26 and the network address is 180.70.65.192, then one can combine the two as one piece of information: 180.70.65.192/26. Figure 18.32 shows a simple forwarding module and forwarding table for a router with only three interfaces.

Figure 18.32 Simplified forwarding module in classless address



The job of the forwarding module is to search the table, row by row. In each row, the n leftmost bits of the destination address (prefix) are kept and the rest of the bits (suffix) are set to 0s. If the resulting address (which we call the network address), matches with the address in the first column, the information in the next two columns is

extracted; otherwise the search continues. Normally, the last row has a default value in the first column (not shown in the figure), which indicates all destination addresses that did not match the previous rows.

Sometimes, the literature explicitly shows the value of the n leftmost bits that should be matched with the n leftmost bits of the destination address. The concept is the same, but the presentation is different. For example, instead of giving the address-mask combination of 180.70.65.192/26, we can give the value of the 26 leftmost bits as shown below.

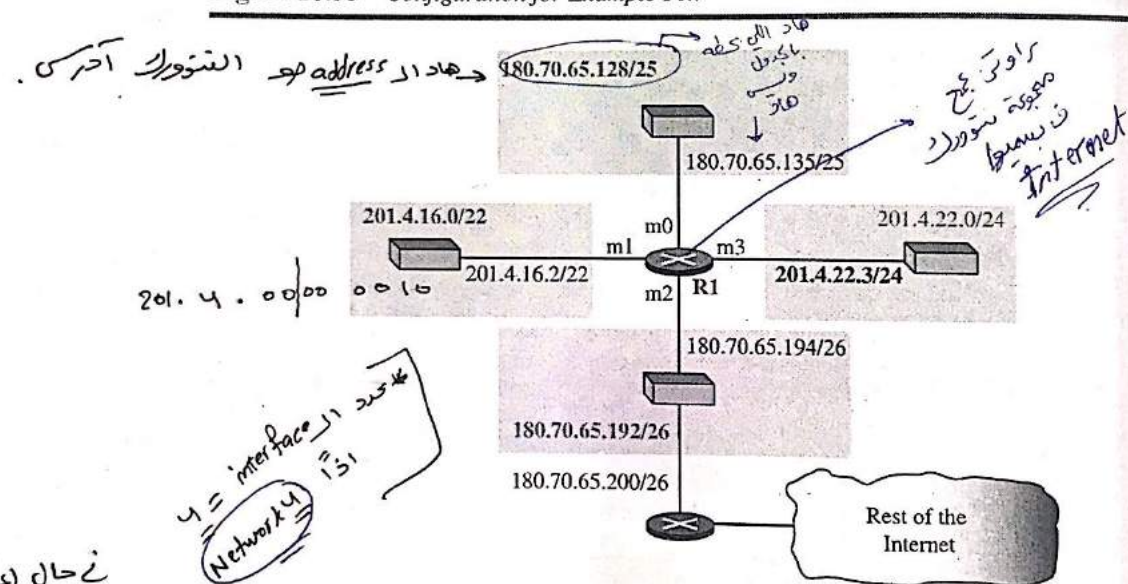
10110100 01000110 01000001 11

Note that we still need to use an algorithm to find the prefix and compare it with the bit pattern. In other words, the algorithm is still needed, but the presentation is different. We use this format in our forwarding tables in the exercises when we use smaller address spaces just for practice.

Example 18.7

Make a forwarding table for router R1 using the configuration in Figure 18.33.

Figure 18.33 Configuration for Example 18.7



Solution

Table 18.2 shows the corresponding table.

Table 18.2 Forwarding table for router R1 in Figure 18.33

Network address/mask	Next hop	Interface
180.70.65.192/26	—	m2
180.70.65.128/25	—	m0
201.4.22.0/24	—	m3
201.4.16.0/22	—	m1
Default	180.70.65.200	m2

في حال اطلاق
 DIP 180.70.65.140
 هذا اقرب من m0
 140 = 10001100

هذا مثال كان 240
 11110000
 برابطه متباين
 مع m0 و m2 في 25 bit
 اكون مرتبة الكلاسا
 من اقرب الى ابعث
 domain
 في حال اول فانني
 accept فانني

يحدد ال interface
 Network
 4 = interface
 131

always match
 في حال كونها
 Match
 مع ال interface
 في ال bit

Example 18.8

Instead of Table 18.2, we can use Table 18.3, in which the network address/mask is given in bits.

Table 18.3 Forwarding table for router R1 in Figure 18.33 using prefix bits

Leftmost bits in the destination address	Next hop	Interface
10110100 01000110 01000001 11 → 26 bit	—	m2
10110100 01000110 01000001 1 → 25 bit	—	m0
11001001 00000100 00011100 → 24 bit	—	m3
11001001 00000100 000100 → 22 bit	—	m1
Default	180.70.65.200	m2

26 bit
البيانات
المالزمة

180.70.65.128
المعرف عدد
Host
2⁶ - 2
= 62
↓
4

When a packet arrives whose leftmost 26 bits in the destination address match the bits in the first row, the packet is sent out from interface m2. When a packet arrives whose leftmost 25 bits in the address match the bits in the second row, the packet is sent out from interface m0, and so on. The table clearly shows that the first row has the longest prefix and the fourth row has the shortest prefix. The longer prefix means a smaller range of addresses; the shorter prefix means a larger range of addresses.

عدد البتات في الـ suffix
32 - 26 = 6
الـ suffix
= 2⁶ - 2
= 62
لو بيدي حسب عدد
Network
= 2² = 4

Example 18.9

Show the forwarding process if a packet arrives at R1 in Figure 18.33 with the destination address 180.70.65.140.

Solution

The router performs the following steps:

1. The first mask (26) is applied to the destination address. The result is 180.70.65.128, which does not match the corresponding network address.
2. The second mask (25) is applied to the destination address. The result is 180.70.65.128, which matches the corresponding network address. The next-hop address and the interface number m0 are extracted for forwarding the packet.

1000 1100
عدد البتات في الـ suffix
32 - 26 = 6
256 - 192 = 64
← (D) يجب

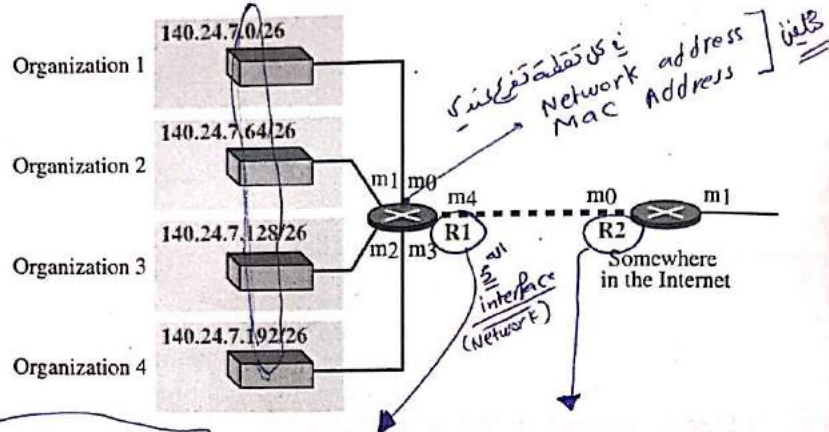
حل (2) bit
تقايقو

Address Aggregation

When we use classful addressing, there is only one entry in the forwarding table for each site outside the organization. The entry defines the site even if that site is subnetted. When a packet arrives at the router, the router checks the corresponding entry and forwards the packet accordingly. When we use classless addressing, it is likely that the number of forwarding table entries will increase. This is because the intent of classless addressing is to divide up the whole address space into manageable blocks. The increased size of the table results in an increase in the amount of time needed to search the table. To alleviate the problem, the idea of address aggregation was designed. In Figure 18.34 we have two routers.

R1 is connected to networks of four organizations that each use 64 addresses. R2 is somewhere far from R1. R1 has a longer forwarding table because each packet must be correctly routed to the appropriate organization. R2, on the other hand, can have a very small forwarding table. For R2, any packet with destination 140.24.7.0 to 140.24.7.255

Figure 18.34 Address aggregation



Forwarding table for R1

Network address/mask	Next-hop address	Interface
140.24.7.0/26	-----	m0
140.24.7.64/26	-----	m1
140.24.7.128/26	-----	m2
140.24.7.192/26	-----	m3
0.0.0.0/0	address of R2	m4

Forwarding table for R2

Network address/mask	Next-hop address	Interface
140.24.7.0/24	-----	m0
0.0.0.0/0	default router	m1

* هو نه لو طاربتكم عادي لان ال Prefix مساهي
 Default
 عادي لان ال Prefix مساهي
 كذا المقارنة

is sent out from interface m0 regardless of the organization number. This is called address aggregation because the blocks of addresses for four organizations are aggregated into one larger block. R2 would have a longer forwarding table if each organization had addresses that could not be aggregated into one block.

Longest Mask Matching

What happens if one of the organizations in the previous figure is not geographically close to the other three? For example, if organization 4 cannot be connected to router R1 for some reason, can we still use the idea of address aggregation and still assign block 140.24.7.192/26 to organization 4? The answer is yes, because routing in classless addressing uses another principle, **longest mask matching**. This principle states that the forwarding table is sorted from the longest mask to the shortest mask. In other words, if there are three masks, /27, /26, and /24, the mask /27 must be the first entry and /24 must be the last. Let us see if this principle solves the situation in which organization 4 is separated from the other three organizations. Figure 18.35 shows the situation.

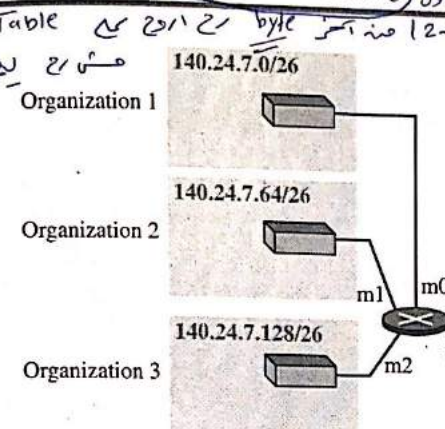
Suppose a packet arrives at router R2 for organization 4 with destination address 140.24.7.200. The first mask at router R2 is applied, which gives the network address 140.24.7.192. The packet is routed correctly from interface m1 and reaches organization 4. If, however, the forwarding table was not stored with the longest prefix first, applying the /24 mask would result in the incorrect routing of the packet to router R1.

مثال نوکانه کندی 140.24.7.200 / 26

Figure 18.35 Longest mask matching

دفعه و نسبت از length (mask) مرتبه ای address

م3 درجه فته
در default م3
Match



Forwarding table for R1

Network address/mask	Next-hop address	Interface
140.24.7.0/26	-----	m0
140.24.7.64/26	-----	m1
140.24.7.128/26	-----	m2
0.0.0.0/0	default router	m3

Forwarding table for R2

Network address/mask	Next-hop address	Interface
140.24.7.192/26	-----	m1
140.24.7.0/24	address of R1	m0
0.0.0.0/0	default router	m2

و البته از به خاطر مرتبه ای address در forwarding table

م0 م3 درجه فته

م0 م3 درجه فته

lock up Time

در 24 - همان

فقط

Hierarchical Routing

To solve the problem of gigantic forwarding tables, we can create a sense of hierarchy in the forwarding tables. In Chapter 2, we mentioned that the Internet today has a sense of hierarchy. We said that the Internet is divided into backbone and national ISPs. National ISPs are divided into regional ISPs, and regional ISPs are divided into local ISPs. If the forwarding table has a sense of hierarchy like the Internet architecture, the forwarding table can decrease in size.

Let us take the case of a local ISP. A local ISP can be assigned a single, but large, block of addresses with a certain prefix length. The local ISP can divide this block into smaller blocks of different sizes, and assign these to individual users and organizations, both large and small. If the block assigned to the local ISP starts with a.b.c.d/n, the ISP can create blocks starting with e.f.g.h/m, where m may vary for each customer and is greater than n.

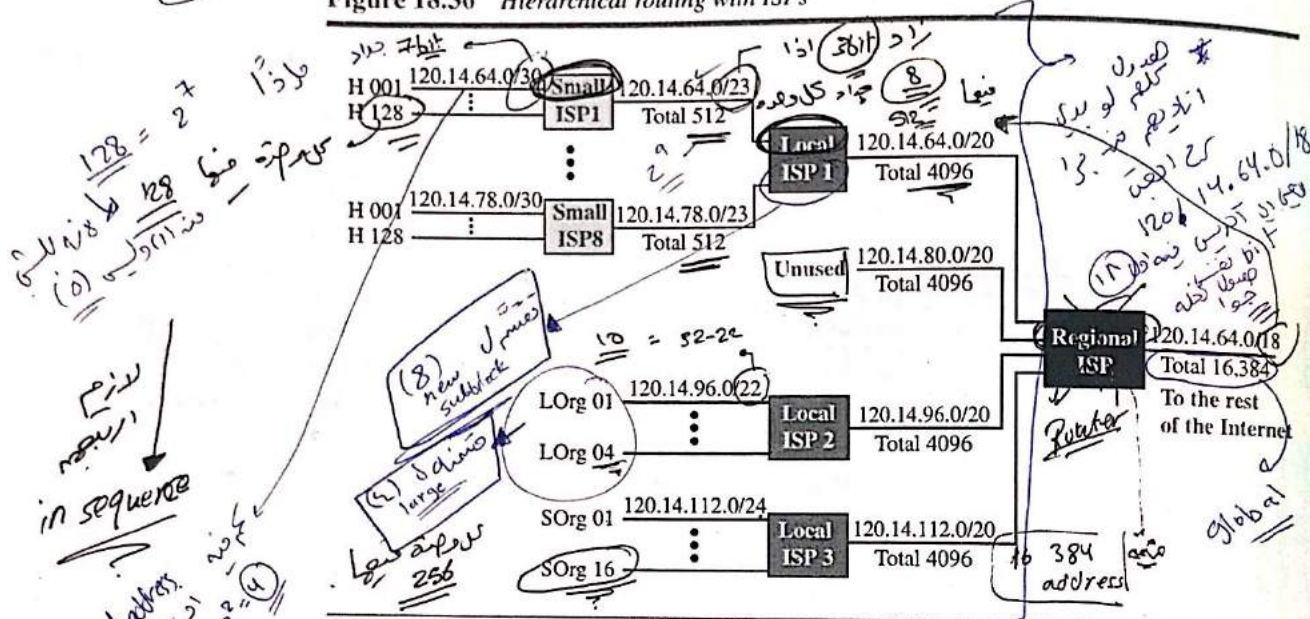
How does this reduce the size of the forwarding table? The rest of the Internet does not have to be aware of this division. All customers of the local ISP are defined as a.b.c.d/n to the rest of the Internet. Every packet destined for one of the addresses in this large block is routed to the local ISP. There is only one entry in every router in the world for all of these customers. They all belong to the same group. Of course, inside the local ISP, the router must recognize the subblocks and route the packet to the destined customer. If one of the customers is a large organization, it also can create another level of hierarchy by subnetting and dividing its subblock into smaller subblocks (or sub-subblocks). In classless routing, the levels of hierarchy are unlimited as long as we follow the rules of classless addressing.

64 → Binary 01.00000000
 01
 10
 11

Example 18.10

As an example of hierarchical routing, let us consider Figure 18.36. A regional ISP is granted 16,384 addresses starting from 120.14.64.0. The regional ISP has decided to divide this block into four subblocks, each with 4096 addresses. Three of these subblocks are assigned to three local ISPs; the second subblock is reserved for future use. Note that the mask for each block is /20 because the original block with mask /18 is divided into 4 blocks.

Figure 18.36 Hierarchical routing with ISPs



The first local ISP has divided its assigned subblock into 8 smaller blocks and assigned each to a small ISP. Each small ISP provides services to 128 households (H001 to H128), each using four addresses. Note that the mask for each small ISP is now /23 because the block is further divided into 8 blocks. Each household has a mask of /30, because a household has only 4 addresses ($2^{32-30} = 4$). The second local ISP has divided its block into 4 blocks and has assigned the addresses to 4 large organizations (LOrg01 to LOrg04). Note that each large organization has 1024 addresses and the mask is /22.

The third local ISP has divided its block into 16 blocks and assigned each block to a small organization (SOrg01 to SOrg16). Each small organization has 256 addresses and the mask is /24. There is a sense of hierarchy in this configuration. All routers in the Internet send a packet with destination address 120.14.64.0 to 120.14.127.255 to the regional ISP. The regional ISP sends every packet with destination address 120.14.64.0 to 120.14.79.255 to Local ISP1. Local ISP1 sends every packet with destination address 120.14.64.0 to 120.14.64.3 to H001.

Geographical Routing

To decrease the size of the forwarding table even further, we need to extend hierarchical routing to include geographical routing. We must divide the entire address space into a few large blocks. We assign a block to America, a block to Europe, a block to Asia, a block to Africa, and so on. The routers of ISPs outside of Europe will have only one entry for packets to Europe in their forwarding tables. The routers of ISPs outside of America will have only one entry for packets to America in their forwarding tables, and so on.

kind of aggregation

تقسيم المساحة الكلية الى كتل جغرافية
 block

32-13
 19

Forwarding Table Search Algorithms

In classless addressing, there is no network information in the destination address. The simplest, but not the most efficient, search method is called the longest prefix match (as we discussed before). The forwarding table can be divided into buckets, one for each prefix. The router first tries the longest prefix. If the destination address is found in this bucket, the search is complete. If the address is not found, the next prefix is searched, and so on. It is obvious that this type of search takes a long time.

One solution is to change the data structure used for searching. Instead of a list, other data structures (such as a tree or a binary tree) can be used. One candidate is a trie (a special kind of tree). However, this discussion is beyond the scope of this book.

18.5.2 Forwarding Based on Label

المشغلة الشبكية
Routing

علاوة على ذلك
Routing

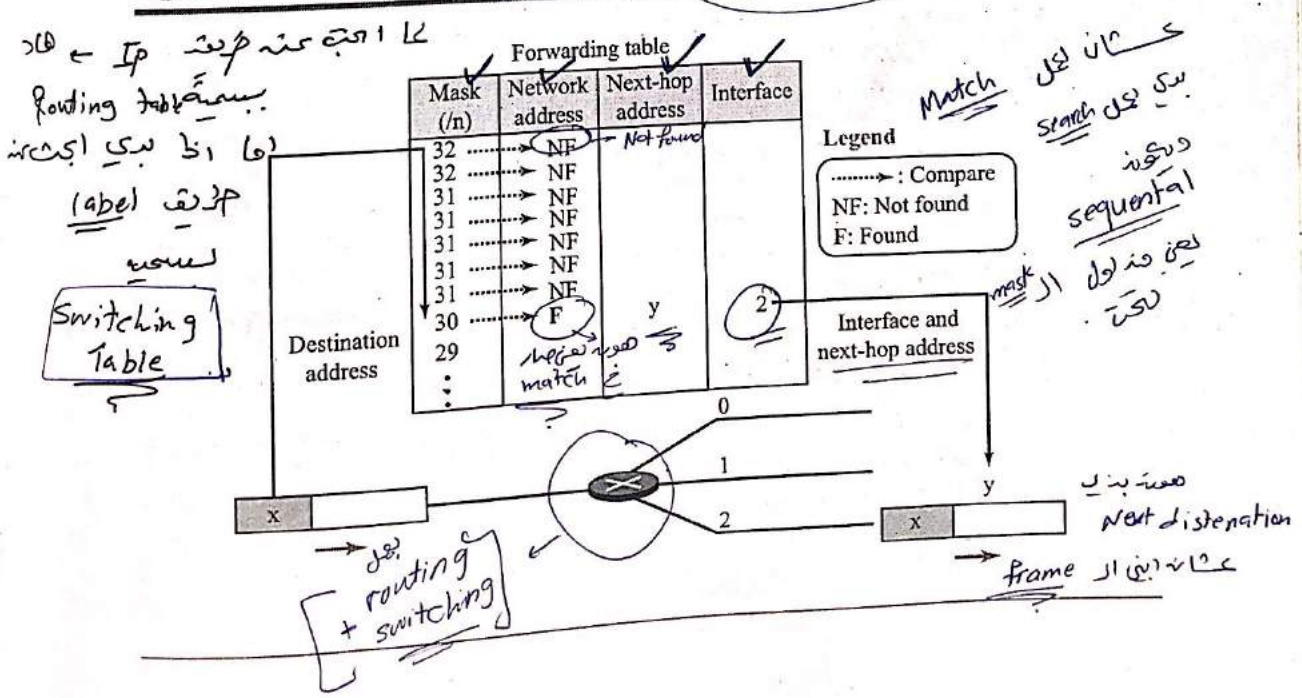
In the 1980s, an effort started to somehow change IP to behave like a connection-oriented protocol in which the routing is replaced by switching. As we discussed earlier in the chapter, in a connectionless network (datagram approach), a router forwards a packet based on the destination address in the header of the packet. On the other hand, in a connection-oriented network (virtual-circuit approach), a switch forwards a packet based on the label attached to the packet. Routing is normally based on searching the contents of a table; switching can be done by accessing a table using an index. In other words, routing involves searching; switching involves accessing.

بسر
اسم
switching

Example 18.11

Figure 18.37 shows a simple example of searching in a forwarding table using the longest mask algorithm. Although there are some more efficient algorithms today, the principle is the same.

Figure 18.37 Example 18.11: Forwarding based on destination address

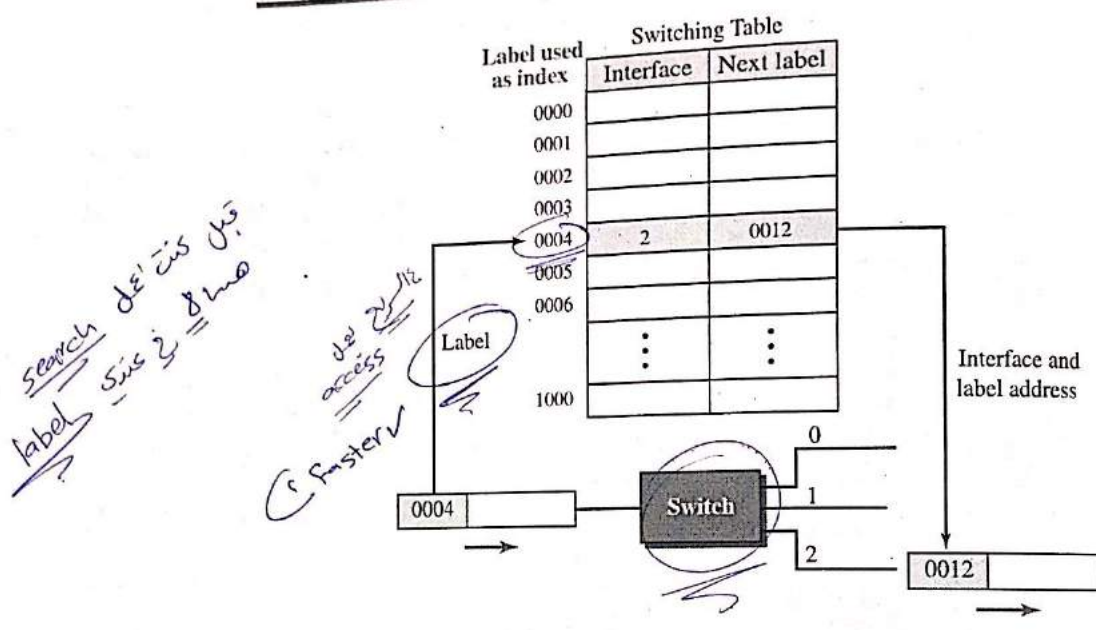


When the forwarding algorithm gets the destination address of the packet, it needs to delve into the mask column. For each entry, it needs to apply the mask to find the destination network address. It then needs to check the network addresses in the table until it finds the match. The router then extracts the next-hop address and the interface number to be delivered to the data-link layer.

Example 18.12

Figure 18.38 shows a simple example of using a label to access a switching table. Since the labels are used as the index to the table, finding the information in the table is immediate.

Figure 18.38 Example 18.12: Forwarding based on label



Multi-Protocol Label Switching (MPLS)

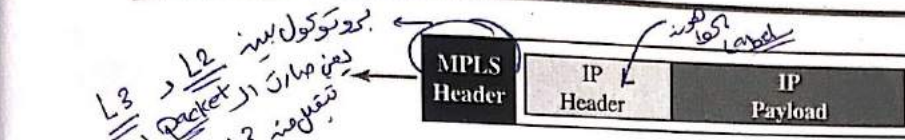
During the 1980s, several vendors created routers that implement switching technology. Later IETF approved a standard that is called Multi-Protocol Label Switching. In this standard, some conventional routers in the Internet can be replaced by MPLS routers, which can behave like a router and a switch. When behaving like a router, MPLS can forward the packet based on the destination address; when behaving like a switch, it can forward a packet based on the label.

A New Header

To simulate connection-oriented switching using a protocol like IP, the first thing that is needed is to add a field to the packet that carries the label discussed later. The IPv4 packet format does not allow this extension (although this field is provided in the IPv6 packet format, as we will see later). The solution is to encapsulate the IPv4 packet in an MPLS packet (as though MPLS were a layer between the data-link layer and the

network layer). The whole IP packet is encapsulated as the payload in an MPLS packet and an MPLS header is added. Figure 18.39 shows the encapsulation.

Figure 18.39 MPLS header added to an IP packet



The MPLS header is actually a stack of subheaders that is used for multilevel hierarchical switching, as we will discuss shortly. Figure 18.40 shows the format of an MPLS header in which each subheader is 32 bits (4 bytes) long.

Figure 18.40 MPLS header made of a stack of labels

0		20	24		31
Label	Exp	S	TTL		
Label	Exp	S	TTL		
...					
Label	Exp	S	TTL		

The following is a brief description of each field:

- ❑ **Label.** This 20-bit field defines the label that is used to index the forwarding table in the router.
- ❑ **Exp.** This 3-bit field is reserved for experimental purposes.
- ❑ **S.** The one-bit stack field defines the situation of the subheader in the stack. When the bit is 1, it means that the header is the last one in the stack.
- ❑ **TTL.** This 8-bit field is similar to the TTL field in the IP datagram. Each visited router decrements the value of this field. When it reaches zero, the packet is discarded to prevent looping.

Hierarchical Switching

A stack of labels in MPLS allows hierarchical switching. This is similar to conventional hierarchical routing. For example, a packet with two labels can use the top label to forward the packet through switches outside an organization; the bottom label can be used to route the packet inside the organization to reach the destination subnet.

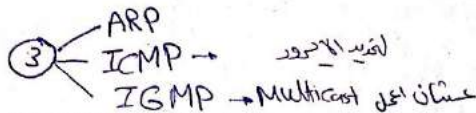
18.5.3 Routers as Packet Switches

As we may have guessed by now, the packet switches that are used in the network layer are called routers. Routers can be configured to act as either a datagram switch or a virtual-circuit switch. We have discussed the structure of a packet-switch in Chapter 8. The discussion in that chapter can be applied to any router used in the Internet.

CHAPTER 19

Network-Layer Protocols

① IP → un. Reliable. + connection less



* يعني في بعض الواجهات التي المفروض
توفرها هاد السروتيكون بس
مش توفرها.

In the previous chapter we introduced the network layer and discussed the services provided by this layer. We also discussed the logical addresses used in this layer. In this chapter, we show how the network layer is implemented in the TCP/IP protocol suite. The protocols in the network layer have gone through a few versions; in this chapter, we concentrate on the current version (v4). The next generation, which is on the horizon, is discussed in Chapter 22.

- The first section discusses the IPv4 protocol. It first describes the IPv4 datagram format. It then explains the purpose of fragmentation in a datagram. The section then briefly discusses options fields and their purpose in a datagram. The section finally mentions some security issues in IPv4, which are addressed in Chapter 32.
- The second section discusses ICMPv4, one of the auxiliary protocols used in the network layer to help IPv4. First, it briefly discusses the purpose of each option. The section then shows how ICMP can be used as a debugging tool. The section finally shows how the checksum is calculated for an ICMPv4 message.
- The third section discusses the mobile IP, whose use is increasing every day when people temporarily move their computers from one place to another. The section first describes the issue of address change in this situation. It then shows the three phases involved in the process. The section finally explains the inefficiency involved in this process and some solutions.

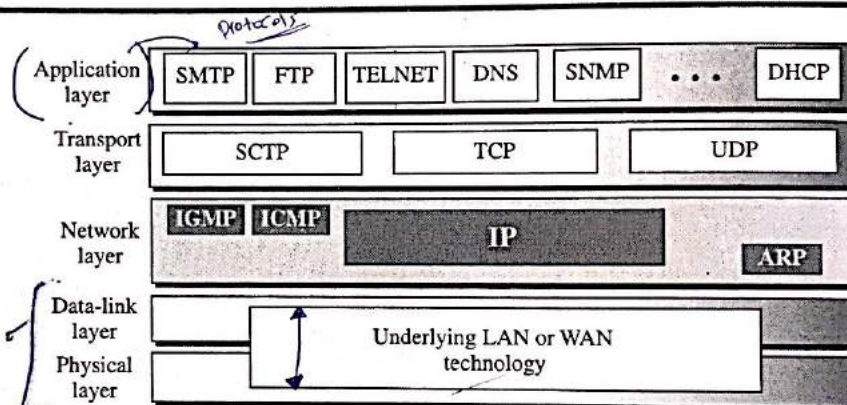
→ connection address

شکل ال (IP) متعلق بتوقع
الجغرافيا

19.1 INTERNET PROTOCOL (IP)

The network layer in version 4 can be thought of as one main protocol and three auxiliary ones. The main protocol, Internet Protocol version 4 (IPv4), is responsible for packetizing, forwarding, and delivery of a packet at the network layer. The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery. The Internet Group Management Protocol (IGMP) is used to help IPv4 in multicasting. The Address Resolution Protocol (ARP) is used to glue the network and data-link layers in mapping network-layer addresses to link-layer addresses. Figure 19.1 shows the positions of these four protocols in the TCP/IP protocol suite.

Figure 19.1 Position of IP and other network-layer protocols in TCP/IP protocol suite

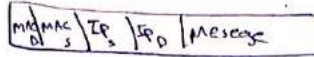


We will discuss IPv4 and ICMPv4 in this chapter. IGMP will be discussed when we talk about multicasting in Chapter 21. We have discussed ARP in Chapter 9 when we talked about link-layer addresses.

IPv4 is an unreliable datagram protocol—a best-effort delivery service. The term *best-effort* means that IPv4 packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network. If reliability is important, IPv4 must be paired with a reliable transport-layer protocol such as TCP. An example of a more commonly understood best-effort delivery service is the post office. The post office does its best to deliver the regular mail but does not always succeed. If an unregistered letter is lost or damaged, it is up to the sender or would-be recipient to discover this. The post office itself does not keep track of every letter and cannot notify a sender of loss or damage of one.

IPv4 is also a connectionless protocol that uses the datagram approach. This means that each datagram is handled independently, and each datagram can follow a different route to the destination. This implies that datagrams sent by the same source to the same destination could arrive out of order. Again, IPv4 relies on a higher-level protocol to take care of all these problems.

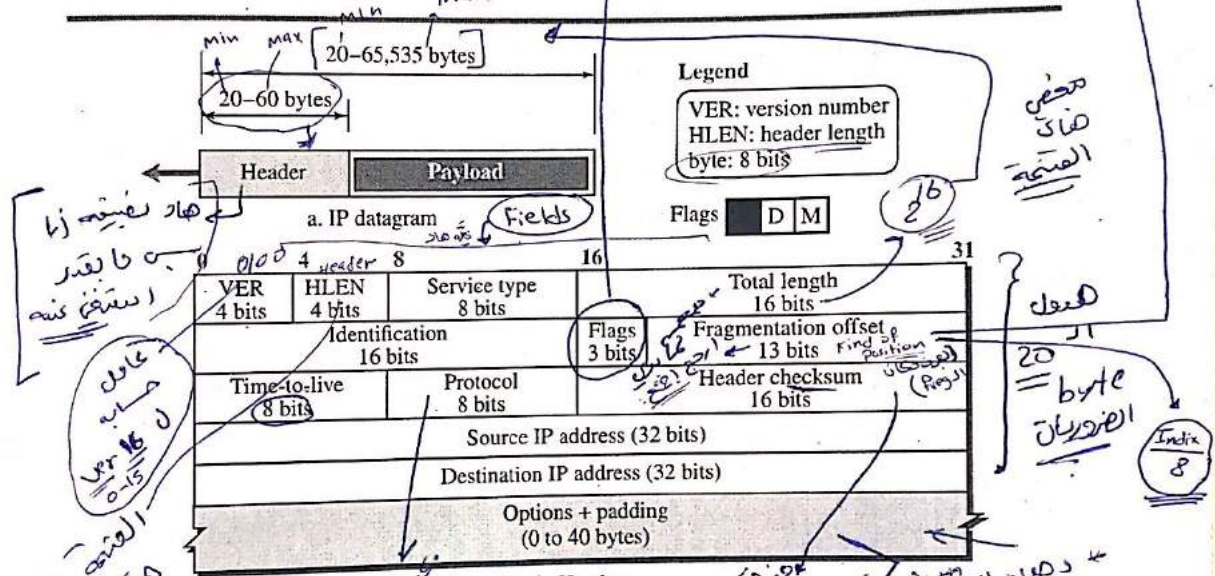
- 1 Min & Max Length
- 2 Field
- 3 Length of each Field



19.1.1 Datagram Format

In this section, we begin by discussing the first service provided by IPv4, packetizing. We show how IPv4 defines the format of a packet in which the data coming from the upper layer or other protocols are encapsulated. Packets used by the IP are called datagrams. Figure 19.2 shows the IPv4 datagram format. A datagram is a variable-length packet consisting of two parts: header and payload (data). The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections.

Figure 19.2 IP datagram



Discussing the meaning and rationale for the existence of each field is essential to understanding the operation of IPv4; a brief description of each field is in order.

- **Version Number.** The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.
- **Header Length.** The 4-bit header length (HLEN) field defines the total length of the datagram header in 4-byte words. The IPv4 datagram has a variable-length header. When a device receives a datagram, it needs to know when the header stops and the data, which is encapsulated in the packet, starts. However, to make the value of the header length (number of bytes) fit in a 4-bit header length, the total length of the header is calculated as 4-byte words. The total length is divided by 4 and the value is inserted in the field. The receiver needs to multiply the value of this field by 4 to find the total length.
- **Service Type.** In the original design of the IP header, this field was referred to as type of service (TOS), which defined how the datagram should be handled. In the late 1990s, IETF redefined the field to provide *differentiated services* (DiffServ).

When we discuss differentiated services in Chapter 30, we will be in a better situation to define the bits in this field. The use of 4-byte words for the length header is also logical because the IP header always needs to be aligned in 4-byte boundaries.

- **Total Length.** This 16-bit field defines the total length (header plus data) of the IP datagram in bytes. A 16-bit number can define a total length of up to 65,535 (when all bits are 1s). However, the size of the datagram is normally much less than this. This field helps the receiving device to know when the packet has completely arrived. To find the length of the data coming from the upper layer, subtract the header length from the total length. The header length can be found by multiplying the value in the HLEN field by 4.

$$\text{Length of data} = \text{total length} - (\text{HLEN}) \times 4$$

Though a size of 65,535 bytes might seem large, the size of the IPv4 datagram may increase in the near future as the underlying technologies allow even more throughput (greater bandwidth).

One may ask why we need this field anyway. When a machine (router or host) receives a frame, it drops the header and the trailer, leaving the datagram. Why include an extra field that is not needed? The answer is that in many cases we really do not need the value in this field. However, there are occasions in which the datagram is not the only thing encapsulated in a frame; it may be that padding has been added. For example, the Ethernet protocol has a minimum and maximum restriction on the size of data that can be encapsulated in a frame (46 to 1500 bytes). If the size of an IPv4 datagram is less than 46 bytes, some padding will be added to meet this requirement. In this case, when a machine decapsulates the datagram, it needs to check the total length field to determine how much is really data and how much is padding.

- **Identification, Flags, and Fragmentation Offset.** These three fields are related to the fragmentation of the IP datagram when the size of the datagram is larger than the underlying network can carry. We discuss the contents and importance of these fields when we talk about fragmentation in the next section.

- **Time-to-live.** Due to some malfunctioning of routing protocols (discussed later) a datagram may be circulating in the Internet, visiting some networks over and over without reaching the destination. This may create extra traffic in the Internet. The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately two-times the maximum number of routers between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.

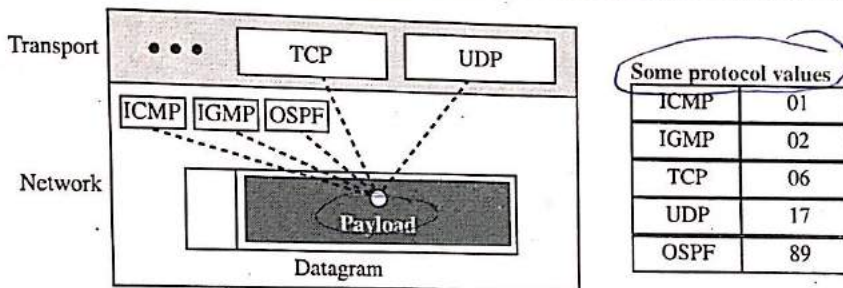
- **Protocol.** In TCP/IP, the data section of a packet, called the *payload*, carries the whole packet from another protocol. A datagram, for example, can carry a packet belonging to any transport-layer protocol such as UDP or TCP. A datagram can also carry a packet from other protocols that directly use the service of the IP, such as some routing protocols or some auxiliary protocols. The Internet authority has given

شكل بيكي ايهن صبح Source الي Destination وديه قسمة 15
 فانا بجزله بـ (Time to live) زيده
 (20) يعني space زيده
 وكل ما توصل الميعاد راوتر بعلما
 Processing على dicramble بيده
 1
 اذا ما رات صفة صال 0
 Zero فعلا كل لعلي الميعاد
 dis count

decapsulation → encapsulation
 استرجاع البيانات

any protocol that uses the service of IP a unique 8-bit number which is inserted in the protocol field. When the payload is encapsulated in a datagram at the source IP, the corresponding protocol number is inserted in this field; when the datagram arrives at the destination, the value of this field helps to define to which protocol the payload should be delivered. In other words, this field provides multiplexing at the source and demultiplexing at the destination, as shown in Figure 19.3. Note that the protocol fields at the network layer play the same role as the port numbers at the transport layer (Chapters 23 and 24). However, we need two port numbers in a transport-layer packet because the port numbers at the source and destination are different, but we need only one protocol field because this value is the same for each protocol no matter whether it is located at the source or the destination.

Figure 19.3 Multiplexing and demultiplexing using the value of the protocol field



- Header checksum.** IP is not a reliable protocol; it does not check whether the payload carried by a datagram is corrupted during the transmission. IP puts the burden of error checking of the payload on the protocol that owns the payload, such as UDP or TCP. The datagram header, however, is added by IP, and its error-checking is the responsibility of IP. Errors in the IP header can be a disaster. For example, if the destination IP address is corrupted, the packet can be delivered to the wrong host. If the protocol field is corrupted, the payload may be delivered to the wrong protocol. If the fields related to the fragmentation are corrupted, the datagram cannot be reassembled correctly at the destination, and so on. For these reasons, IP adds a header checksum field to check the header, but not the payload. We need to remember that, since the value of some fields, such as TTL, which are related to fragmentation and options, may change from router to router, the checksum needs to be recalculated at each router. As we discussed in Chapter 10, checksum in the Internet normally uses a 16-bit field, which is the complement of the sum of other fields calculated using 1s complement arithmetic.

- Source and Destination Addresses.** These 32-bit source and destination address fields define the IP address of the source and destination respectively. The source host should know its IP address. The destination IP address is either known by the protocol that uses the service of IP or is provided by the DNS as described in Chapter 26. Note that the value of these fields must remain unchanged during the

node ke
 processing
 ke liye
 checksum
 field

time the IP datagram travels from the source host to the destination host. IP addresses were discussed in Chapter 18.

- **Options.** A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software. This means that all implementations must be able to handle options if they are present in the header. The existence of options in a header creates some burden on the datagram handling; some options can be changed by routers, which forces each router to recalculate the header checksum. There are one-byte and multi-byte options that we will briefly discuss later in the chapter. The complete discussion is posted at the book website.
- **Payload.** Payload, or data, is the main reason for creating a datagram. Payload is the packet coming from other protocols that use the service of IP. Comparing a datagram to a postal package, payload is the content of the package; the header is only the information written on the package.

Example 19.1

An IPv4 packet has arrived with the first 8 bits as $(01000010)_2$. The receiver discards the packet. Why?

Solution

There is an error in this packet. The 4 leftmost bits $(0100)_2$ show the version, which is correct. The next 4 bits $(0010)_2$ show an invalid header length ($2 \times 4 = 8$). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Example 19.2

In an IPv4 packet, the value of HLEN is $(1000)_2$. How many bytes of options are being carried by this packet?

Solution

The HLEN value is 8, which means the total number of bytes in the header is 8×4 , or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

Example 19.3

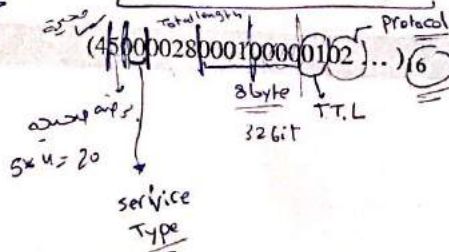
In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is $(0028)_{16}$. How many bytes of data are being carried by this packet?

Solution

The HLEN value is 5, which means the total number of bytes in the header is 5×4 , or 20 bytes (no options). The total length is $(0028)_{16}$ or 40 bytes, which means the packet is carrying 20 bytes of data ($40 - 20$).

Example 19.4

An IPv4 packet has arrived with the first few hexadecimal digits as shown.



Handwritten notes for Example 19.1:

- نصفه (2) طرز اجزا
- ن (4) = 8
- min 20
- ن 15

Handwritten notes for Example 19.3:

- 15
- 5
- 12 option

Handwritten notes for Example 19.4:

- 21
- 10
- 8
- 4
- 2
- 32
- 16
- 8
- 2
- 32 + 8 = 40

How many hops can this packet travel before being dropped? The data belong to what upper-layer protocol?

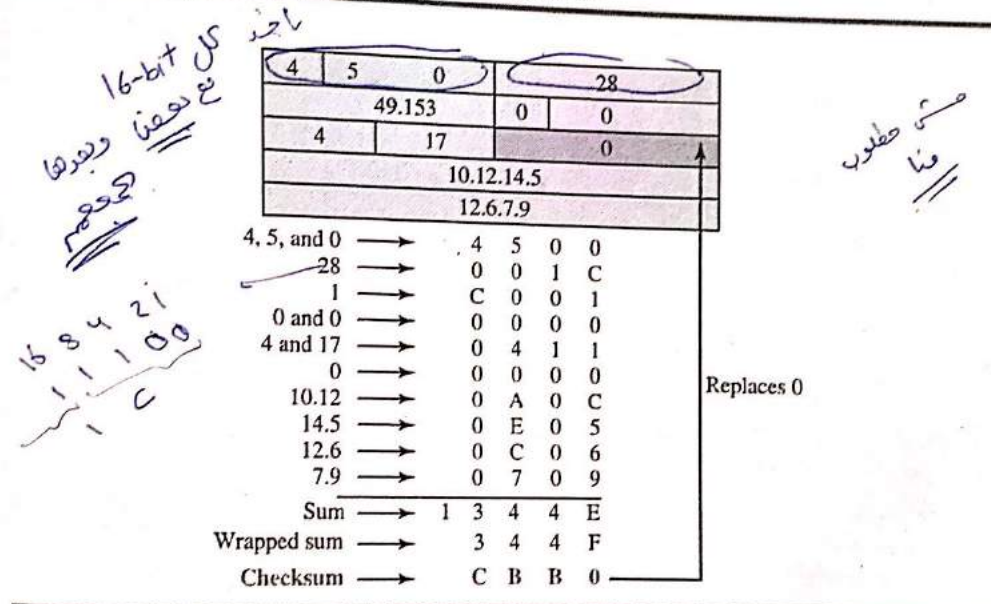
Solution

To find the time-to-live field, we skip 8 bytes (16 hexadecimal digits). The time-to-live field is the ninth byte which is (01)₁₆. This means the packet can travel only one hop. The protocol field is the next byte (02)₁₆, which means that the upper-layer protocol is IGMP.

Example 19.5

Figure 19.4 shows an example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented after wrapping the leftmost digit. The result is inserted in the checksum field.

Figure 19.4 Example of checksum calculation in IPv4



Note that the calculation of wrapped sum and checksum can also be done as follows in hexadecimal:

Wrapped Sum = Sum mod FFFF

Checksum = FFFF - Wrapped Sum

19.1.2 Fragmentation

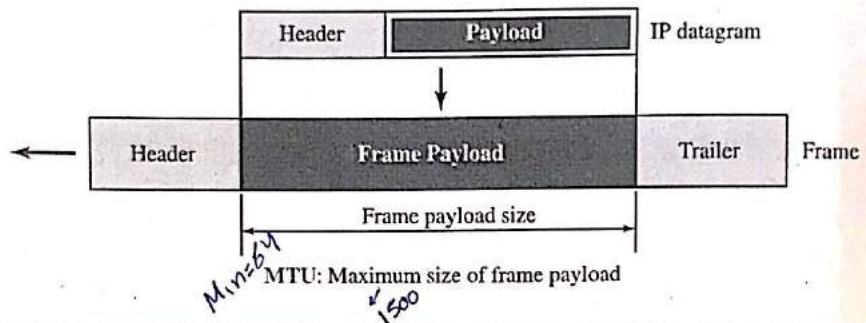
A datagram can travel through different networks. Each router decapsulates the IP datagram from the frame it receives, processes it, and then encapsulates it in another frame. The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled. The format and size of the sent frame depend on the protocol used by the physical-network through which the frame is going to travel. For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

اگر size یعنی ٹوٹنے کا IP پروٹوکول سے زیادہ ہو

Maximum Transfer Unit (MTU)

Each link-layer protocol has its own frame format. One of the features of each format is the maximum size of the payload that can be encapsulated. In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network (see Figure 19.5).

Figure 19.5 Maximum transfer unit (MTU)



The value of the MTU differs from one physical network protocol to another. For example, the value for a LAN is normally 1500 bytes, but for a WAN it can be larger or smaller.

انہی کے لئے کی جانے والی Length Maximum length of the IP = 65,535

In order to make the IP protocol independent of the physical network, the designers decided to make the maximum length of the IP datagram equal to 65,535 bytes. This makes transmission more efficient if one day we use a link-layer protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible for it to pass through these networks. This is called fragmentation.

When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some have been changed. A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram may be fragmented several times before it reaches the final destination.

Source اور ڈسٹنیشن

A datagram can be fragmented by the source host or any router in the path. The reassembly of the datagram, however, is done only by the destination host, because each fragment becomes an independent datagram. Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all of the fragments belonging to the same datagram should finally arrive at the destination host. So it is logical to do the reassembly at the final destination. An even stronger objection for reassembling packets during the transmission is the loss of efficiency it incurs.

header

When we talk about fragmentation, we mean that the payload of the IP datagram is fragmented. However, most parts of the header, with the exception of some options, must be copied by all fragments. The host or router that fragments a datagram must change the values of three fields: flags, fragmentation offset, and total length. The rest

Handwritten notes and arrows pointing to the fields mentioned in the text: flags, fragmentation offset, and total length.

of the fields must be copied. Of course, the value of the checksum must be recalculated regardless of fragmentation.

Fields Related to Fragmentation

We mentioned before that three fields in an IP datagram are related to fragmentation: identification, flags, and fragmentation offset. Let us explain these fields now.

The 16-bit identification field identifies a datagram originating from the source host. The combination of the identification and source IP address must uniquely define a datagram as it leaves the source host. To guarantee uniqueness, the IP protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IP protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by one. As long as the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied into all fragments. In other words, all fragments have the same identification number, which is also the same as the original datagram. The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value should be assembled into one datagram.

خانه
Ident

The 3-bit flags field defines three flags. The leftmost bit is reserved (not used). The second bit (D bit) is called the do not fragment bit. If its value is 1, the machine must not fragment the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host (discussed later). If its value is 0, the datagram can be fragmented if necessary. The third bit (M bit) is called the more fragment bit. If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment.

بنا
پرو
لا
151-220-011
First frag
لا

The 13-bit fragmentation offset field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes. Figure 19.6 shows a datagram with a data size of 4000 bytes fragmented into three fragments. The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is $0/8 = 0$. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is $1400/8 = 175$. Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is $2800/8 = 350$.

index of the first byte

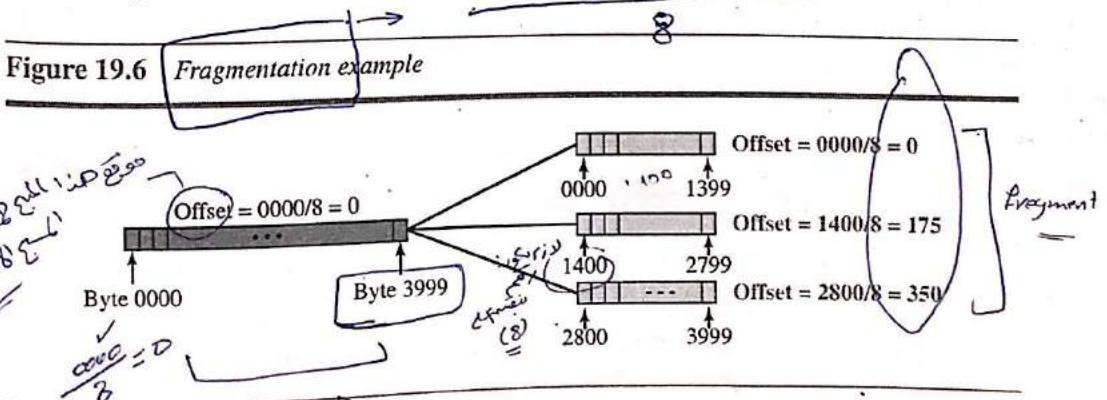


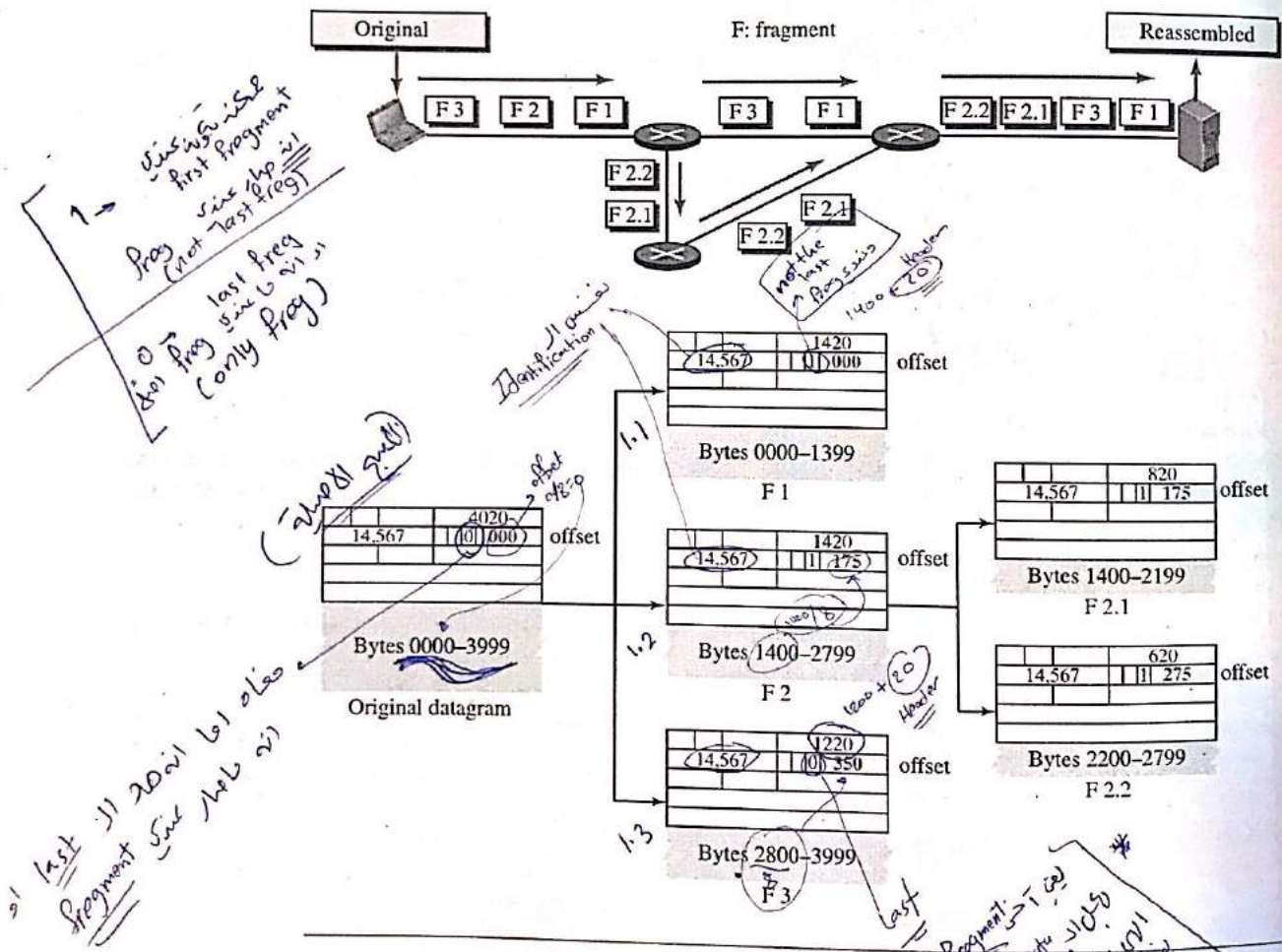
Figure 19.6 Fragmentation example

Datagram = 4000 Byte + 20 byte header

Remember that the value of the offset is measured in units of 8 bytes. This is done because the length of the offset field is only 13 bits long and cannot represent a sequence of bytes greater than 8191. This forces hosts or routers that fragment datagrams to choose the size of each fragment so that the first byte number is divisible by 8.

Figure 19.7 shows an expanded view of the fragments in the previous figure. The original packet starts at the client; the fragments are reassembled at the server. The value of the identification field is the same in all fragments, as is the value of the flags field with the more bit set for all fragments except the last. Also, the value of the offset field for each fragment is shown. Note that although the fragments arrived out of order at the destination, they can be correctly reassembled.

Figure 19.7 Detailed fragmentation example



The figure also shows what happens if a fragment itself is fragmented. In this case the value of the offset field is always relative to the original datagram. For example, in the figure, the second fragment is itself fragmented later into two fragments of

800 bytes and 600 bytes, but the offset shows the relative position of the fragments to the original data.

It is obvious that even if each fragment follows a different path and arrives out of order, the final destination host can reassemble the original datagram from the fragments received (if none of them is lost) using the following strategy:

- The first fragment has an offset field value of zero.
- Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.
- Divide the total length of the first and second fragment by 8. The third fragment has an offset value equal to that result.
- Continue the process. The last fragment has its M bit set to 0.
- Continue the process. The last fragment has a more bit value of 0.

Example 19.6

A packet has arrived with an M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

Example 19.7

A packet has arrived with an M bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset).

Example 19.8

A packet has arrived with an M bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, or a middle fragment?

Solution

Because the M bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 19.9

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

Example 19.10

A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the total length field is 100. What are the numbers of the first byte and the last byte?

Solution

The first byte number is $100 \times 8 = 800$. The total length is 100 bytes, and the header length is 20 bytes (5×4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

19.1.3 (Options)

The header of the IPv4 datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises the options that can be a maximum of 40 bytes (in multiples of 4-bytes) to preserve the boundary of the header.

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IPv4 header, option processing is required of the IPv4 software. This means that all implementations must be able to handle options if they are present in the header. Options are divided into two broad categories: single-byte options and multiple-byte options. We give a brief description of options here; for a complete description, see the book website under Extra Materials.

The complete discussion of options in IPv4 is included in the book website under Extra Materials for Chapter 19.

Single-Byte Options

There are two single-byte options.

No Operation

A *no-operation option* is a 1-byte option used as a filler between options.

End of Option

An *end-of-option option* is a 1-byte option used for padding at the end of the option field. It, however, can only be used as the last option.

Multiple-Byte Options

There are four multiple-byte options.

Record Route

A *record route option* is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.

Strict Source Route

A *strict source route option* is used by the source to predetermine a route for the datagram as it travels through the Internet. Dictation of a route by the source can be useful for several purposes. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput. Alternatively, it may choose a route that is

safer or more reliable for the sender's purpose. For example, a sender can choose a route so that its datagram does not travel through a competitor's network.

If a datagram specifies a strict source route, all the routers defined in the option must be visited by the datagram. A router must not be visited if its IPv4 address is not listed in the datagram. If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued. If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded and an error message issued.

Loose Source Route

A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

Timestamp

A timestamp option is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal time or Greenwich mean time. Knowing the time a datagram is processed can help users and managers track the behavior of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another. We say estimate because, although all routers may use Universal time, their local clocks may not be synchronized.

19.1.4 Security of IPv4 Datagrams

The IPv4 protocol, as well as the whole Internet, was started when the Internet users trusted each other. No security was provided for the IPv4 protocol. Today, however, the situation is different; the Internet is not secure anymore. Although we will discuss network security in general and IP security in particular in Chapters 31 and 32, here we give a brief idea about the security issues in IP protocol and the solutions. There are three security issues that are particularly applicable to the IP protocol: packet sniffing, packet modification, and IP spoofing.

Packet Sniffing → (passive attack)

An intruder may intercept an IP packet and make a copy of it. Packet sniffing is a passive attack, in which the attacker does not change the contents of the packet. This type of attack is very difficult to detect because the sender and the receiver may never know that the packet has been copied. Although packet sniffing cannot be stopped, encryption of the packet can make the attacker's effort useless. The attacker may still sniff the packet, but the content is not detectable.

Packet Modification

The second type of attack is to modify the packet. The attacker intercepts the packet, changes its contents, and sends the new packet to the receiver. The receiver believes that the packet is coming from the original sender. This type of attack can be detected using a data integrity mechanism. The receiver, before opening and using the contents of the message, can use this mechanism to make sure that the packet has not been changed during the transmission. We discuss packet integrity in Chapter 32.

تجسس یعنی
صدا فشرقی کردن
ال connection
محل انحصار برقرار
حالی تمام با یکدیگر
و دیگر فشرقی کردن
دقیقتر دیدن سوزن
باشیم

اطلاعات را کدگذاری
(decryption)
(encryption)

گوشه فقط یعنی بدون تغییر

* یعنی صدا فشرقی های ال connection و محل ارسال [modification] صدا بی هیچ

ان هذا
يشتمل
على
IP spoofing

IP Spoofing

An attacker can masquerade as somebody else and create an IP packet that carries the source address of another computer. An attacker can send an IP packet to a bank pretending that it is coming from one of the customers. This type of attack can be prevented using a origin authentication mechanism (see Chapter 32).

نحري
طوبى
للحق

IPSec

The IP packets today can be protected from the previously mentioned attacks using a protocol called IPSec (IP Security). This protocol, which is used in conjunction with the IP protocol creates a connection-oriented service between two entities in which they can exchange IP packets without worrying about the three attacks discussed above. We will discuss IPSec in detail in Chapter 32; here it is enough to mention that IPSec provides the following four services:

- ❑ **Defining Algorithms and Keys.** The two entities that want to create a secure channel between themselves can agree on some available algorithms and keys to be used for security purposes.
- ❑ **Packet Encryption.** The packets exchanged between two parties can be encrypted for privacy using one of the encryption algorithms and a shared key agreed upon in the first step. This makes the packet sniffing attack useless.
- ❑ **Data Integrity.** Data integrity guarantees that the packet is not modified during the transmission. If the received packet does not pass the data integrity test, it is discarded. This prevents the second attack, packet modification, described above.
- ❑ **Origin Authentication.** IPSec can authenticate the origin of the packet to be sure that the packet is not created by an imposter. This can prevent IP spoofing attacks as described above.

19.2 ICMPv4

The IPv4 has no error-reporting or error-correcting mechanism. What happens if something goes wrong? What happens if a router must discard a datagram because it cannot find a route to the final destination, or because the time-to-live field has a zero value? What happens if the final destination host must discard the received fragments of a datagram because it has not received all fragments within a predetermined time limit? These are examples of situations where an error has occurred and the IP protocol has no built-in mechanism to notify the original host.

The IP protocol also lacks a mechanism for host and management queries. A host sometimes needs to determine if a router or another host is alive. And sometimes a network manager needs information from another host or router.

The Internet Control Message Protocol version 4 (ICMPv4) has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol. ICMP itself is a network-layer protocol. However, its messages are not passed directly to the data-link layer as would be expected. Instead, the messages are first encapsulated inside IP datagrams before going to the lower layer. When an IP datagram encapsulates

طوبى
ببعض
الاصناف
التي
error control

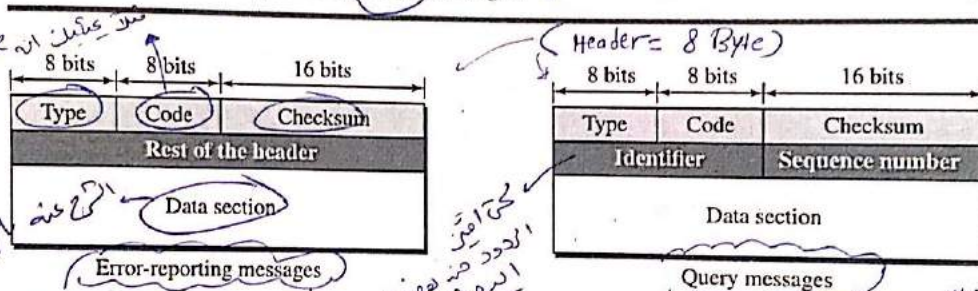
an ICMP message, the value of the protocol field in the IP datagram is set to 1 to indicate that the IP payload is an ICMP message.

19.2.1 MESSAGES

ICMP messages are divided into two broad categories: *error-reporting messages* and *query messages*. The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet. The query messages, which occur in pairs, help a host or a network manager get specific information from a router or another host. For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network and routers can help a node redirect its messages.

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all. As Figure 19.8 shows, the first field, ICMP type, defines the type of the message. The code field specifies the reason for the particular message type. The last common field is the checksum field (to be discussed later in the chapter). The rest of the header is specific for each message type.

Figure 19.8 General format of ICMP messages



Type and code values

Error-reporting messages	Query messages
03: Destination unreachable (codes 0 to 15)	08 and 09: Echo request and reply (only code 0)
04: Source quench (only code 0)	13 and 14: Timestamp request and reply (only code 0)
05: Redirection (codes 0 to 3)	
11: Time exceeded (codes 0 and 1)	
12: Parameter problem (codes 0 and 1)	

The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of query.

We give a brief description of the ICMPv4 messages here; for a complete description see the book website under Extra Materials for Chapter 19.

The complete discussion of messages in ICMPv4 is included in the book website under Extra Materials for Chapter 19.

بعض استفسارات
تعميم
وتقسيم
البيانات

ايضا في الخبيرة ال error

Type + code
تقسيم
بشكل
ICMP message

من الممكن ان يكون
8

البيانات
577

بشكل
البيانات

نقطة
من رسالة

Redirect
بالتالي

من الممكن ان يكون
تقسيم
بشكل
من البيانات
لا يمكن ان يكون
بشكل
(192.168.1.1)

بشكل
Request
Reply

Data section

Error Reporting Messages

Since IP is an unreliable protocol, one of the main responsibilities of ICMP is to report some errors that may occur during the processing of the IP datagram. ICMP does not correct errors, it simply reports them. Error correction is left to the higher-level protocols. Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses. ICMP uses the source IP address to send the error message to the source (originator) of the datagram. To make the error-reporting process simple, ICMP follows some rules in reporting messages. First, no error message will be generated for a datagram having a multicast address or special address (such as *this host* or *loopback*). Second, no ICMP error message will be generated in response to a datagram carrying an ICMP error message. Third, no ICMP error message will be generated for a fragmented datagram that is not the first fragment.

Note that all error messages contain a data section that includes the IP header of the original datagram plus the first 8 bytes of data in that datagram. The original datagram header is added to give the original source, which receives the error message, information about the datagram itself. The 8 bytes of data are included because the first 8 bytes provide information about the port numbers (UDP and TCP) and sequence number (TCP). This information is needed so the source can inform the protocols (TCP or UDP) about the error.

The following are important points about ICMP error messages:

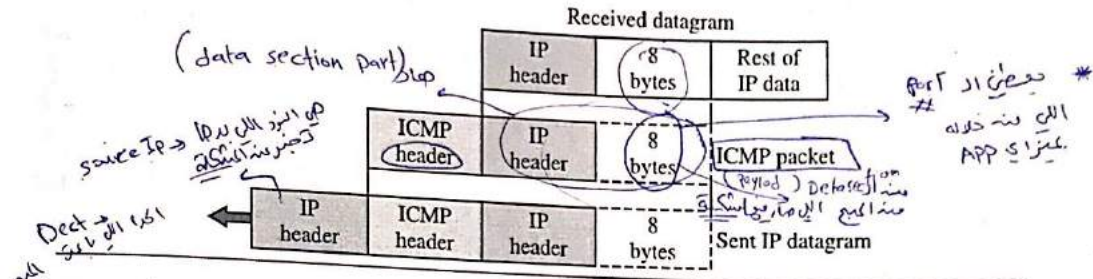
- No ICMP error message will be generated in response to a datagram carrying an ICMP error message.
- No ICMP error message will be generated for a fragmented datagram that is not the first fragment.
- No ICMP error message will be generated for a datagram having a multicast address.
- No ICMP error message will be generated for a datagram having a special address such as 127.0.0.0 or 0.0.0.0.

Note that all error messages contain a data section that includes the IP header of the original datagram plus the first 8 bytes of data in that datagram. The original datagram header is added to give the original source, which receives the error message, information about the datagram itself. The 8 bytes of data are included because, as we will see in Chapter 24 on UDP and TCP protocols, the first 8 bytes provide information about the port numbers (UDP and TCP) and sequence number (TCP). This information is needed so the source can inform the protocols (TCP or UDP) about the error. ICMP forms an error packet, which is then encapsulated in an IP datagram (see Figure 19.9).

Destination Unreachable

The most widely used error message is the destination unreachable (type 3). This message uses different codes (0 to 15) to define the type of error message and the reason why a datagram has not reached its final destination. For example, code 0 tells the

Figure 19.9 Contents of data field for the error messages



source that a host is unreachable. This may happen, for example, when we use the HTTP protocol to access a web page, but the server is down. The message "destination host is not reachable" is created and sent back to the source.

Source Quench

Another error message is called the *source quench* (type 4) message, which informs the sender that the network has encountered congestion and the datagram has been dropped; the source needs to slow down sending more datagrams. In other words, ICMP adds a kind of congestion control mechanism to the IP protocol by using this type of message.

Redirection Message

The *redirection message* (type 5) is used when the source uses a wrong router to send out its message. The router redirects the message to the appropriate router, but informs the source that it needs to change its default router in the future. The IP address of the default router is sent in the message.

We discussed the purpose of the *time-to-live* (TTL) field in the IP datagram and explained that it prevents a datagram from being aimlessly circulated in the Internet. When the TTL value becomes 0, the datagram is dropped by the visiting router and a *time exceeded* message (type 11) with code 0 is sent to the source to inform it about the situation. The time-exceeded message (with code 1) can also be sent when not all fragments of a datagram arrive within a predefined period of time.

Parameter Problem

A *parameter problem message* (type 12) can be sent when either there is a problem in the header of a datagram (code 0) or some options are missing or cannot be interpreted (code 1).

Query Messages

Interestingly, query messages in ICMP can be used independently without relation to an IP datagram. Of course, a query message needs to be encapsulated in a datagram, as a carrier. Query messages are used to probe or test the liveliness of hosts or routers in the Internet, find the one-way or the round-trip time for an IP datagram between two devices, or even find out whether the clocks in two devices are synchronized. Naturally, query messages come in pairs: request and reply.

The *echo request* (type 8) and the *echo reply* (type 0) pair of messages are used by a host or a router to test the liveliness of another host or router. A host or router sends

an echo request message to another host or router; if the latter is alive, it responds with an echo reply message. We shortly see the applications of this pair in two debugging tools: *ping* and *traceroute*.

The *timestamp request* (type 13) and the *timestamp reply* (type 14) pair of messages are used to find the round-trip time between two devices or to check whether the clocks in two devices are synchronized. The timestamp request message sends a 32-bit number, which defines the time the message is sent. The timestamp reply resends that number, but also includes two new 32-bit numbers representing the time the request was received and the time the response was sent. If all timestamps represent Universal time, the sender can calculate the one-way and round-trip time.

Deprecated Messages

Three pairs of messages are declared obsolete by IETF:

1. *Information request and replay* messages are not used today because their duties are done by the Address Resolution Protocol (ARP) discussed in Chapter 9.
2. *Address mask request and reply* messages are not used today because their duties are done by the Dynamic Host Configuration Protocol (DHCP), discussed in Chapter 18.
3. *Router solicitation and advertisement* messages are not used today because their duties are done by the Dynamic Host Configuration Protocol (DHCP), discussed in Chapter 18.

19.2.2 Debugging Tools :-

There are several tools that can be used in the Internet for debugging. We can determine the viability of a host or router. We can trace the route of a packet. We introduce two tools that use ICMP for debugging: *ping* and *traceroute*.

Ping

We can use the *ping* program to find if a host is alive and responding. We use *ping* here to see how it uses ICMP packets. The source host sends ICMP echo-request messages; the destination, if alive, responds with ICMP echo-reply messages. The *ping* program sets the identifier field in the echo-request and echo-reply message and starts the sequence number from 0; this number is incremented by 1 each time a new message is sent. Note that *ping* can calculate the round-trip time. It inserts the sending time in the data section of the message. When the packet arrives, it subtracts the arrival time from the departure time to get the round-trip time (RTT).

Example 19.11

The following shows how we send a *ping* message to the *auniversity.edu* site. We set the identifier field in the echo request and reply message and start the sequence number from 0; this number is incremented by one each time a new message is sent. Note that *ping* can calculate the round-trip time. It inserts the sending time in the data section of the message. When the packet arrives, it subtracts the arrival time from the departure time to get the *round-trip time* (rtt).

```
$ ping auniversity.edu
```

```
PING auniversity.edu (152.181.8.3) 56 (84) bytes of data:
```

```
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=0
```

```
ttl=62
```

```
time=1.91 ms
```

اسم ہوسٹ اور
IP

```

64 bytes from auniversity.edu (152.181.8.3): icmp_seq=1    ttl=62    time=2.04 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=2    ttl=62    time=1.90 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=3    ttl=62    time=1.97 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=4    ttl=62    time=1.93 ms
--- auniversity.edu statistics ---
6 packets transmitted, 6 received, 0% packet loss
rtt min/avg/max = 1.90/1.95/2.04 ms

```

Traceroute or Tracert

The *traceroute* program in UNIX or *tracert* in Windows can be used to trace the path of a packet from a source to the destination. It can find the IP addresses of all the routers that are visited along the path. The program is usually set to check for the maximum of 30 hops (routers) to be visited. The number of hops in the Internet is normally less than this. Since these two programs behave differently in Unix and Windows, we explain them separately.

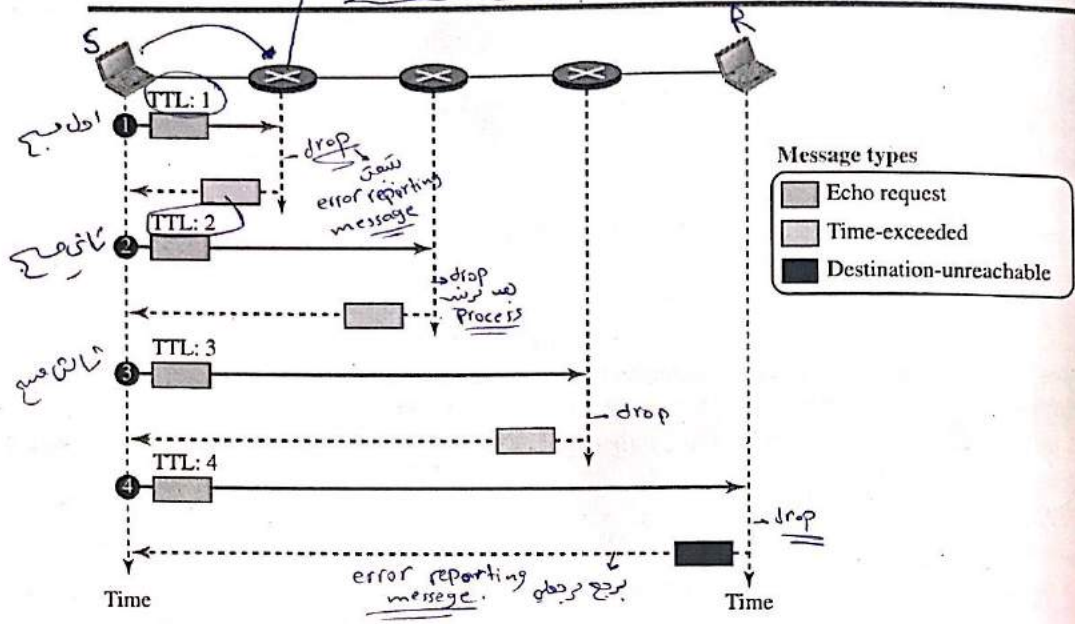
Traceroute

The *traceroute* program is different from the *ping* program. The *ping* program gets help from two query messages; the *traceroute* program gets help from two error-reporting messages: time-exceeded and destination-unreachable. The *traceroute* is an application-layer program, but only the client program is needed, because, as we can see, the client program never reaches the application layer in the destination host. In other words, there is no *traceroute* server program. The *traceroute* application program is encapsulated in a UDP user datagram, but *traceroute* intentionally uses a port number that is not available at the destination. If there are n routers in the path, the *traceroute* program sends $(n + 1)$ messages. The first n messages are discarded by the n routers, one by each router; the last message is discarded by the destination host. The *traceroute* client program uses the $(n + 1)$ ICMP error-reporting messages received to find the path between the routers. We will show shortly that the *traceroute* program does not need to know the value of n ; it is found automatically. In Figure 19.10, the value of n is 3.

The first *traceroute* message is sent with time-to-live (TTL) value set to 1; the message is discarded at the first router and a time-exceeded ICMP error message is sent, from which the *traceroute* program can find the IP address of the first router (the source IP address of the error message) and the router name (in the data section of the message). The second *traceroute* message is sent with TTL set to 2, which can find the IP address and the name of the second router. Similarly, the third message can find the information about router 3. The fourth message, however, reaches the destination host. This host is also dropped, but for another reason. The destination host cannot find the port number specified in the UDP user datagram. This time ICMP sends a different message, the destination-unreachable message with code 3. After receiving this different ICMP message, to show the port number is not found. After receiving this different ICMP message, the *traceroute* program knows that the final destination is reached. It uses the information in the received message to find the IP address and the name of the final destination.

Process
decrement (TTL)
و
Process
و
Process

Figure 19.10 Use of ICMPv4 in traceroute



The *traceroute* program also sets a timer to find the round-trip time for each router and the destination. Most *traceroute* programs send three messages to each device, with the same TTL value, to be able to find a better estimate for the round-trip time. The following shows an example of a *traceroute* program, which uses three probes for each device and gets three RTTs.

```
$ traceroute printers.com
traceroute to printers.com (13.1.69.93), 30 hops max, 38-byte packets
 1 route.front.edu      (153.18.31.254)      0.622 ms    0.891 ms    0.875 ms
 2 ceneric.net          (137.164.32.140)    3.069 ms    2.875 ms    2.930 ms
 3 satire.net           (132.16.132.20)     3.071 ms    2.876 ms    2.929 ms
 4 alpha.printers.com   (13.1.69.93)        5.922 ms    5.048 ms    4.922 ms
```

Tracert

The *tracert* program in windows behaves differently. The *tracert* messages are encapsulated directly in IP datagrams. The *tracert*, like *traceroute*, sends echo-request messages. However, when the last echo request reaches the destination host, an echo-replay message is issued.

19.2.3 ICMP Checksum

In ICMP the checksum is calculated over the entire message (header and data).

Example 19.12

Figure 19.11 shows an example of checksum calculation for a simple echo-request message. We randomly chose the identifier to be 1 and the sequence number to be 9. The message is divided

CHAPTER 20

[Unicast Routing] :-

In an internet, the goal of the network layer is to deliver a datagram from its source to its destination or destinations. If a datagram is destined for only one destination (one-to-one delivery), we have *unicast routing*. If the datagram is destined for several destinations (one-to-many delivery), we have *multicast routing*.

In the previous chapters, we have shown that the routing can be possible if a router has a forwarding table to forward a packet to the appropriate next node on its way to the final destination or destinations. To make the forwarding tables of the router, the Internet needs routing protocols that will be active all the time in the background and update the forwarding tables.

In this chapter we discuss only unicast routing; multicast routing will be discussed in the next chapter. This chapter is divided into three sections:

- The first section introduces the concept of unicast routing and describes the general ideas behind it. The section then describes least-cost routing and least-cost trees.
main Algorithm :-
- The second section discusses common routing algorithms used in the Internet. The section first describes distance-vector routing. It then describes link-state routing.
Finally, it explains path-vector routing.
- The third section explores unicast-routing protocols corresponding to the unicast-routing algorithms discussed in the second-section. This section first defines the structure of the Internet as seen by the unicast-routing protocols. It then describes RIP, a protocol that implements the distance-vector routing algorithm. The section next describes OSPF, a protocol that implements the link-state routing algorithm. Finally, the section describes the BGP, a protocol that implements the path-vector routing algorithm.

دو نوع کوکلاز

Routing Protocol
 Routing Algorithms
 بهی قضا حاکم بی این این
 فاصله الی این فضا بی این
 Time لا

20.1 INTRODUCTION

Unicast routing in the Internet, with a large number of routers and a huge number of hosts, can be done only by using hierarchical routing; routing in several steps using different routing algorithms. In this section, we first discuss the general concept of unicast routing in an *internet*, an internetwork made of networks connected by routers. After the routing concepts and algorithms are understood, we show how we can apply them to the Internet using hierarchical routing.

این ال design لا تبق
 به مکان واحد کن تبق
 به عده امکان به عده پروتوکول
 به عمل و نظایف های پروتوکول
 به طری ال service

20.1.1 General Idea

In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables. The source host needs no forwarding table because it delivers its packet to the default router in its local network. The destination host needs no forwarding table either because it receives the packet from its default router in its local network. This means that only the routers that glue together the networks in the internet need forwarding tables. With the above explanation, routing a packet from its source to its destination means routing the packet from a *source router* (the default router of the source host) to a *destination router* (the router connected to the destination network). Although a packet needs to visit the source and the destination routers, the question is what other routers the packet should visit. In other words, there are several routes that a packet can travel from the source to the destination; what must be determined is which route the packet should take.

لاستند
 Default gateway
 address
 source
 router
 Forwarding Tables
 Dist
 Cost

An Internet as a Graph

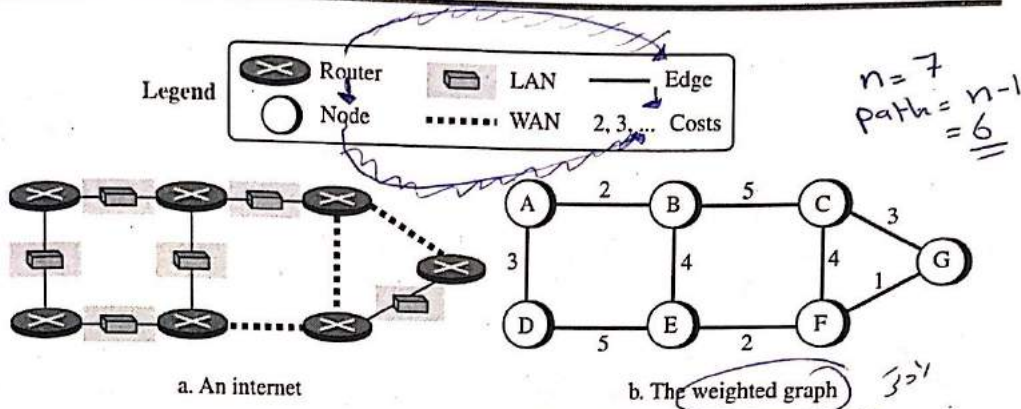
To find the best route, an internet can be modeled as a *graph*. A graph in computer science is a set of *nodes* and *edges* (lines) that connect the nodes. To model an internet as a graph, we can think of each router as a node and each network between a pair of routers as an edge. An internet is, in fact, modeled as a *weighted graph*, in which each edge is associated with a cost. If a weighted graph is used to represent a geographical area, the nodes can be cities and the edges can be roads connecting the cities; the weights, in this case, are distances between cities. In routing, however, the cost of an edge has a different interpretation in different routing protocols, which we discuss in a later section. For the moment, we assume that there is a cost associated with each edge. If there is no edge between the nodes, the cost is infinity. Figure 20.1 shows how an internet can be modeled as a graph.

به هزینه وقت اد
 اد node الی بی این
 به

20.1.2 Least-Cost Routing

When an internet is modeled as a *weighted graph*, one of the ways to interpret the *best* route from the source router to the destination router is to find the *least cost* between the two. In other words, the source router chooses a route to the destination router in such a way that the *total cost* for the route is the least cost among all possible routes. In Figure 20.1, the best route between A and E is A-B-E, with the cost of 6. This means that each router needs to find the least-cost route between itself and all the other routers to be able to route a packet using this criteria.

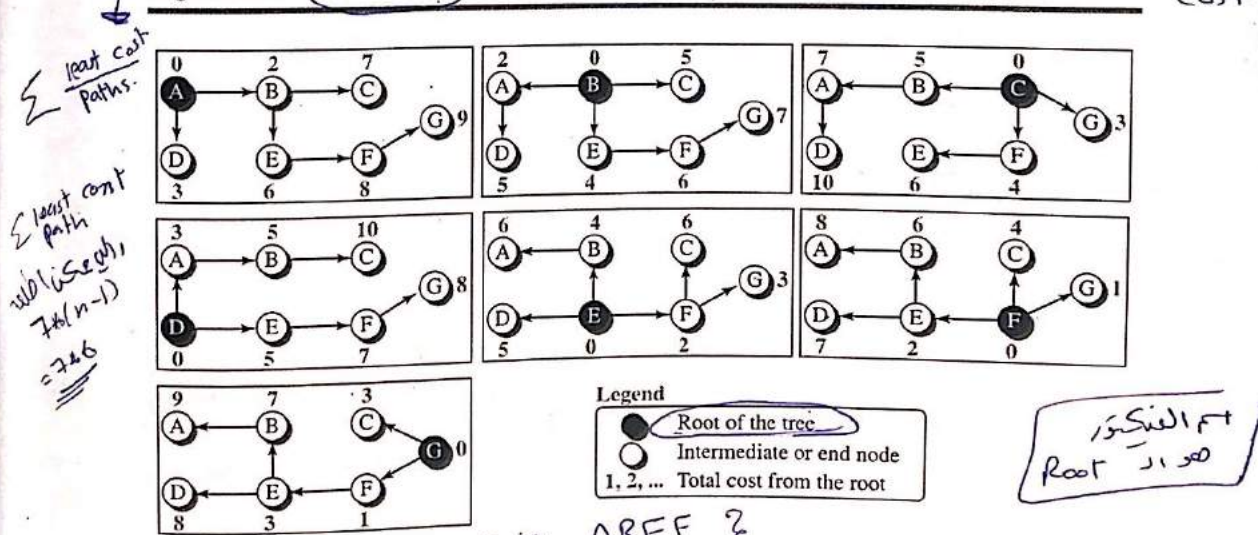
Figure 20.1 An internet and its graphical representation



Least-Cost Trees

If there are N routers in an internet, there are $(N - 1)$ least-cost paths from each router to any other router. This means we need $N \times (N - 1)$ least-cost paths for the whole internet. If we have only 10 routers in an internet, we need 90 least-cost paths. A better way to see all of these paths is to combine them in a **least-cost tree**. A least-cost tree is a tree with the source router as the root that spans the whole graph (visits all other nodes) and in which the path between the root and any other node is the shortest. In this way, we can have only one shortest-path tree for each node; we have N least-cost trees for the whole internet. We show how to create a least-cost tree for each node later in this section; for the moment, Figure 20.2 shows the seven least-cost trees for the internet in Figure 20.1.

Figure 20.2 Least-cost trees for nodes in the internet of Figure 20.1



Handwritten notes and calculations:
 $F \cup A \cup B \cup E \cup F$
 $F \cup B \cup E \cup F$
 $F \cup E \cup F$
 Tree \rightarrow details ipāw path n n

The least-cost trees for a weighted graph can have several properties if they are created using consistent criteria.

1. The least-cost route from X to Y in X's tree is the inverse of the least-cost route from Y to X in Y's tree; the cost in both directions is the same. For example, in Figure 20.2, the route from A to F in A's tree is (A → B → E → F), but the route from F to A in F's tree is (F → E → B → A), which is the inverse of the first route. The cost is 8 in each case.
2. Instead of travelling from X to Z using X's tree, we can travel from X to Y using X's tree and continue from Y to Z using Y's tree. For example, in Figure 20.2, we can go from A to G in A's tree using the route (A → B → E → F → G). We can also go from A to E in A's tree (A → B → E) and then continue in E's tree using the route (E → F → G). The combination of the two routes in the second case is the same route as in the first case. The cost in the first case is 9; the cost in the second case is also 9 (6 + 3).

20.2 ROUTING ALGORITHMS :-

After discussing the general idea behind least-cost trees and the forwarding tables that can be made from them, now we concentrate on the routing algorithms. Several routing algorithms have been designed in the past. The differences between these methods are in the way they interpret the least cost and the way they create the least-cost tree for each node. In this section, we discuss the common algorithms; later we show how a routing protocol in the Internet implements one of these algorithms.

استعملوا الـ algo
عن بعد

20.2.1 Distance-Vector Routing :-

The distance-vector (DV) routing uses the goal we discussed in the introduction, to find the best route. In distance-vector routing, the first thing each node creates is its own least-cost tree with the rudimentary information it has about its immediate neighbors. The incomplete trees are exchanged between immediate neighbors to make the trees more and more complete and to represent the whole internet. We can say that in distance-vector routing, a router continuously tells all of its neighbors what it knows about the whole internet (although the knowledge can be incomplete).

Before we show how incomplete least-cost trees can be combined to make complete ones, we need to discuss two important topics: the Bellman-Ford equation and the concept of distance vectors, which we cover next.

Bellman-Ford Equation :-

The heart of distance-vector routing is the famous Bellman-Ford equation. This equation is used to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes (a, b, c, ...) when the costs between the source and the intermediary nodes and the least costs between the intermediary nodes and the destination are given. The following shows the general case in which D_{ij} is the shortest distance and c_{ij} is the cost between nodes i and j.

$$D_{xy} = \min \{ \underbrace{(c_{xa} + D_{ay})}_{\text{cost}}, \underbrace{(c_{xb} + D_{by})}_{\text{cost}}, \underbrace{(c_{xc} + D_{cy})}_{\text{cost}}, \dots \}$$

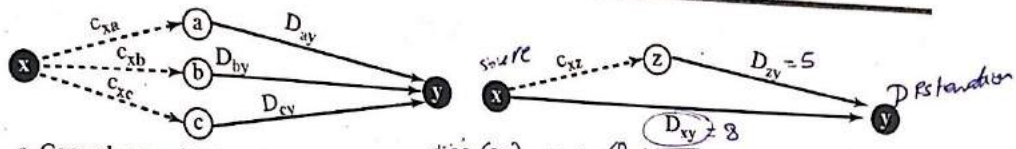
في الـ min
الـ min

In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as z, if the latter is shorter. In this case, the equation becomes simpler, as shown below:

$$D_{xy} = \min \{ D_{xy}, (c_{xz} + D_{zy}) \}$$

Figure 20.3 shows the idea graphically for both cases.

Figure 20.3 Graphical idea behind Bellman-Ford equation



a. General case with three intermediate nodes

b. Updating a path with a new route

فصل لبر صاعدي (Paulure) رح لبر (oo) ففناز
 في صفها اي اربنا لبر بنا د كلج (Path)
 برع اجد ال (Path)

We can say that the Bellman-Ford equation enables us to build a new least-cost path from previously established least-cost paths. In Figure 20.3, we can think of (a→y), (b→y), and (c→y) as previously established least-cost paths and (x→y) as the new least-cost path. We can even think of this equation as the builder of a new least-cost tree from previously established least-cost trees if we use the equation repeatedly. In other words, the use of this equation in distance-vector routing is a witness that this method also uses least-cost trees, but this use may be in the background.

We will shortly show how we use the Bellman-Ford equation and the concept of distance vectors to build least-cost paths for each node in distance-vector routing, but first we need to discuss the concept of a distance vector.

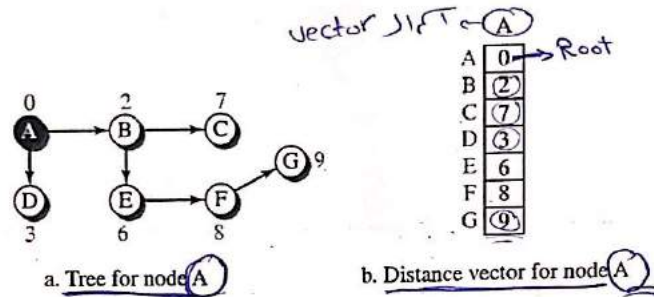
Distance Vectors

The concept of a distance vector is the rationale for the name distance-vector routing. A least-cost tree is a combination of least-cost paths from the root of the tree to all destinations. These paths are graphically glued together to form the tree. Distance-vector routing unglues these paths and creates a distance vector, a one-dimensional array to represent the tree. Figure 20.4 shows the tree for node A in the internet in Figure 20.1 and the corresponding distance vector.

Note that the name of the distance vector defines the root, the indexes define the destinations, and the value of each cell defines the least cost from the root to the destination. A distance vector does not give the path to the destinations as the least-cost tree does; it gives only the least costs to the destinations. Later we show how we can change a distance vector to a forwarding table, but we first need to find all distance vectors for an internet.

We know that a distance vector can represent least-cost paths in a least-cost tree, but the question is how each node in an internet originally creates the corresponding vector. Each node in an internet, when it is booted, creates a very rudimentary distance vector with the minimum information the node can obtain from its neighborhood. The node sends some greeting messages out of its interfaces and discovers the identity of the immediate neighbors and the distance between itself and each neighbor. It then

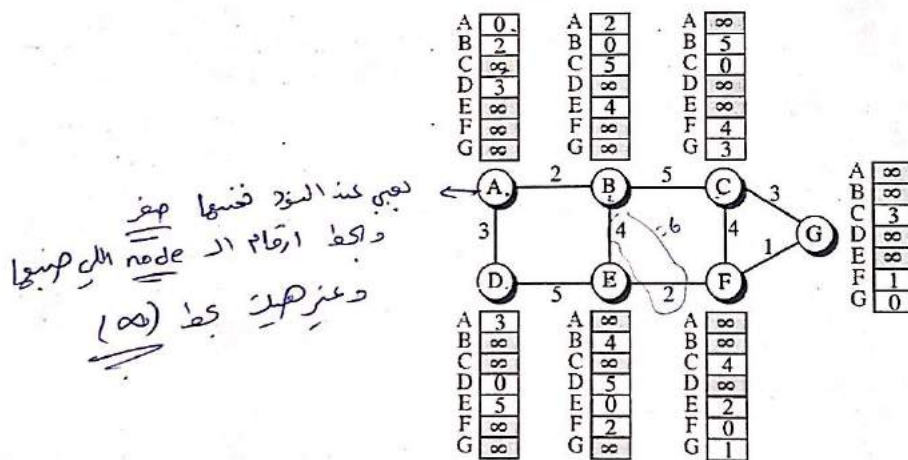
Figure 20.4 The distance vector corresponding to a tree



makes a simple distance vector by inserting the discovered distances in the corresponding cells and leaves the value of other cells as infinity. Do these distance vectors represent least-cost paths? They do, considering the limited information a node has. When we know only one distance between two nodes, it is the least cost. Figure 20.5 shows all distance vectors for our internet. However, we need to mention that these vectors are made asynchronously, when the corresponding node has been booted; the existence of all of them in a figure does not mean synchronous creation of them.

send what it know to neighbors

Figure 20.5 The first distance vector for an internet

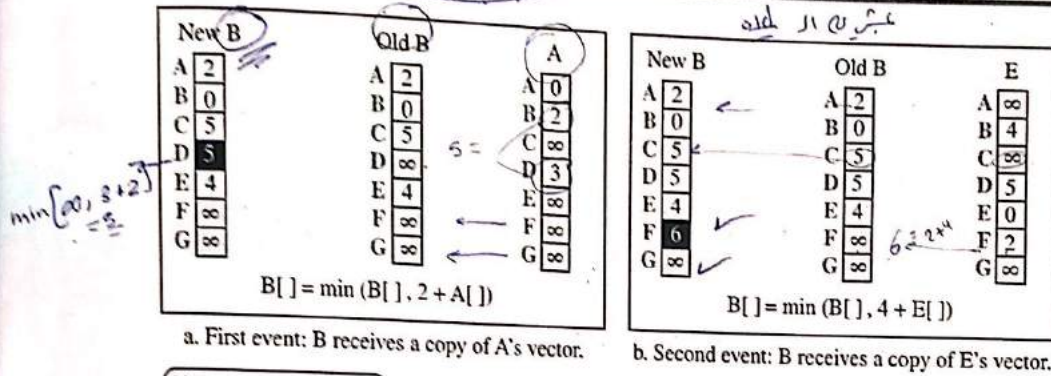


دعني عند السؤ قضتها صفر
داكط ارقام ال node ال صيرتيا
وعز صيرت بط (∞)

These rudimentary vectors cannot help the internet to effectively forward a packet. For example, node A thinks that it is not connected to node G because the corresponding cell shows the least cost of infinity. To improve these vectors, the nodes in the internet need to help each other by exchanging information. After each node has created its vector, it sends a copy of the vector to all its immediate neighbors. After a node receives a distance vector from a neighbor, it updates its distance vector using the Bellman-Ford equation (second case). However, we need to understand that we need to update, not

only one least cost, but N of them in which N is the number of the nodes in the internet. If we are using a program, we can do this using a loop; if we are showing the concept on paper, we can show the whole vector instead of the N separate equations. We show the whole vector instead of seven equations for each update in Figure 20.6. The figure shows two asynchronous events, happening one after another with some time in

Figure 20.6 Updating distance vectors



Note:
 $X[]$: the whole vector

between. In the first event, node A has sent its vector to node B. Node B updates its vector using the cost $c_{BA} = 2$. In the second event, node E has sent its vector to node B. Node B updates its vector using the cost $c_{EA} = 4$.

After the first event, node B has one improvement in its vector: its least cost to node D has changed from infinity to 5 (via node A). After the second event, node B has one more improvement in its vector; its least cost to node F has changed from infinity to 6 (via node E). We hope that we have convinced the reader that exchanging vectors eventually stabilizes the system and allows all nodes to find the ultimate least cost between themselves and any other node. We need to remember that after updating a node, it immediately sends its updated vector to all neighbors. Even if its neighbors have received the previous vector, the updated one may help more.

Distance-Vector Routing Algorithm

Now we can give a simplified pseudocode for the distance-vector routing algorithm, as shown in Table 20.1. The algorithm is run by its node independently and asynchronously.

Table 20.1 Distance-Vector Routing Algorithm for a Node

```

1 Distance_Vector_Routing ()
2 {
3     // Initialize (create initial vectors for the node)
4     D[myself] = 0
    
```

Table 20.1 Distance-Vector Routing Algorithm for a Node (continued)

```

5   for (y = 1 to N)
6   {
7       if (y is a neighbor)
8           D[y] = c[myself][y]
9       else
10          D[y] = ∞
11  }
12  send vector {D[1], D[2], ..., D[N]} to all neighbors
13  // Update (improve the vector with the vector received from a neighbor)
14  repeat (forever)
15  {
16      wait (for a vector Dw from a neighbor w or any change in the link)
17      for (y = 1 to N)
18      {
19          D[y] = min [D[y], (c[myself][w] + Dw[y])] // Bellman-Ford equation
20      }
21      if (any change in the vector)
22          send vector {D[1], D[2], ..., D[N]} to all neighbors
23  }
24  ) // End of Distance Vector

```

Whole Internal

Lines 4 to 11 initialize the vector for the node. Lines 14 to 23 show how the vector can be updated after receiving a vector from the immediate neighbor. The *for* loop in lines 17 to 20 allows all entries (cells) in the vector to be updated after receiving a new vector. Note that the node sends its vector in line 12, after being initialized, and in line 22, after it is updated.

Count to Infinity

A problem with distance-vector routing is that any decrease in cost (good news) propagates quickly, but any increase in cost (bad news) will propagate slowly. For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance-vector routing, this takes some time. The problem is referred to as *count to infinity*. It sometimes takes several updates before the cost for a broken link is recorded as infinity by all routers.

Two-Node Loop

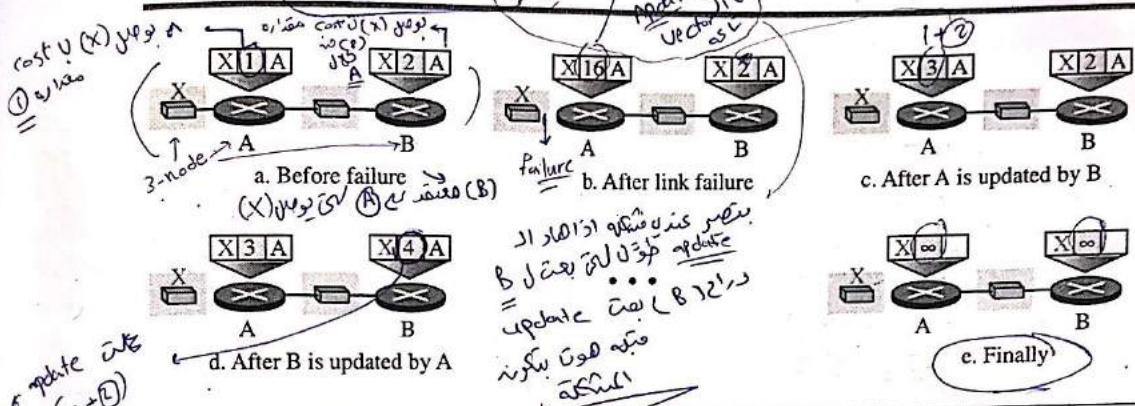
One example of count to infinity is the two-node loop problem. To understand the problem, let us look at the scenario depicted in Figure 20.7.

The figure shows a system with three nodes. We have shown only the portions of the forwarding table needed for our discussion. At the beginning, both nodes A and B

Hop count → ∞ Destination

في لما يكون العدد اى حده (15) اجبا ∞ في المثل عند A = 16 (هيك يكون مكانها ∞)

Figure 20.7 Two-node instability



know how to reach node X. But suddenly, the link between A and X fails. Node A changes its table. If A can send its table to B immediately, everything is fine. However, the system becomes unstable if B sends its forwarding table to A before receiving A's forwarding table. Node A receives the update and, assuming that B has found a way to reach X, immediately updates its forwarding table. Now A sends its new update to B. Now B thinks that something has been changed around A and updates its forwarding table. The cost of reaching X increases gradually until it reaches infinity. At this moment, both A and B know that X cannot be reached. However, during this time the system is not stable. Node A thinks that the route to X is via B; node B thinks that the route to X is via A. If A receives a packet destined for X, the packet goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem. A few solutions have been proposed for instability of this kind.

Split Horizon

One solution to instability is called split horizon. In this strategy, instead of flooding the table through each interface, each node sends only part of its table through each interface. If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows). Taking information from node A, modifying it, and sending it back to node A is what creates the confusion. In our scenario, node B eliminates the last line of its forwarding table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later, when node A sends its forwarding table to B, node B also corrects its forwarding table. The system becomes stable after the first update: both node A and node B know that X is not reachable.

Poison Reverse

Using the split-horizon strategy has one drawback. Normally, the corresponding protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess whether this is due to the split-horizon strategy (the source of information was A) or because B has not received any news about X recently. In the poison reverse strategy B can still advertise the value for X, but if the source of

Handwritten notes on the left margin:
 - مقارنه (1)
 - مقارنه (2)
 - مقارنه (3)
 - مقارنه (4)
 - مقارنه (5)
 - مقارنه (6)
 - مقارنه (7)
 - مقارنه (8)
 - مقارنه (9)
 - مقارنه (10)
 - مقارنه (11)
 - مقارنه (12)
 - مقارنه (13)
 - مقارنه (14)
 - مقارنه (15)
 - مقارنه (16)
 - مقارنه (17)
 - مقارنه (18)
 - مقارنه (19)
 - مقارنه (20)
 - مقارنه (21)
 - مقارنه (22)
 - مقارنه (23)
 - مقارنه (24)
 - مقارنه (25)
 - مقارنه (26)
 - مقارنه (27)
 - مقارنه (28)
 - مقارنه (29)
 - مقارنه (30)
 - مقارنه (31)
 - مقارنه (32)
 - مقارنه (33)
 - مقارنه (34)
 - مقارنه (35)
 - مقارنه (36)
 - مقارنه (37)
 - مقارنه (38)
 - مقارنه (39)
 - مقارنه (40)
 - مقارنه (41)
 - مقارنه (42)
 - مقارنه (43)
 - مقارنه (44)
 - مقارنه (45)
 - مقارنه (46)
 - مقارنه (47)
 - مقارنه (48)
 - مقارنه (49)
 - مقارنه (50)

Handwritten notes on the right margin:
 - ولكن
 - في بعض
 - المعينات
 - طبقا
 - البروتوكول
 - تستخدم
 - وهدا
 - فهم
 - تفعل
 - هادي
 - وين
 - تجاوز
 - ال
 - يعتبر
 - و
 - قابل
 - restable

X	A
---	---

وال (A) معانا انه ط تعرف

information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

Three-Node Instability

The two-node instability can be avoided using split horizon combined with poison reverse. However, if the instability is between three nodes, stability cannot be guaranteed.

بعض اوقات Link اور Edge
 كم عليه (delay) و هو عليه
 Traffic
 (في سوال عن حاله)
 الحرف

20.2.2 Link-State Routing

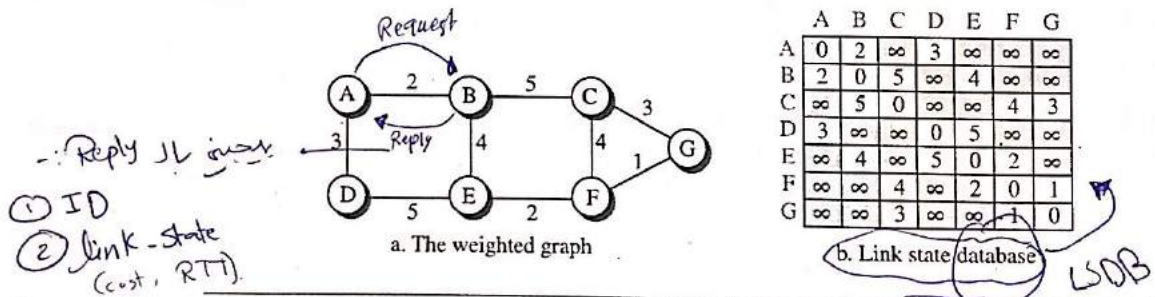


A routing algorithm that directly follows our discussion for creating least-cost trees and forwarding tables is link-state (LS) routing. This method uses the term link-state to define the characteristic of a link (an edge) that represents a network in the internet. In this algorithm the cost associated with an edge defines the state of the link. Links with lower costs are preferred to links with higher costs; if the cost of a link is infinity, it means that the link does not exist or has been broken.

Link-State Database (LSDB)

To create a least-cost tree with this method, each node needs to have a complete map of the network, which means it needs to know the state of each link. The collection of states for all links is called the link-state database (LSDB). There is only one LSDB for the whole internet; each node needs to have a duplicate of it to be able to create the least-cost tree. Figure 20.8 shows an example of an LSDB for the graph in Figure 20.1. The LSDB can be represented as a two-dimensional array (matrix) in which the value of each cell defines the cost of the corresponding link.

Figure 20.8 Example of a link-state database

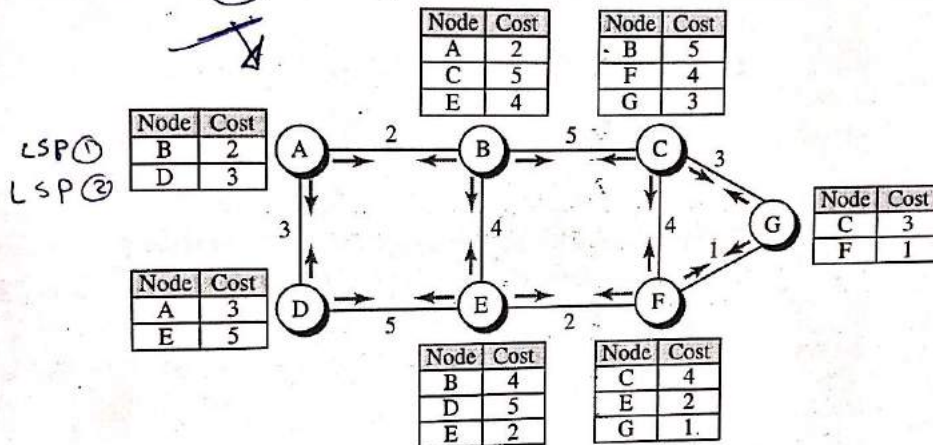


∴ Reply لا يحتاج
 ① ID
 ② link-state (cost, RTT)

Now the question is how each node can create this LSDB that contains information about the whole internet. This can be done by a process called flooding. Each node can send some greeting messages to all its immediate neighbors (those nodes to which it is connected directly) to collect two pieces of information for each neighboring node: the identity of the node and the cost of the link. The combination of these two pieces of information is called the LS packet (LSP); the LSP is sent out of each interface, as shown in Figure 20.9 for our internet in Figure 20.1. When a node receives an LSP from one of its interfaces, it compares the LSP with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer or the first one received, the node discards the old LSP (if there is one) and keeps the received one. It then sends a copy of it out of each

interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the network (where a node has only one interface). We need to convince ourselves that, after receiving all new LSPs, each node creates the comprehensive LSDB as shown in Figure 20.9. This LSDB is the same for each node and shows the whole map of the internet. In other words, a node can make the whole map if it needs to, using this LSDB.

Figure 20.9 LSPs created and sent out by each node to build LSDB



We can compare the link-state routing algorithm with the distance-vector routing algorithm. In the distance-vector routing algorithm, each router tells its neighbors what it knows about the whole internet; in the link-state routing algorithm, each router tells the whole internet what it knows about its neighbors.

الروتين
المسافة
Distance
link state

Formation of Least-Cost Trees

To create a least-cost tree for itself, using the shared LSDB, each node needs to run the famous **Dijkstra Algorithm**. This iterative algorithm uses the following steps:

1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.
2. The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree. After this node is added to the tree, the cost of all other nodes not in the tree needs to be updated because the paths may have been changed.
3. The node repeats step 2 until all nodes are added to the tree.

We need to convince ourselves that the above three steps finally create the least-cost tree. Table 20.2 shows a simplified version of Dijkstra's algorithm.

Table 20.2 Dijkstra's Algorithm

1	Dijkstra's Algorithm ()
2	{
3	// Initialization
4	Tree = {root} // Tree is made only of the root

Table 20.2 Dijkstra's Algorithm (continued)

```

5   for (y = 1 to N)           // N is the number of nodes
6   {
7       if (y is the root)
8           D[y] = 0           // D[y] is shortest distance from root to node y
9       else if (y is a neighbor)
10          D[y] = c[root][y]  // c[x][y] is cost between nodes x and y in LSDB
11      else
12          D[y] = ∞           // non neighbor
13  }
14  // Calculation
15  repeat
16  {
17      find a node w, with D[w] minimum among all nodes not in the Tree
18      Tree = Tree ∪ {w}      // Add w to tree
19      // Update distances for all neighbors of w
20      for (every node x, which is a neighbor of w and not in the Tree)
21      {
22          D[x] = min{D[x], (D[w] + c[w][x])}
23      }
24  } until (all nodes included in the Tree)
25 } // End of Dijkstra
    
```

Lines 4 to 13 implement step 1 in the algorithm. Lines 16 to 23 implement step 2 in the algorithm. Step 2 is repeated until all nodes are added to the tree.

Figure 20.10 shows the formation of the least-cost tree for the graph in Figure 20.8 using Dijkstra's algorithm. We need to go through an initialization step and six iterations to find the least-cost tree.

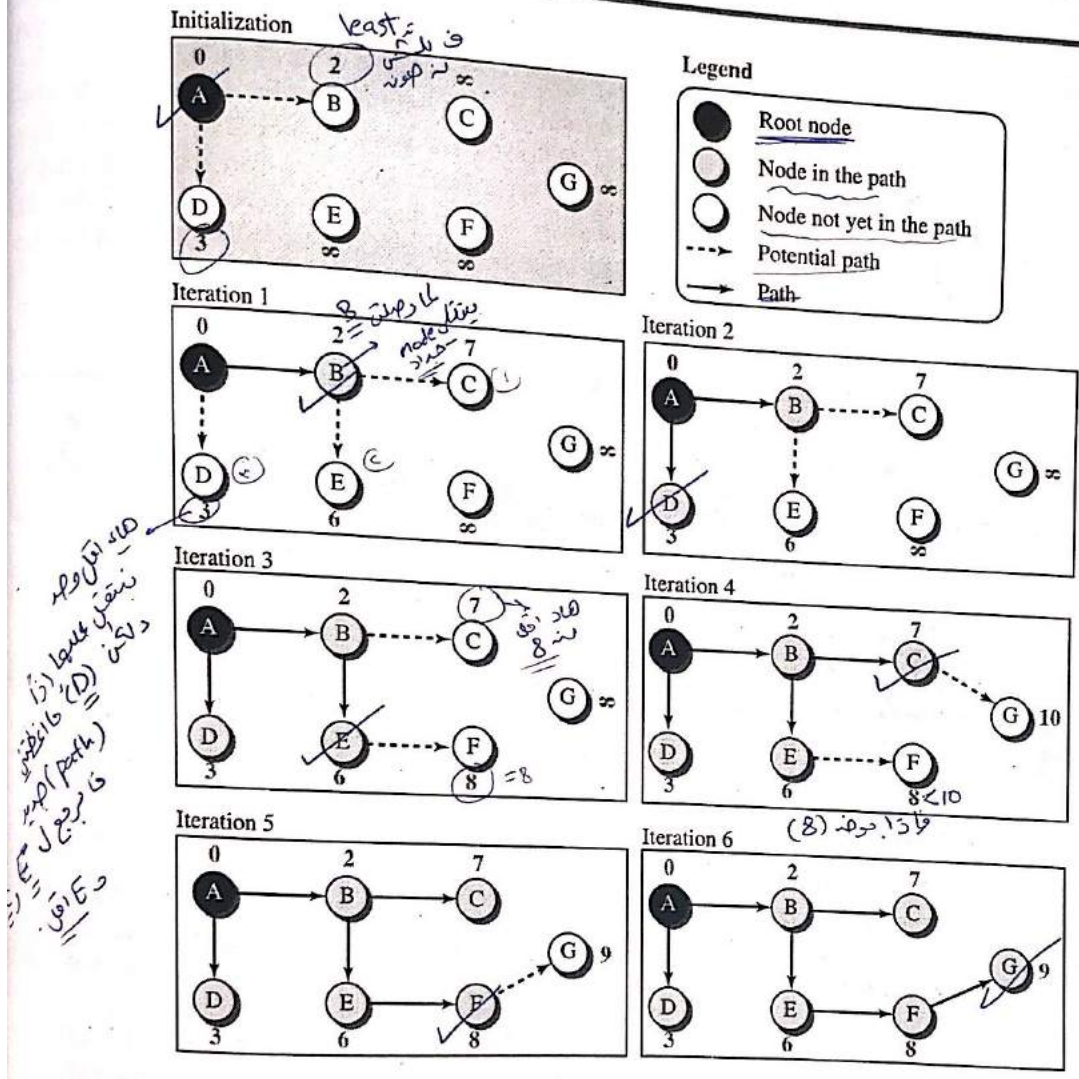
20.2.3 Path-Vector Routing

Both link-state and distance-vector routing are based on the least-cost goal. However, there are instances where this goal is not the priority. For example, assume that there are some routers in the internet that a sender wants to prevent its packets from going through. For example, a router may belong to an organization that does not provide enough security or it may belong to a commercial rival of the sender which might inspect the packets for obtaining information. Least-cost routing does not prevent a packet from passing through an area when that area is in the least-cost path. In other words, the least-cost goal, applied by (LS or DV routing), does not allow a sender to apply specific policies to the route a packet may take. Aside from safety and security, there are occasions, as discussed in the next section, in which the goal of routing is merely reachability: to allow the packet to reach its destination more efficiently without assigning costs to the route.

حسب كل ال (node) ليحتمو بعضي الوقت
 كتي فايبر كتي
 load ال ال Traffic
 (Flooding) داسه
 دليمنه انه فايبر
 بعض الوقت نيه
 ظلال ال Timer

سي فضل ال (Path-Vector Routing) انا برد (Policy)
 عند مقارنتها بختيار ال Path ال بدني اعمى نيه لهدك فايبر كتي ال least cost path

Figure 20.10 Least-cost tree



To respond to these demands, a third routing algorithm, called path-vector (PV) routing has been devised. Path-vector routing does not have the drawbacks of LS or DV routing as described above because it is not based on least-cost routing. The best route is determined by the source using the policy it imposes on the route. In other words, the source can control the path. Although path-vector routing is not actually used in an internet, and is mostly designed to route a packet between ISPs, we discuss the principle of this method in this section as though applied to an internet. In the next section, we show how it is used in the Internet.

Spanning Trees = - redundant paths

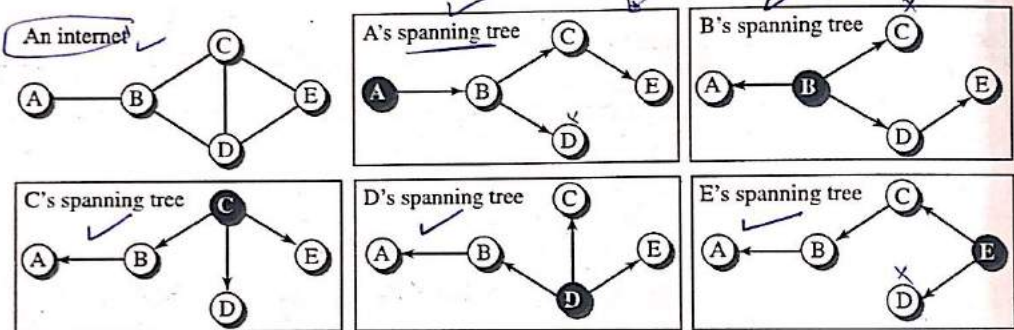
In path-vector routing, the path from a source to all destinations is also determined by the best spanning tree. The best spanning tree, however, is not the least-cost tree; it is

یعنی Tree به معنای گره های گونمندی
 یعنی بهترین (Best) بینار (D) است

the tree determined by the source when it imposes its own policy. If there is more than one route to a destination, the source can choose the route that meets its policy best. A source may apply several policies at the same time. One of the common policies uses the minimum number of nodes to be visited (something similar to least-cost). Another common policy is to avoid some nodes as the middle node in a route.

Figure 20.11 shows a small internet with only five nodes. Each source has created its own spanning tree that meets its policy. The policy imposed by all sources is to use the minimum number of nodes to reach a destination. The spanning tree selected by A and E is such that the communication does not pass through D as a middle node. Similarly, the spanning tree selected by B is such that the communication does not pass through C as a middle node.

Figure 20.11 Spanning trees in path-vector routing



لانه عمدي
اكثر من path
بودي شي
دور...
node

Creation of Spanning Trees

Path-vector routing, like distance-vector routing, is an asynchronous and distributed routing algorithm. The spanning trees are made, gradually and asynchronously, by each node. When a node is booted, it creates a path vector based on the information it can obtain about its immediate neighbor. A node sends greeting messages to its immediate neighbors to collect these pieces of information. Figure 20.12 shows all of these path vectors for our internet in Figure 20.11. Note, however, that we do not mean that all of these tables are created simultaneously; they are created when each node is booted. The figure also shows how these path vectors are sent to immediate neighbors after they have been created (arrows).

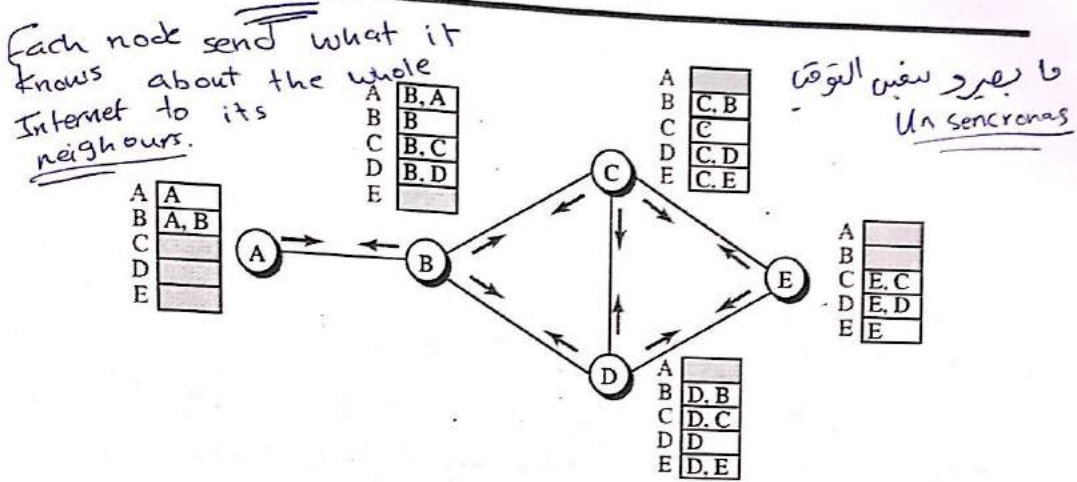
Each node, after the creation of the initial path vector, sends it to all its immediate neighbors. Each node, when it receives a path vector from a neighbor, updates its path vector using an equation similar to the Bellman-Ford, but applying its own policy instead of looking for the least cost. We can define this equation as

$$Path(x, y) = \text{best} \{ Path(x, y), [x + Path(v, y)] \} \text{ for all } v\text{'s in the internet.}$$

In this equation, the operator (+) means to add x to the beginning of the path. We also need to be cautious to avoid adding a node to an empty path because an empty path means one that does not exist.

يعني...
A B C E
A B D E
Bellman-Ford

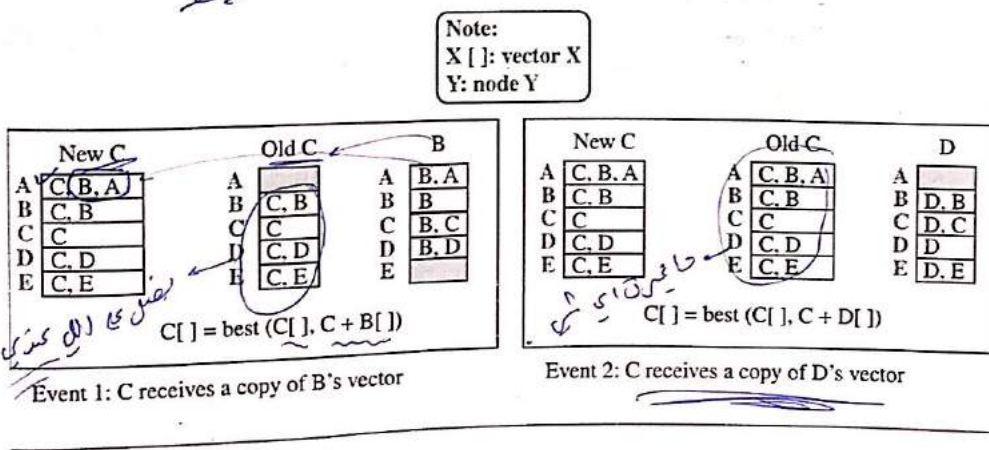
Figure 20.12 Path vectors made at booting time



The policy is defined by selecting the *best* of multiple paths. Path-vector routing also imposes one more condition on this equation: If Path (v, y) includes x, that path is discarded to avoid a loop in the path. In other words, x does not want to visit itself when it selects a path to y.

Figure 20.13 shows the path vector of node C after two events. In the first event, node C receives a copy of B's vector, which improves its vector: now it knows how to reach node A. In the second event, node C receives a copy of D's vector, which does not change its vector. As a matter of fact the vector for node C after the first event is stabilized and serves as its forwarding table.

Figure 20.13 Updating path vectors



Path-Vector Algorithm

Based on the initialization process and the equation used in updating each forwarding table after receiving path vectors from neighbors, we can write a simplified version of the path vector algorithm as shown in Table 20.3.

Table 20.3 Path-vector algorithm for a node

```

1 Path_Vector_Routing ( )
2 {
3   // Initialization
4   for (y=1 to N)
5   {
6     if (y is myself)
7       Path[y] = myself
8     else if (y is a neighbor)
9       Path[y] = myself + neighbor node
10    else
11      Path[y] = empty
12  }
13  Send vector {Path[1], Path[2], ..., Path[y]} to all neighbors
14  // Update
15  repeat (forever) up to date
16  {
17    wait (for a vector Path_w from a neighbor w)
18    for (y = 1 to N)
19    {
20      if (Path_w includes myself)
21        discard the path // Avoid any loop
22      else
23        Path[y] = best {Path[y], (myself + Path_w[y])}
24    }
25    If (there is a change in the vector)
26      Send vector {Path[1], Path[2], ..., Path[y]} to all neighbors
27  }
28 } // End of Path Vector

```

Handwritten notes in Urdu and English are present around the code. On line 17, there is a note: "wait (for a vector Path_w from a neighbor w)". To the right, there is a diagram showing a node 'A' with an arrow pointing to 'B', with 'AB' written below it. On line 18, there is a note: "for (y = 1 to N)". To the right, there is a note: "یہاں B سے B کے لیے". On line 20, there is a note: "if (Path_w includes myself)". To the right, there is a note: "وہاں Path_w". On line 21, there is a note: "discard the path". To the right, there is a note: "تاکہ لوپ نہ آئے" and "Path_w". On line 22, there is a note: "else". To the right, there is a note: "وہاں Path_w". On line 23, there is a note: "Path[y] = best {Path[y], (myself + Path_w[y])}". To the right, there is a note: "WAB". On line 25, there is a note: "If (there is a change in the vector)". To the right, there is a note: "تاکہ لوپ نہ آئے". On line 26, there is a note: "Send vector {Path[1], Path[2], ..., Path[y]} to all neighbors". To the right, there is a note: "یہاں B سے B کے لیے". On line 28, there is a note: "End of Path Vector".

Lines 4 to 12 show the initialization for the node. Lines 17 to 24 show how the node updates its vector after receiving a vector from the neighbor. The update process is repeated forever. We can see the similarities between this algorithm and the DV algorithm.

20.3 UNICAST ROUTING PROTOCOLS

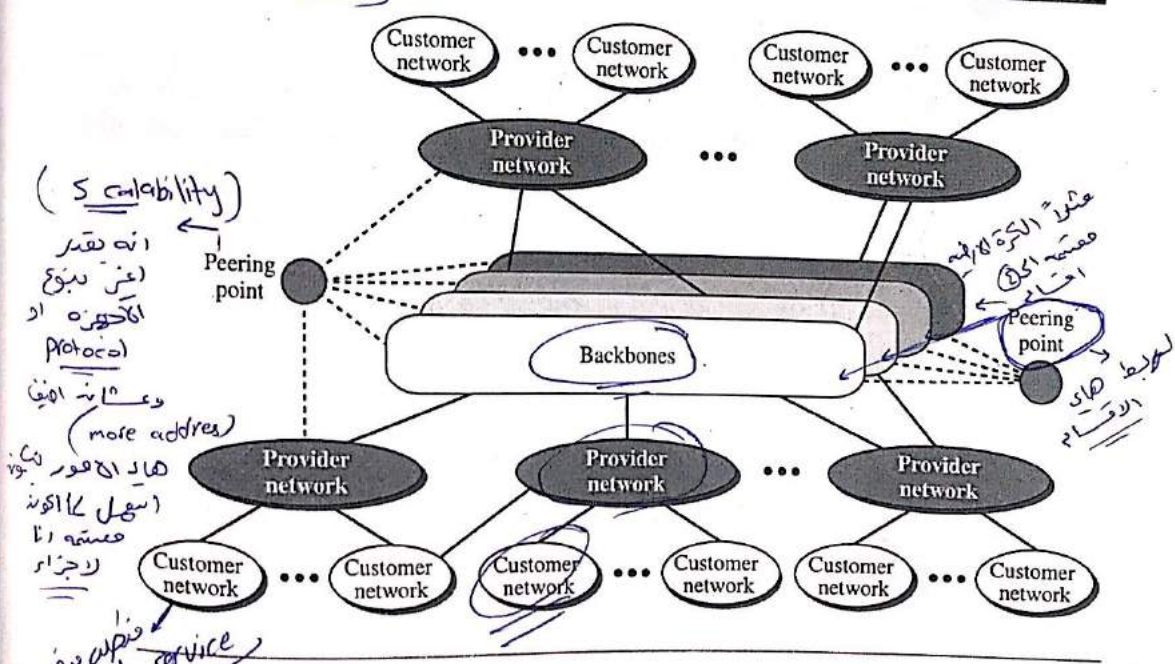
① Algo
 ② Domain
 ③ Message format
 ④ Timers
 شبك
 Topology
 IP → layer 3

In the previous section, we discussed unicast routing algorithms; in this section, we discuss unicast routing protocols used in the Internet. Although three protocols we discuss here are based on the corresponding algorithms we discussed before, a protocol is more than an algorithm. A protocol needs to define its domain of operation, the messages exchanged, communication between routers, and interaction with protocols in other domains. After an introduction, we discuss three common protocols used in the Internet: Routing Information Protocol (RIP), based on the distance-vector algorithm, Open Shortest Path First (OSPF), based on the link-state algorithm, and Border Gateway Protocol (BGP), based on the path-vector algorithm.

20.3.1 Internet Structure

Before discussing unicast routing protocols, we need to understand the structure of today's Internet. The Internet has changed from a tree-like structure, with a single backbone, to a multi-backbone structure run by different private corporations today. Although it is difficult to give a general view of the Internet today, we can say that the Internet has a structure similar to what is shown in Figure 20.14.

Figure 20.14 Internet structure



There are several backbones run by private communication companies that provide global connectivity. These backbones are connected by some peering points that allow connectivity between backbones. At a lower level, there are some provider networks that use the backbones for global connectivity but provide services to Internet customers.

Finally, there are some customer networks that use the services provided by the provider networks. Any of these three entities (backbone, provider network, or customer network) can be called an Internet Service Provider or ISP. They provide services, but at different levels.

Hierarchical Routing

The Internet today is made of a huge number of networks and routers that connect them. It is obvious that routing in the Internet cannot be done using a single protocol for two reasons: a scalability problem and an administrative issue. Scalability problem means that the size of the forwarding tables becomes huge, searching for a destination in a forwarding table becomes time-consuming, and updating creates a huge amount of traffic. The administrative issue is related to the Internet structure described in Figure 20.14. As the figure shows, each ISP is run by an administrative authority. The administrator needs to have control in its system. The organization must be able to use as many subnets and routers as it needs, may desire that the routers be from a particular manufacturer, may wish to run a specific routing algorithm to meet the needs of the organization, and may want to impose some policy on the traffic passing through its ISP.

ماذا تعني
البريد

Hierarchical routing means considering each ISP as an autonomous system (AS). Each AS can run a routing protocol that meets its needs, but the global Internet runs a global protocol to glue all ASs together. The routing protocol run in each AS is referred to as intra-AS routing protocol, intradomain routing protocol, or interior gateway protocol (IGP); the global routing protocol is referred to as inter-AS routing protocol, interdomain routing protocol, or exterior gateway protocol (EGP). We can have several intradomain routing protocols, and each AS is free to choose one, but it should be clear that we should have only one interdomain protocol that handles routing between these entities. Presently, the two common intradomain routing protocols are RIP and OSPF; the only interdomain routing protocol is BGP. The situation may change when we move to IPv6.

Autonomous Systems

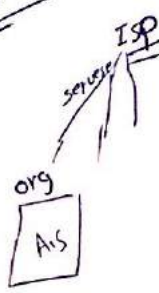
As we said before, each ISP is an autonomous system when it comes to managing networks and routers under its control. Although we may have small, medium-size, and large ASs, each AS is given an autonomous number (ASN) by the ICANN. Each ASN is a 16-bit unsigned integer that uniquely defines an AS. The autonomous systems, however, are not categorized according to their size; they are categorized according to the way they are connected to other ASs. We have stub ASs, multihomed ASs, and transient ASs. The type, as we see will later, affects the operation of the interdomain routing protocol in relation to that AS.

مستقل وليس Automatic

مجموعة من الشبكات التي تملكها هيئة واحدة أو system بغير المساهمة
انه يتبادل مع هي ال systems الموجودة فيه حسب تطلبها الآخر

Stub AS. A stub AS has only one connection to another AS. The data traffic can be either initiated or terminated in a stub AS; the data cannot pass through it. A good example of a stub AS is the customer network, which is either the source or the sink of data. زي النت اللى جاي في البيت بجي اي Traffic جاي لهاد الراوتر

Multihomed AS. A multihomed AS can have more than one connection to other ASs, but it does not allow data traffic to pass through it. A good example of such an AS is some of the customer ASs that may use the services of more than one provider network, but their policy does not allow data to be passed through them.



بعض ال customer اللى بالصفحة اللي قبلت لانه مربوط مع (provider) انه يجي Traffic فيه ممكنة

من ار Provider التي البغية التي من
انها تكون (نقطة عبور)

Transient AS. A transient AS is connected to more than one other AS and also allows the traffic to pass through. The provider networks and the backbone are good examples of transient ASs.

20.3.2 Routing Information Protocol (RIP)

Internal	External
RIP	BGP
OSPF	

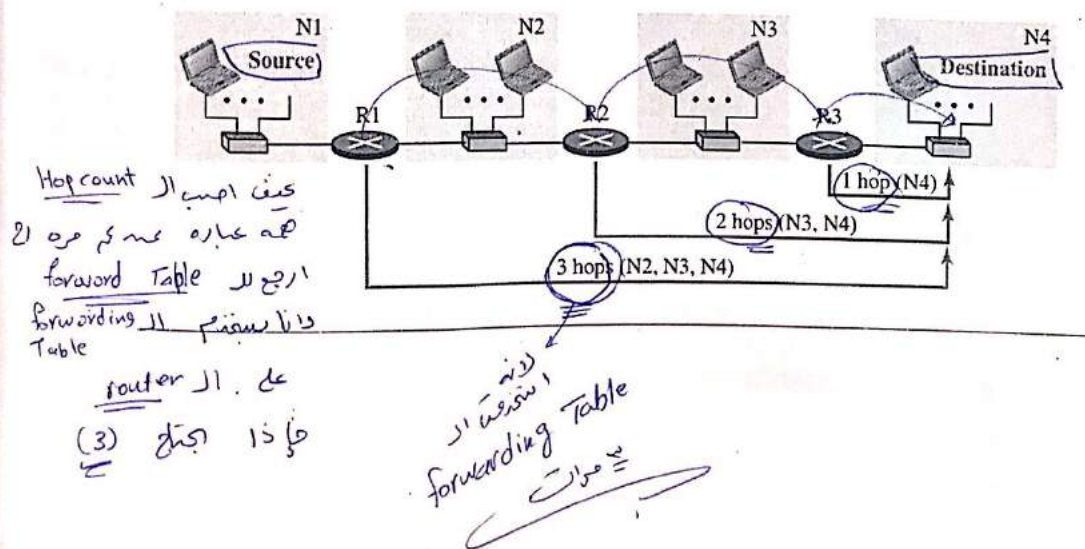
The **Routing Information Protocol (RIP)** is one of the most widely used intradomain routing protocols based on the distance-vector routing algorithm we described earlier. RIP was started as part of the Xerox Network System (XNS), but it was the Berkeley Software Distribution (BSD) version of UNIX that helped make the use of RIP widespread.



Hop Count

A router in this protocol basically implements the distance-vector routing algorithm shown in Table 20.1. However, the algorithm has been modified as described below. First, since a router in an AS needs to know how to forward a packet to different networks (subnets) in an AS, RIP routers advertise the cost of reaching different networks instead of reaching other nodes in a theoretical graph. In other words, the cost is defined between a router and the network in which the destination host is located. Second, to make the implementation of the cost simpler (independent from performance factors of the routers and links, such as delay, bandwidth, and so on), the cost is defined as the number of hops, which means the number of networks (subnets) a packet needs to travel through from the source router to the final destination host. Note that the network in which the source host is connected is not counted in this calculation because the source host does not use a forwarding table; the packet is delivered to the default router. Figure 20.15 shows the concept of hop count advertised by three routers from a source host to a destination host. In RIP, the maximum cost of a path can be 15, which means 16 is considered as infinity (no connection). For this reason, RIP can be used only in autonomous systems in which the diameter of the AS is not more than 15 hops.

Figure 20.15 Hop counts in RIP



Forwarding Tables

Although the distance-vector algorithm we discussed in the previous section is concerned with exchanging distance vectors between neighboring nodes, the routers in an autonomous system need to keep forwarding tables to forward packets to their destination networks. A forwarding table in RIP is a three-column table in which the first column is the address of the destination network, the second column is the address of the next router to which the packet should be forwarded, and the third column is the cost (the number of hops) to reach the destination network. Figure 20.16 shows the three forwarding tables for the routers in Figure 20.15. Note that the first and the third columns together convey the same information as does a distance vector, but the cost shows the number of hops to the destination networks.

Figure 20.16 Forwarding tables

Forwarding table for R1			Forwarding table for R2			Forwarding table for R3		
Destination network	Next router	Cost in hops	Destination network	Next router	Cost in hops	Destination network	Next router	Cost in hops
N1	—	1	N1	R1	2	N1	R2	3
N2	—	1	N2	—	1	N2	R2	2
N3	R2	2	N3	—	1	N3	—	1
N4	R2	3	N4	R3	2	N4	—	1

update في الـ

(number of hops)

1 + (R1 to R2) في الـ $cost = 1 + cost$

Although a forwarding table in RIP defines only the next router in the second column, it gives the information about the whole least-cost tree based on the second property of these trees, discussed in the previous section. For example, R1 defines that the next router for the path to N4 is R2; R2 defines that the next router to N4 is R3; R3 defines that there is no next router for this path. The tree is then R1 → R2 → R3 → N4.

A question often asked about the forwarding table is what the use of the third column is. The third column is not needed for forwarding the packet, but it is needed for updating the forwarding table when there is a change in the route, as we will see shortly.

Application layer

RIP Implementation (Layer 3) LS protocol

RIP is implemented as a process that uses the service of UDP on the well-known port number 520. In BSD, RIP is a daemon process (a process running in the background), named *routed* (abbreviation for *route daemon* and pronounced *route-dee*). This means that, although RIP is a routing protocol to help IP route its datagrams through the AS, the RIP messages are encapsulated inside UDP user datagrams, which in turn are encapsulated inside IP datagrams. In other words, RIP runs at the application layer, but creates forwarding tables for IP at the network layer.

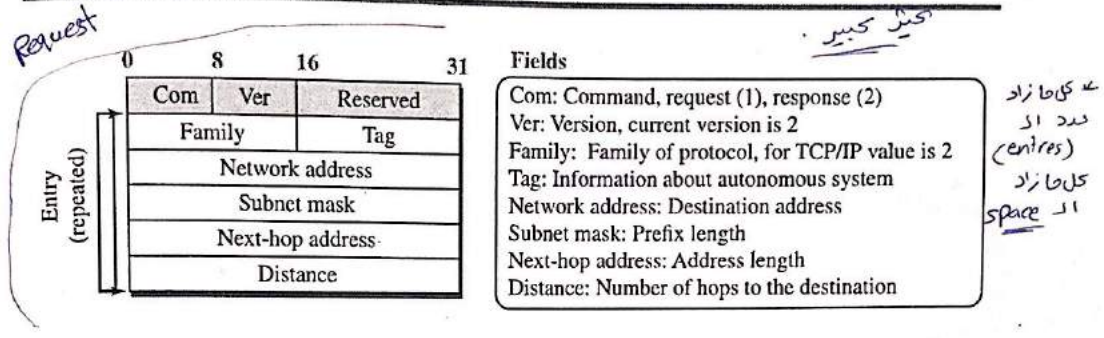
الـ $update$ في الـ router في الـ Forwarding table
 الـ $Back ground$ (Back ground)

RIP has gone through two versions: RIP-1 and RIP-2. The second version is backward compatible with the first section; it allows the use of more information in the RIP messages that were set to 0 in the first version. We discuss only RIP-2 in this section.

RIP Messages

Two RIP processes, a client and a server, like any other processes, need to exchange messages. RIP-2 defines the format of the message, as shown in Figure 20.17. Part of the message, which we call entry, can be repeated as needed in a message. Each entry carries the information related to one line in the forwarding table of the router that sends the message.

Figure 20.17 RIP message format



RIP has two types of messages: request and response. A request message is sent by a router that has just come up or by a router that has some time-out entries. A request message can ask about specific entries or all entries. A response (or update) message can be either solicited or unsolicited. A solicited response message is sent only in answer to a request message. It contains information about the destination specified in the corresponding request message. An unsolicited response message, on the other hand, is sent periodically, every 30 seconds or when there is a change in the forwarding table.

RIP Algorithm

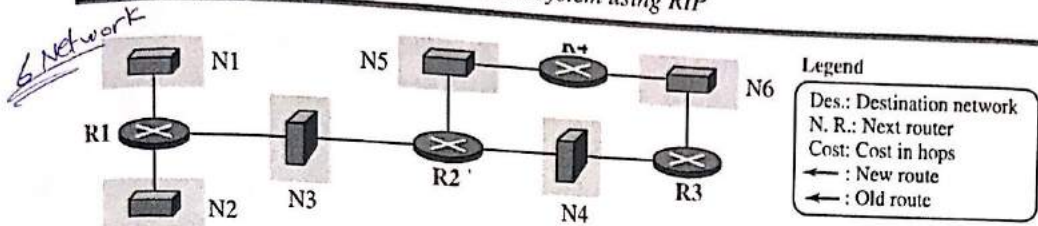
Distance vector

RIP implements the same algorithm as the distance-vector routing algorithm we discussed in the previous section. However, some changes need to be made to the algorithm to enable a router to update its forwarding table:

- ❑ Instead of sending only distance vectors, a router needs to send the whole contents of its forwarding table in a response message. *پورا فوروورڈنگ ٹیبل (next router) کو بھیجنا (Table)*
- ❑ The receiver adds one hop to each cost and changes the next router field to the address of the sending router. We call each route in the modified forwarding table the received route and each route in the old forwarding table the old route. The received router selects the old routes as the new ones except in the following three cases: *پہلے سے موجود*
 1. If the received route does not exist in the old forwarding table, it should be added to the route.
 2. If the cost of the received route is lower than the cost of the old one, the received route should be selected as the new one.
 3. If the cost of the received route is higher than the cost of the old one, but the value of the next router is the same in both routes, the received route should be selected as the new one. This is the case where the route was actually advertised.

پہلے سے موجود
node (S)
وہاں سے ہی الٹا
صوبہ کسی کی فہرست

Figure 20.18 Example of an autonomous system using RIP



Forwarding tables after all routers booted

R1			R2			R3			R4		
Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost
N1		1	N3		1	N4		1	N5		1
N2		1	N4		1	N5		1	N6		1
N3		1	N5		1						

تحديث
بيانات

Changes in the forwarding tables of R1, R3, and R4 after they receive a copy of R2's table

New R1			Old R1			R2 Seen by R1		
Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost
N1		1	N1		1	N1		1
N2		1	N2		1	N2		1
N3		1	N3		1	N3		1
N4	R2	2				N4	R2	2
N5	R2	2				N5	R2	2

New R3			Old R3			R2 Seen by R3		
Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost
N3	R2	2	N4		1	N3	R2	2
N4		1	N6		1	N4	R2	2
N5	R2	2				N5	R2	2
N6		1						

New R4			Old R4			R2 Seen by R4		
Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost
N3	R2	2	N5		1	N3	R2	2
N4	R2	2	N6		1	N4	R2	2
N5		1				N5	R2	2
N6		1				N6	R2	2

R2 is update
R1 is update
R3 is update
R4 is update

cost=2
لانه لو
جاء
Inode
ناتج عن ر2
R1

Forwarding tables for all routers after they have been stabilized

Final R1			Final R2			Final R3			Final R4		
Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost	Des.	N.R.	Cost
N1		1	N1	R1	2	N1	R2	3	N1	R2	3
N2		1	N2	R1	2	N2	R2	3	N2	R2	3
N3		1	N3		1	N3	R2	2	N3	R2	2
N4	R2	2	N4		1	N4		1	N4	R2	2
N5	R2	2	N5		1	N5	R2	2	N5		1
N6	R2	3	N6	R3	2	N6		1	N6		1

مصدر الـ
طريقه
مستقبل
cost=3

مصدر الـ
مستقبل
neighbor
فصلها بين
يوجد في السطوح

Robustness. As we said before, distance-vector routing is based on the concept that each router sends what it knows about the whole domain to its neighbors. This means that the calculation of the forwarding table depends on information received from immediate neighbors, which in turn receive their information from their own neighbors. If there is a failure or corruption in one router, the problem will be propagated to all routers and the forwarding in each router will be affected.

20.3.3 Open Shortest Path First (OSPF)

link-state Algo.

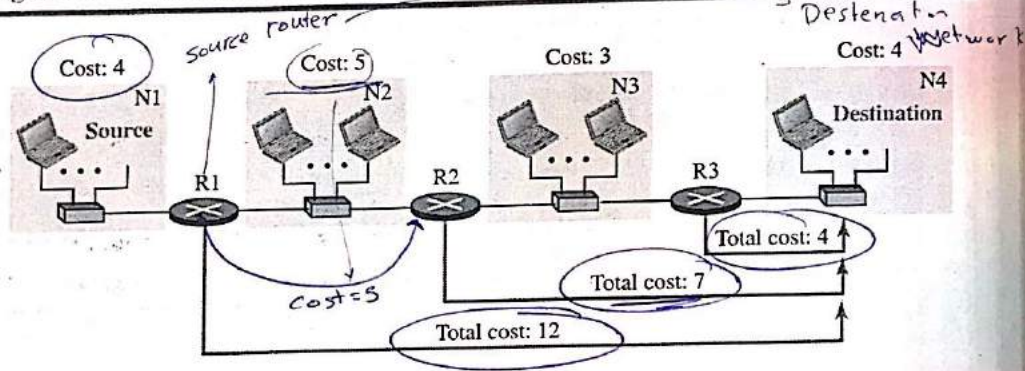
Open Shortest Path First (OSPF) is also an intradomain routing protocol like RIP, but it is based on the link-state routing protocol we described earlier in the chapter. OSPF is an *open* protocol, which means that the specification is a public document.

Metric

In OSPF, like RIP, the cost of reaching a destination from the host is calculated from the source router to the destination network. However, each link (network) can be assigned a weight based on the throughput, round-trip time, reliability, and so on. An administration can also decide to use the hop count as the cost. An interesting point about the cost in OSPF is that different service types (TOSs) can have different weights as the cost. Figure 20.19 shows the idea of the cost from a router to the destination host network. We can compare the figure with Figure 20.15 for the RIP.

Router → Host

Figure 20.19 Metric in OSPF



Forwarding Tables

Each OSPF router can create a forwarding table after finding the shortest-path tree between itself and the destination using Dijkstra's algorithm, described earlier in the chapter. Figure 20.20 shows the forwarding tables for the simple AS in Figure 20.19. Comparing the forwarding tables for the OSPF and RIP in the same AS, we find that the only difference is the cost values. In other words, if we use the hop count for OSPF, the tables will be exactly the same. The reason for this consistency is that both protocols use the shortest-path trees to define the best route from a source to a destination.

Areas

Compared with RIP, which is normally used in small ASs, OSPF was designed to be able to handle routing in a small or large autonomous system. However, the formation of shortest-path trees in OSPF requires that all routers flood the whole AS with their LSPs to create the global LSDB. Although this may not create a problem in a small AS, it may have created a huge volume of traffic in a large AS. To prevent this, the AS needs to be divided into small sections called areas. Each area acts as a small independent domain for flooding LSPs. In other words, OSPF uses another level of hierarchy in routing: the first level is the autonomous system, the second is the area.

(A.S) still update 1, 2, 3, 4, 5, 6 performance

AS (Large) 1, 2, 3, 4, 5, 6

Figure 20.20 Forwarding tables in OSPF

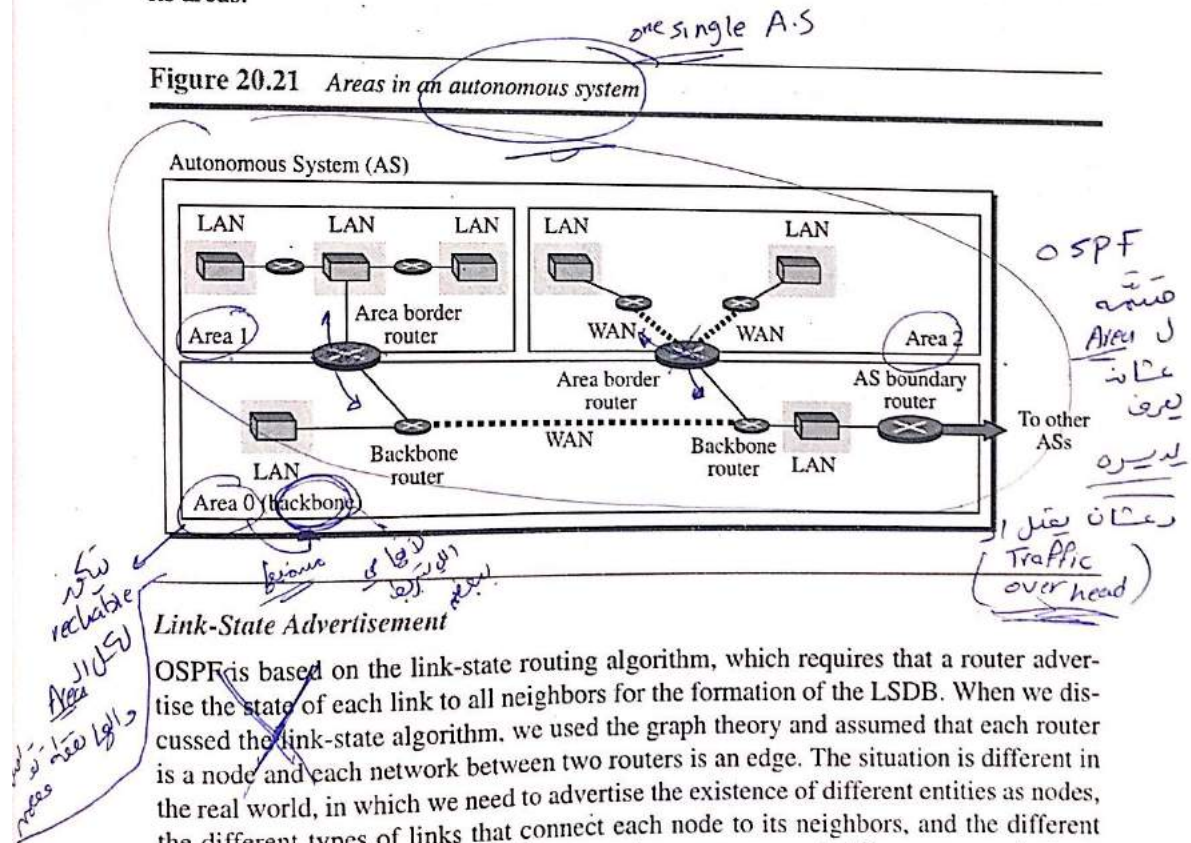
Destination network	Next router	Cost
N1	—	4
N2	—	5
N3	R2	8
N4	R2	12

Destination network	Next router	Cost
N1	R1	9
N2	—	5
N3	—	3
N4	R3	7

Destination network	Next router	Cost
N1	R2	12
N2	R2	8
N3	—	3
N4	—	4

However, each router in an area needs to know the information about the link states not only in its area but also in other areas. For this reason, one of the areas in the AS is designated as the backbone area, responsible for gluing the areas together. The routers in the backbone area are responsible for passing the information collected by each area to all other areas. In this way, a router in an area can receive all LSPs generated in other areas. For the purpose of communication, each area has an area identification. The area identification of the backbone is zero. Figure 20.21 shows an autonomous system and its areas.

Figure 20.21 Areas in an autonomous system



Link-State Advertisement

OSPF is based on the link-state routing algorithm, which requires that a router advertise the state of each link to all neighbors for the formation of the LSDB. When we discussed the link-state algorithm, we used the graph theory and assumed that each router is a node and each network between two routers is an edge. The situation is different in the real world, in which we need to advertise the existence of different entities as nodes, the different types of links that connect each node to its neighbors, and the different types of cost associated with each link. This means we need different types of advertisements, each capable of advertising different situations. We can have five types of

2 collected by the area to the backbone. As we discussed earlier, this type of information exchange is needed to glue the areas together.

- **Summary link to AS.** This is done by an AS router that advertises the summary links from other ASs to the backbone area of the current AS, information which later can be disseminated to the areas so that they will know about the networks in other ASs. The need for this type of information exchange is better understood when we discuss inter-AS routing (BGP).
- **External link.** This is also done by an AS router to announce the existence of a single network outside the AS to the backbone area to be disseminated into the areas.

OSPF Implementation

4. IPv4
layer 3

OSPF is implemented as a program in the network layer, using the service of the IP for propagation. An IP datagram that carries a message from OSPF sets the value of the protocol field to 89. This means that, although OSPF is a routing protocol to help IP to route its datagrams inside an AS, the OSPF messages are encapsulated inside datagrams. OSPF has gone through two versions: version 1 and version 2. Most implementations use version 2.

OSPF Messages

OSPF is a very complex protocol; it uses five different types of messages. In Figure 20.23, we first show the format of the OSPF common header (which is used in all messages) and the link-state general header (which is used in some messages). We then give the outlines of five message types used in OSPF. The *hello* message (type 1) is used by a router to introduce itself to the neighbors and announce all neighbors that it already knows. The *database description* message (type 2) is normally sent in response to the hello message to allow a newly joined router to acquire the full LSDB. The *link-state request* message (type 3) is sent by a router that needs information about a specific LS. The *link-state update* message (type 4) is the main OSPF message used for building the LSDB. This message, in fact, has five different versions (router link, network link, summary link to network, summary link to AS border router, and external link), as we discussed before. The *link-state acknowledgment* message (type 5) is used to create reliability in OSPF; each router that receives a link-state update message needs to acknowledge it.

Authentication

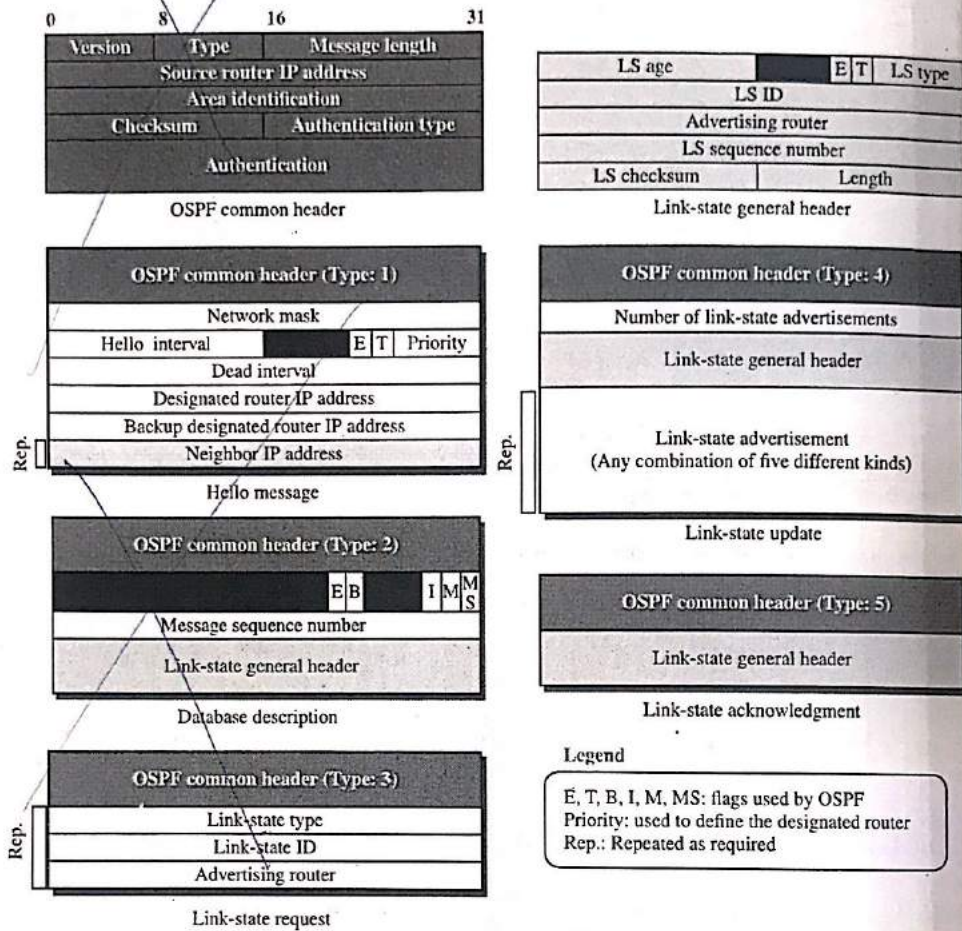
As Figure 20.23 shows, the OSPF common header has the provision for authentication of the message sender. As we will discuss in Chapters 31 and 32, this prevents a malicious entity from sending OSPF messages to a router and causing the router to become part of the routing system to which it actually does not belong.

OSPF Algorithm

OSPF implements the link-state routing algorithm we discussed in the previous section. However, some changes and augmentations need to be added to the algorithm:

- After each router has created the shortest-path tree, the algorithm needs to use it to create the corresponding routing algorithm.

Figure 20.23 OSPF message formats



- ❑ The algorithm needs to be augmented to handle sending and receiving all five types of messages.

Performance

Before ending this section, let us briefly discuss the performance of OSPF:

- ❑ **Update Messages.** The link-state messages in OSPF have a somewhat complex format. They also are flooded to the whole area. If the area is large, these messages may create heavy traffic and use a lot of bandwidth.
- ❑ **Convergence of Forwarding Tables.** When the flooding of LSPs is completed, each router can create its own shortest-path tree and forwarding table; convergence is fairly quick. However, each router needs to run Dijkstra's algorithm, which may take some time.

Handwritten notes in Arabic: "load ↑", "فlooded في كل المساحة", "لا يمكن إرسالها في كل المساحة", "Area J", "فlooded في كل المساحة", "30", "50".

Handwritten notes: "CONVERGENCE", "↑", "↓", "50", "30".

- Robustness. The OSPF protocol is more robust than RIP because, after receiving the completed LSDB, each router is independent and does not depend on other routers in the area. Corruption or failure in one router does not affect other routers as seriously as in RIP.

autonomous system

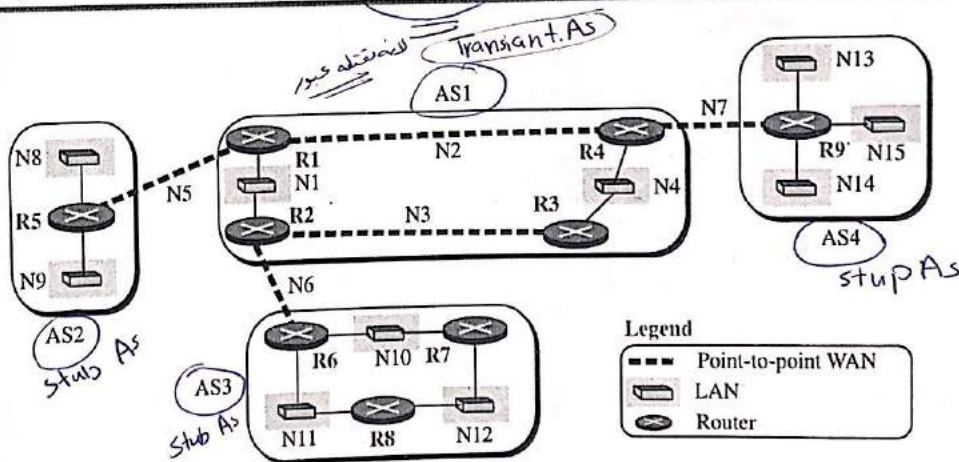
20.3.4 Border Gateway Protocol Version 4 (BGP4)

The Border Gateway Protocol version 4 (BGP4) is the only interdomain routing protocol used in the Internet today. BGP4 is based on the path-vector algorithm we described before, but it is tailored to provide information about the reachability of networks in the Internet.

Introduction

BGP, and in particular BGP4, is a complex protocol. In this section, we introduce the basics of BGP and its relationship with intradomain routing protocols (RIP or OSPF). Figure 20.24 shows an example of an internet with four autonomous systems. AS2, AS3, and AS4 are *stub* autonomous systems; AS1 is a transient one. In our example, data exchange between AS2, AS3, and AS4 should pass through AS1.

Figure 20.24 A sample internet with four ASs



Each autonomous system in this figure uses one of the two common intradomain protocols, RIP or OSPF. Each router in each AS knows how to reach a network that is in its own AS, but it does not know how to reach a network in another AS.

To enable each router to route a packet to any network in the internet, we first install a variation of BGP4, called external BGP (eBGP), on each border router (the one at the edge of each AS which is connected to a router at another AS). We then install the second variation of BGP, called internal BGP (iBGP), on all routers. This means that the border routers will be running three routing protocols (intradomain, eBGP, and iBGP), but other routers are running two protocols (intradomain and iBGP). We discuss the effect of each BGP variation separately.

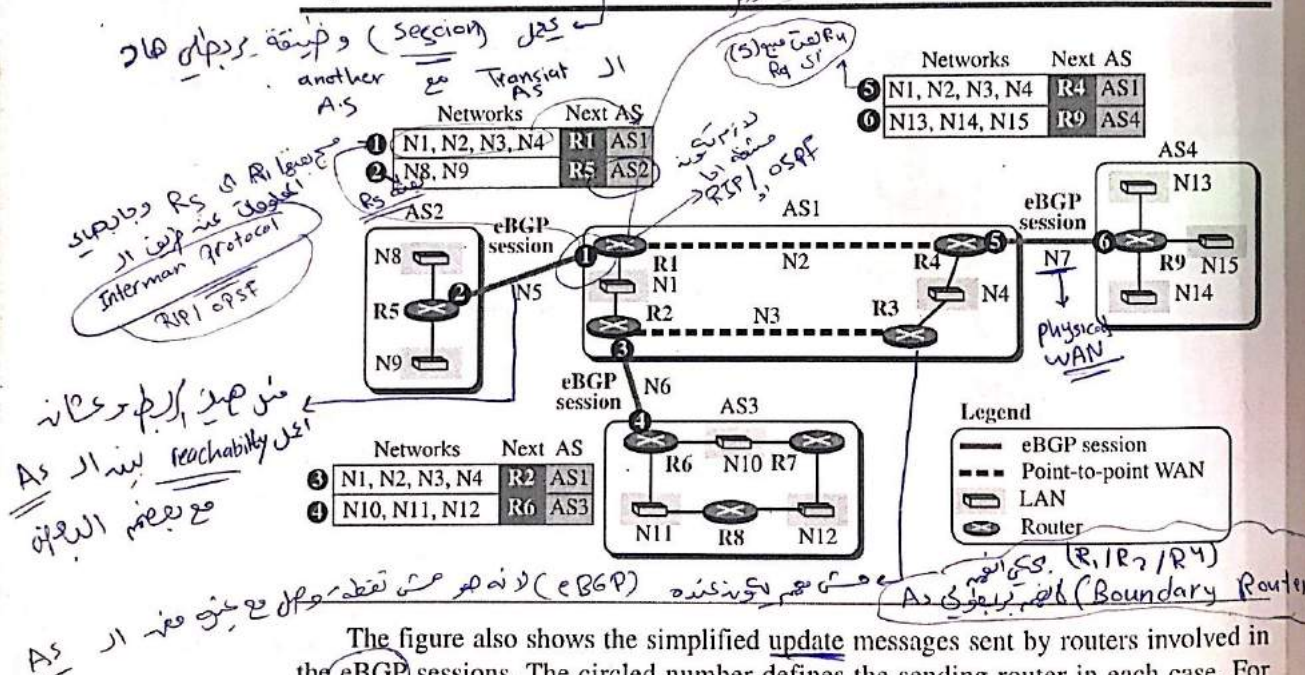
Operation of External BGP (eBGP)

We can say that BGP is a kind of point-to-point protocol. When the software is installed on two routers, they try to create a TCP connection using the well-known port 179. In other words, a pair of client and server processes continuously communicate with each other to exchange messages. The two routers that run the BGP processes are called **BGP peers** or **BGP speakers**. We discuss different types of messages exchanged between two peers, but for the moment we are interested in only the update messages (discussed later) that announce reachability of networks in each AS.

The eBGP variation of BGP allows two physically connected border routers in two different ASs to form pairs of eBGP speakers and exchange messages. The routers that are eligible in our example in Figure 20.24 form three pairs: R1-R5, R2-R6, and R4-R9. The connection between these pairs is established over three physical WANs (N5, N6, and N7). However, there is a need for a logical TCP connection to be created over the physical connection to make the exchange of information possible. Each logical connection in BGP parlance is referred to as a **session**. This means that we need three sessions in our example, as shown in Figure 20.25.

Router (R1, R5)
 البروتوكول الذي يشغله
 بين الروتين الذي يشغله
 (BGP Protocol)

Figure 20.25 eBGP operation



The figure also shows the simplified update messages sent by routers involved in the eBGP sessions. The circled number defines the sending router in each case. For example, message number 1 is sent by router R1 and tells router R5 that N1, N2, N3, and N4 can be reached through router R1 (R1 gets this information from the corresponding intradomain forwarding table). Router R5 can now add these pieces of information at the end of its forwarding table. When R5 receives any packet destined for these four networks, it can use its forwarding table and find that the next router is R1.

The reader may have noticed that the messages exchanged during three eBGP sessions help some routers know how to route packets to some networks in the internet, but

the reachability information is not complete. There are two problems that need to be addressed:

1. Some border routers do not know how to route a packet destined for nonneighbor ASs. For example, R5 does not know how to route packets destined for networks in AS3 and AS4. Routers R6 and R9 are in the same situation as R5: R6 does not know about networks in AS2 and AS4; R9 does not know about networks in AS2 and AS3.
2. None of the nonborder routers know how to route a packet destined for any networks in other ASs.

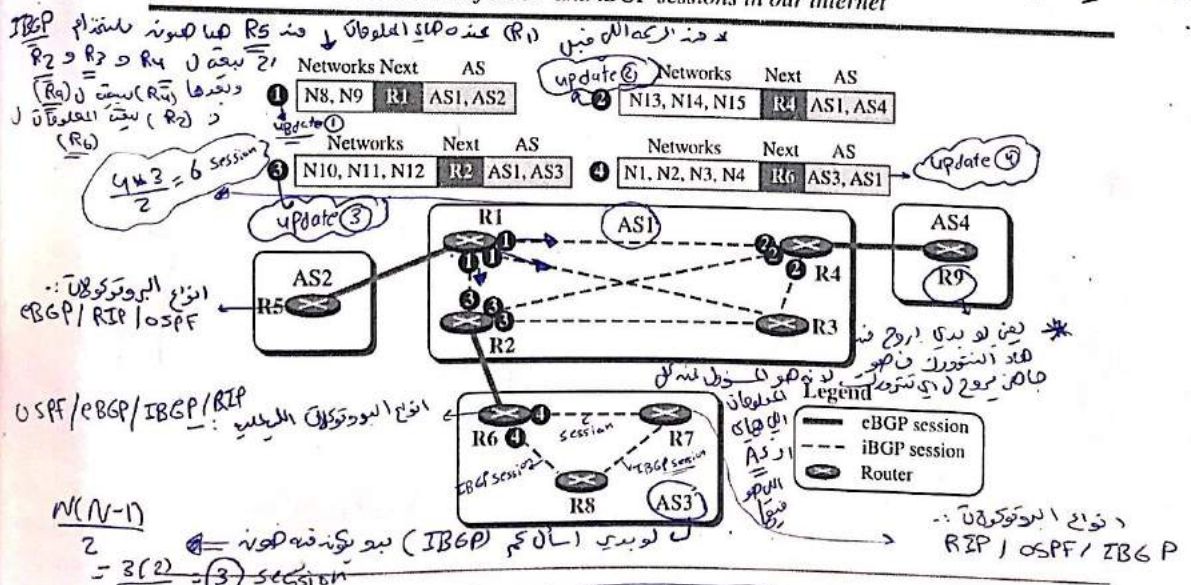
To address the above two problems, we need to allow all pairs of routers (border or nonborder) to run the second variation of the BGP protocol, iBGP.

Operation of Internal BGP (iBGP) :- *يا اكونه بدى لومى نتودون لى ستودوك الازى*
 The iBGP protocol is similar to the eBGP protocol in that it uses the service of TCP on the well-known port 179, but it creates a session between any possible pair of routers inside an autonomous system. However, some points should be made clear. First, if an AS has only one router, there cannot be an iBGP session. For example, we cannot create an iBGP session inside AS2 or AS4 in our internet. Second, if there are n routers in an autonomous system, there should be $[n \times (n - 1) / 2]$ iBGP sessions in that autonomous system (a fully connected mesh) to prevent loops in the system. In other words, each router needs to advertise its own reachability to the peer in the session instead of flooding what it receives from another peer in another session. Figure 20.26 shows the combination of eBGP and iBGP sessions in our internet.

يعنى نوهل معلومات كنه other non Border Router

$$\frac{4 \times 3}{2} = 6$$

Figure 20.26 Combination of eBGP and iBGP sessions in our internet



$$\frac{n(n-1)}{2} = \frac{3(2)}{2} = 3 \text{ session}$$

Note that we have not shown the physical networks inside ASs because a session is made on an overlay network (TCP connection), possibly spanning more than one physical network as determined by the route dictated by intradomain routing protocol. Also note that in this stage only four messages are exchanged. The first message (numbered 1) is sent by R1 announcing that networks N8 and N9 are reachable through the

path AS1-AS2, but the next router is R1. This message is sent, through separate sessions, to R2, R3, and R4. Routers R2, R4, and R6 do the same thing but send different messages to different destinations. The interesting point is that, at this stage, R3, R7, and R8 create sessions with their peers, but they actually have no message to send.

The updating process does not stop here. For example, after R1 receives the update message from R2, it combines the reachability information about AS3 with the reachability information it already knows about AS1 and sends a new update message to R5. Now R5 knows how to reach networks in AS1 and AS3. The process continues when R1 receives the update message from R4. The point is that we need to make certain that at a point in time there are no changes in the previous updates and that all information is propagated through all ASs. At this time, each router combines the information received from eBGP and iBGP and creates what we may call a path table after applying the criteria for finding the best path, including routing policies that we discuss later. To demonstrate, we show the path tables in Figure 20.27 for the routers in Figure 20.24. For example, router R1 now knows that any packet destined for networks N8 or N9 should go through AS1 and AS2 and the next router to deliver the packet to is router R5. Similarly, router R4 knows that any packet destined for networks N10, N11, or N12 should go through AS1 and AS3 and the next router to deliver this packet to is router R1, and so on.

Figure 20.27 Finalized BGP path tables

Networks	Next	Path	Networks	Next	Path	Networks	Next	Path
N8, N9	R5	AS1, AS2	N8, N9	R1	AS1, AS2	N8, N9	R2	AS1, AS2
N10, N11, N12	R2	AS1, AS3	N10, N11, N12	R6	AS1, AS3	N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4	N13, N14, N15	R1	AS1, AS4	N13, N14, N15	R4	AS1, AS4
Path table for R1			Path table for R2			Path table for R3		
N8, N9	R1	AS1, AS2	N1, N2, N3, N4	R1	AS2, AS1	N1, N2, N3, N4	R2	AS3, AS1
N10, N11, N12	R1	AS1, AS3	N10, N11, N12	R1	AS2, AS1, AS3	N8, N9	R2	AS3, AS1, AS2
N13, N14, N15	R9	AS1, AS4	N13, N14, N15	R1	AS2, AS1, AS4	N13, N14, N15	R2	AS3, AS1, AS4
Path table for R4			Path table for R5			Path table for R6		
N1, N2, N3, N4	R6	AS3, AS1	N1, N2, N3, N4	R6	AS3, AS1	N1, N2, N3, N4	R4	AS4, AS1
N8, N9	R6	AS3, AS1, AS2	N8, N9	R6	AS3, AS1, AS2	N8, N9	R4	AS4, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4	N13, N14, N15	R6	AS3, AS1, AS4	N10, N11, N12	R4	AS4, AS1, AS3
Path table for R7			Path table for R8			Path table for R9		

(R1+R2+R3+R4)
هذه المعلومات لانه
فهمه ان (AS) خط
التي عنده ان
Next
عن طريق
IBGP
معاينة عن طريق
ERBGP
(R1) -> (R1)
معاينة عن طريق
IBGP

Injection of Information into Intradomain Routing

The role of an interdomain routing protocol such as BGP is to help the routers inside the AS to augment their routing information. In other words, the path tables collected and organized by BGP are not used, per se, for routing packets; they are injected into intradomain forwarding tables (RIP or OSPF) for routing packets. This can be done in several ways depending on the type of AS.

In the case of a stub AS, the only area border router adds a default entry at the end of its forwarding table and defines the next router to be the speaker router at the end of the eBGP connection. In Figure 20.24, R5 in AS2 defines R1 as the default router for

all networks other than N8 and N9. The situation is the same for router R9 in AS4 with the default router to be R4. In AS3, R6 set its default router to be R2, but R7 and R8 set their default router to be R6. These settings are in accordance with the path tables we describe in Figure 20.27 for these routers. In other words, the path tables are injected into intradomain forwarding tables by adding only one default entry.

In the case of a transient AS, the situation is more complicated. R1 in AS1 needs to inject the whole contents of the path table for R1 in Figure 20.27 into its intradomain forwarding table. The situation is the same for R2, R3, and R4.

One issue to be resolved is the cost value. We know that RIP and OSPF use different metrics. One solution, which is very common, is to set the cost to the foreign networks at the same cost value as to reach the first AS in the path. For example, the cost for R5 to reach all networks in other ASs is the cost to reach N5. The cost for R1 to reach networks N10 to N12 is the cost to reach N6, and so on. The cost is taken from the intradomain forwarding tables (RIP or OSPF).

Figure 20.28 shows the interdomain forwarding tables. For simplicity, we assume that all ASs are using RIP as the intradomain routing protocol. The shaded areas are the augmentation injected by the BGP protocol; the default destinations are indicated as zero.

Figure 20.28 Forwarding tables after injection from BGP

Handwritten notes in Arabic: "R2 الى R1 (2 hops) ...", "AS 1 (reliability)", "Final Destination", "AS 1 (reliability)", "R5", "R6", "R2", "R7", "R6", "Default", "R6".

Des.	Next	Cost
N1	—	1
N4	R4	2
N8	R5	1
N9	R5	1
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R1

Des.	Next	Cost
N1	—	1
N4	R3	2
N8	R1	2
N9	R1	2
N10	R6	1
N11	R6	1
N12	R6	1
N13	R3	3
N14	R3	3
N15	R3	3

Table for R2

Des.	Next	Cost
N1	R2	2
N4	—	1
N8	R2	3
N9	R2	3
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R3

Des.	Next	Cost
N1	R1	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R3	3
N12	R3	3
N13	R9	1
N14	R9	1
N15	R9	1

Table for R4

Des.	Next	Cost
N8	—	1
N9	—	1
0	R1	1

Table for R5

Des.	Next	Cost
N10	—	1
N11	—	1
N12	R7	2
0	R2	1

Table for R6

Des.	Next	Cost
N10	—	1
N11	R6	2
N12	—	1
0	R6	2

Table for R7

Des.	Next	Cost
N10	—	1
N11	—	1
N12	—	1
0	R6	2

Table for R8

Des.	Next	Cost
N13	—	1
N14	—	1
N15	—	1
0	R4	1

Table for R9

Address Aggregation

The reader may have realized that intradomain forwarding tables obtained with the help of the BGP4 protocols may become huge in the case of the global Internet because many destination networks may be included in a forwarding table. Fortunately, BGP4 uses the prefixes as destination identifiers and allows the aggregation of these prefixes, as we discussed in Chapter 18. For example, prefixes 14.18.20.0/26, 14.18.20.64/26, 14.18.20.128/26, and 14.18.20.192/26, can be combined into 14.18.20.0/24 if all four