UNIT III MEDIUM ACCESS SUB LAYER AND TRANSPORT PROTOCOL 10 hrs.

The Medium Access Sub Layer : The channel allocation problem, Multiple access Protocols, Ethernet, Wireless LANs, Broadband Wireless, Bluetooth, Data Link Layer Switching.
The Network Layer: Network Layer Design Issues, Routing Algorithms, Congestion Control Algorithms, Quality of Service.
The Transport Protocol: The Transport Service, Elements of transport protocol, Performance Issues.

The Media Access Control (MAC) data communication Networks protocol sub-layer, also known as the Medium Access Control, is a sub-layer of the data link layer specified in the seven-layer OSI model. The medium access layer was made necessary by systems that share a common communications medium. Typically these are local area networks. The MAC layer is the "low" part of the second OSI layer, the layer of the "data link". In fact, the IEEE divided this layer into two layers "above" is the control layer the logical connection (Logical Link Control, LLC) and "down" the control layer The medium access (MAC).

The LLC layer is standardized by the IEEE as the 802.2 since the beginning 1980 Its purpose is to allow level 3 network protocols (for eg IP) to be based on a single layer (the LLC layer) regardless underlying protocol used, including WiFi, Ethernet or Token Ring, for example. All WiFi data packets so carry a pack LLC, which contains itself packets from the upper network layers. The header of a packet LLC indicates the type of layer 3 protocol in it: most of the time, it is IP protocol, but it could be another protocol, such as IPX (Internet Packet Exchange) for example. Thanks to the LLC layer, it is possible to have at the same time, on the same network, multiple Layer 3 protocols.

In LAN nodes uses the same communication channel for transmission. The MAC sub-layer has two primary responsibilities:

Data encapsulation, including_frame assembly before transmission, and frame parsing/error detection during and after reception. Media access control, including initiation of frame transmission and recovery from transmission failure.

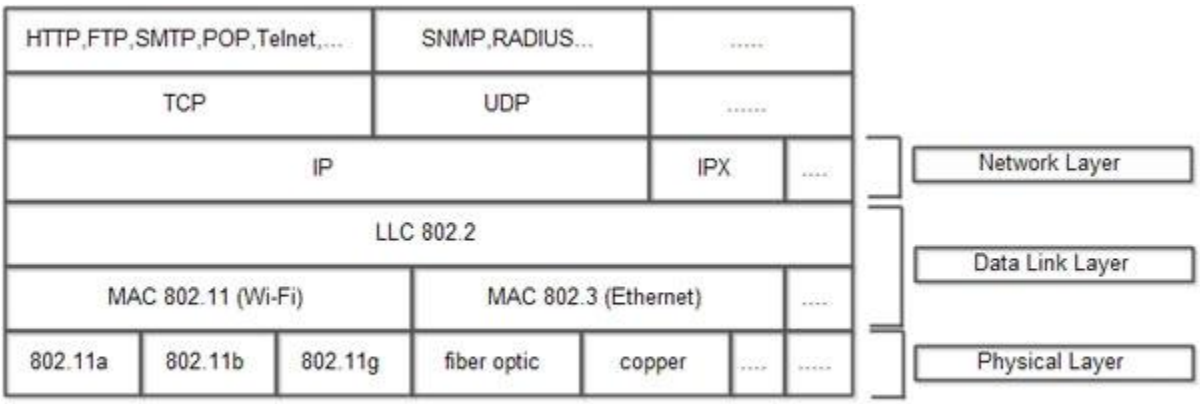


Figure 3.1. MAC layer protocol stack

3.1. The channel allocation problem

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is Frequency Division Multiplexing (FDM). If there are *N* users, the bandwidth is divided into *N* equal-sized portions each user being assigned one portion. Since each user has a private frequency band, there is no interference between users. When there is only a small and constant number of users, each of which has a heavy (buffered) load of traffic (e.g., carriers' switching offices), FDM is a simple and efficient allocation mechanism.

However, when the number of senders is large and continuously varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into *N* regions and fewer than *N* users are currently interested in communicating, a large piece of valuable spectrum will be wasted. If more than *N* users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

$$T_{\text{FDM}} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \tag{3.1}$$

$$T = \frac{1}{\mu C - \lambda} \tag{3.2}$$

However, even assuming that the number of users could somehow be held constant at *N*, dividing the single available channel into static subchannels is inherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They are not using it, and no one else is allowed to use it either. Furthermore, in most computer systems, data traffic is extremely bursty (peak traffic to mean traffic ratios of 1000:1 are common). Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen from a simple queueing theory calculation. Let us start with the mean time delay, T, for a channel of capacity C bps, with an arrival rate of λ frames/sec, each frame having a length drawn from an exponential probability density function with mean 1/μ bits/frame. With these parameters the arrival rate is λ frames/sec and the service rate is μC frames/sec. From queueing theory it can be shown that for Poisson arrival and service times,

For example, if C is 100 Mbps, the mean frame length, 1/μ, is 10,000 bits, and the frame arrival rate, λ, is 5000 frames/sec, then T = 200 μsec. Note that if we ignored the queueing delay and just asked how long it takes to send a 10,000 bit frame on a 100-Mbps network, we would get the (incorrect) answer of 100 μsec. That result only holds when there is no contention for the channel.

Now let us divide the single channel into N independent subchannels, each with capacity C/N bps. The mean input rate on each of the subchannels will now be $λ/N$. Recomputing T we get

The mean delay using FDM is N times worse than if all the frames were somehow magically arranged orderly in a big central queue.

Precisely the same arguments that apply to FDM also apply to time division multiplexing (TDM). Each user is statically allocated every Nth time slot. If a user does not use the allocated slot, it just lies fallow. The same holds if we split up the networks physically. Using our previous example again, if we were to replace the 100-Mbps network with 10 networks of 10 Mbps each and statically allocate each user to one of them, the mean delay would jump from 200 μsec to 2 msec.

Since none of the traditional static channel allocation methods work well with bursty traffic, we will now explore dynamic methods.

Dynamic Channel Allocation in LANs and MANs

Before we get into the first of the many channel allocation methods to be discussed in this chapter, it is worthwhile carefully formulating the allocation problem. Underlying all the work done in this area are five key assumptions, described below.

Station Model. The model consists of N independent **stations** (e.g., computers, telephones, or personal communicators), each with a program or user that generates frames for transmission. Stations are sometimes called **terminals**. The probability of a frame being generated in an interval of length Δt is λΔt, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

Single Channel Assumption. A single channel is available for all communication. All stations can transmit on it and all can receive from it. As far as the hardware is concerned, all stations are equivalent, although protocol software may assign priorities to them.

Collision Assumption. If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a **collision**. All stations can detect collisions. A collided frame must be transmitted again later. There are no errors other than those generated by collisions.

4a. Continuous Time. Frame transmission can begin at any instant. There is no master clock dividing time into discrete intervals.

4b. Slotted Time. Time is divided into discrete intervals (slots). Frame transmissions always begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.

5a. Carrier Sense. Stations can tell if the channel is in use before trying to use it. If the channel is sensed as busy, no station will attempt to use it until it goes idle.

5b. No Carrier Sense. Stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

Some discussion of these assumptions is in order. The first one says that stations are independent and that work is generated at a constant rate. It also implicitly assumes that each station only has one program or user, so while the station is blocked, no new work is generated. More sophisticated models allow multi programmed stations that can generate work while a station is blocked, but the analysis of these stations is much more complex.

The single channel assumption is the heart of the model. There are no external ways to communicate. Stations cannot raise their hands to request that the teacher call on them.

The collision assumption is also basic, although in some systems (notably spread spectrum), this assumption is relaxed, with surprising results. Also, some LANs, such as token rings, pass a special token from station to station, possession of which allows the current holder to transmit a frame. But in the coming sections we will stick to the single channel with contention and collisions model.

Two alternative assumptions about time are possible. Either it is continuous (4a) or it is slotted (4b). Some systems use one and some systems use the other, so we will discuss and analyze both. For a given system, only one of them holds.

Similarly, a network can either have carrier sensing (5a) or not have it (5b). LANs generally have carrier sense. However, wireless networks cannot use it effectively because not every station may be within radio range of every other station. Stations on wired carrier sense networks can terminate their transmission prematurely if they discover that it is colliding with another transmission. Collision detection is rarely done on wireless networks, for engineering reasons. Note that the word "carrier" in this sense refers to an electrical signal on the cable and has nothing to do with the common carriers (e.g., telephone companies) that date back to the Pony Express days.

3.2. MULTIPLE ACCESS PROTOCOLS

Following Protocols are used by Medium Access Layer:

ALOHA: ALOHA is a system for coordinating and arbitrating access to a shared communication channel. It was developed in the 1970s at the University of Hawaii. The original system used terrestrial radio broadcasting, but the system has been implemented in satellite communication

systems. A shared communication system like ALOHA requires a method of handling collisions that occur when two or more systems attempt to transmit on the channel at the same time.

PURE ALOHA

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. However, due to the feedback property of broadcasting, a sender can always find out whether its frame was destroyed by listening to the channel, the same way other users do. With a LAN, the feedback is immediate; with a satellite, there is a delay of 270 msec before the sender knows if the transmission was successful. If listening while transmitting is not possible for some reason, acknowledgements are needed. If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that can lead to conflicts are widely known as **contention** systems.

In the ALOHA system, a node transmits whenever data is available to send. If another node transmits at the same time, a collision occurs, and the frames that were transmitted are lost. However, a node can listen to broadcasts on the medium, even its own, and determine whether the frames were transmitted.

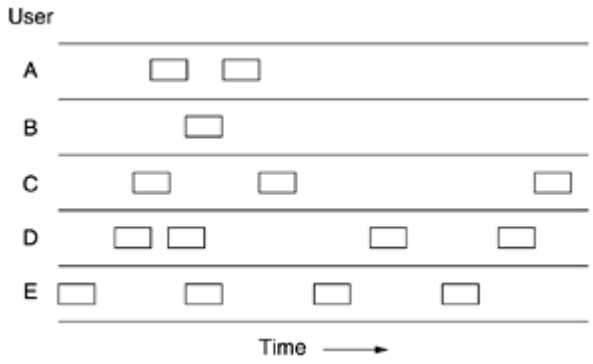


Figure 3.2. Frames are transmitted at completely arbitrary times

SLOTTED ALOHA

In 1972, Roberts published a method for doubling the capacity of an ALOHA system (Roberts, 1972). His proposal was to divide time into discrete intervals, each interval corresponding to one frame. This approach requires the users to agree on slot boundaries. One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

In Roberts' method, which has come to be known as slotted ALOHA, in contrast to Abramson's pure ALOHA, a computer is not permitted to send whenever a carriage return is typed. Instead, it is required to wait for the beginning of the next slot. Thus, the continuous pure

ALOHA is turned into a discrete one. Since the vulnerable period is now halved, the probability of no other traffic during the same slot as our test frame is e-G which leads to

$$S = Ge^{-G} \tag{3.3}$$

Carrier Sensed Multiple Accesses (CSMA): CSMA is a network access method used on shared network topologies such as Ethernet to control access to the network. Devices attached to the network cable listen (carrier sense) before transmitting. If the channel is in use, devices wait before transmitting. MA (Multiple Access) indicates that many devices can connect to and share the same network. All devices have equal access to use the network when it is clear.

Even though devices attempt to sense whether the network is in use, there is a good chance that two stations will attempt to access it at the same time. On large networks, the transmission time between one end of the cable and another is enough that one station may access the cable even though another has already just accessed it. There are two methods for avoiding these so-called collisions, listed here:

CSMA/CD (Carrier Sense Multiple Access/Collision Detection): CD (collision detection) defines what happens when two devices sense a clear channel, and then attempt to transmit at the same time. A collision occurs, and both devices stop transmission, wait for a random amount of time, and then retransmit. This is the technique used to access the 802.3 Ethernet network channel.

This method handles collisions as they occur, but if the bus is constantly busy, collisions can occur so often that performance drops drastically. It is estimated that network traffic must be less than 40 percent of the bus capacity for the network to operate efficiently. If distances are long, time lags occur that may result in inappropriate carrier sensing, and hence collisions.

CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance): In CA collision avoidance), collisions area voided because each node signals its intent to transmit before actually doing so. This method is not popular because it requires excessive overhead that reduces performance.

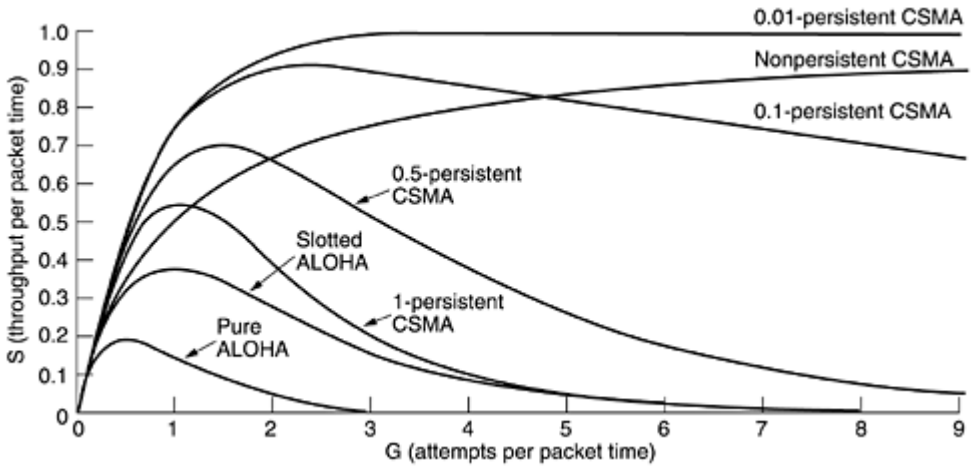


Figure 3.3. Performance comparison of various MAC protocols

CSMA WITH COLLISION DETECTION

Persistent and non-persistent CSMA protocols are clearly an improvement over ALOHA because they ensure that no station begins to transmit when it senses the channel busy. Another improvement is for stations to abort their transmissions as soon as they detect a collision. In other words, if two stations sense the channel to be idle and begin transmitting simultaneously, they will both detect the collision almost immediately. Rather than finish transmitting their frames, which are irretrievably garbled anyway, they should abruptly stop transmitting as soon as the collision is detected. Quickly terminating damaged frames saves time and bandwidth. This protocol, known as **CSMA/CD** (**CSMA with Collision Detection**) is widely used on LANs in the MAC sublayer. In particular, it is the basis of the popular Ethernet LAN, so it is worth devoting some time to looking at it in detail.

CSMA/CD, as well as many other LAN protocols, uses the conceptual model. At the point marked t_0, a station has finished transmitting its frame. Any other station having a frame to send may now attempt to do so. If two or more stations decide to transmit simultaneously, there will be a collision. Collisions can be detected by looking at the power or pulse width of the received signal and comparing it to the transmitted signal.

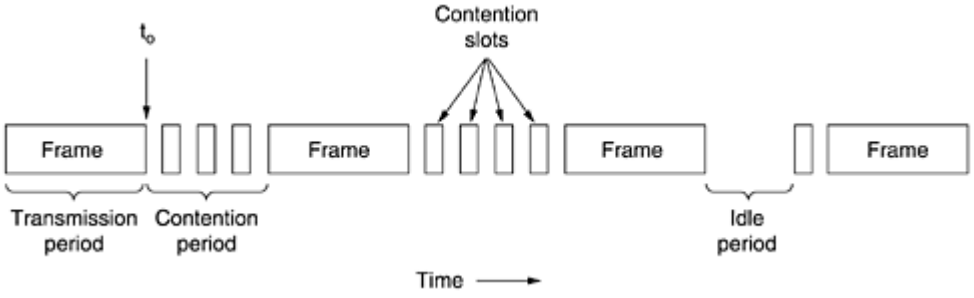


Figure 3.4. CSMA/CD can be in one of three states: contention, transmission, or idle**.**

COLLISION-FREE PROTOCOLS

Although collisions do not occur with CSMA/CD once a station has unambiguously captured the channel, they can still occur during the contention period. These collisions adversely affect the system performance, especially when the cable is long (i.e., large τ) and the frames are short. And CSMA/CD is not universally applicable. In this section, we will examine some protocols that resolve the contention for the channel without any collisions at all, not even during the contention period. Most of these are not currently used in major systems, but in a rapidly changing field, having some protocols with excellent properties available for future systems is often a good thing.

In the protocols to be described, we assume that there are exactly *N* stations, each with a unique address from 0 to *N* - 1 "wired" into it. It does not matter that some stations may be inactive part of the time. We also assume that propagation delay is negligible.

A BIT-MAP PROTOCOL

In our first collision-free protocol, the **basic bit-map method**, each contention period consists of exactly *N* slots. If station 0 has a frame to send, it transmits a 1 bit during the zeroth slot. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1 during slot 1, but only if it has a frame queued. In general, station *j* may announce that it has a frame to send by inserting a 1 bit into slot *j.* After all *N* slots have passed by, each station has complete knowledge of which stations wish to transmit. At that point, they begin transmitting in numerical order

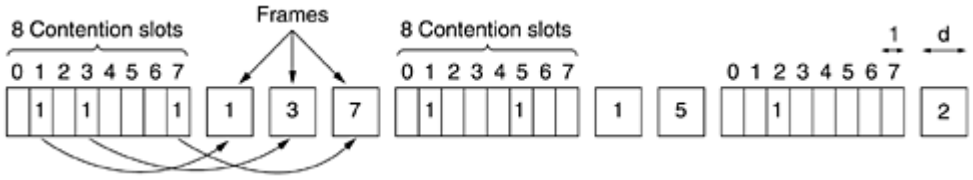


Figure 3.5. Basic bit-map protocol

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another *N* bit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again. Protocols like this in which the desire to transmit is broadcast before the actual transmission are called **reservation protocols**.

WAVELENGTH DIVISION MULTIPLE ACCESS PROTOCOLS

A different approach to channel allocation is to divide the channel into subchannels using FDM, TDM, or both, and dynamically allocate them as needed. Schemes like this are commonly used on fiber optic LANs to permit different conversations to use different wavelengths (i.e., frequencies) at the same time. In this section we will examine one such protocol (Humblet et al., 1992).

A simple way to build an all-optical LAN is to use a passive star coupler. In effect, two fibers from each station are fused to a glass cylinder. One fiber is for output to the cylinder and one is for input from the cylinder. Light output by any station illuminates the cylinder and can be detected by all the other stations. Passive stars can handle hundreds of stations.

To allow multiple transmissions at the same time, the spectrum is divided into channels (wavelength bands). In this protocol, **WDMA** (**Wavelength Division Multiple Access**), each station is assigned two channels. A narrow channel is provided as a control channel to signal the station, and a wide channel is provided so the station can output data frames.

Each channel is divided into groups of time slots, as shown in. Let us call the number of slots in the control channel *m* and the number of slots in the data channel *n* + 1, where *n* of these are for data and the last one is used by the station to report on its status (mainly, which slots on both channels are free). On both channels, the sequence of slots repeats endlessly, with slot 0 being marked in a special way so latecomers can detect it. All channels are synchronized by a single global clock.

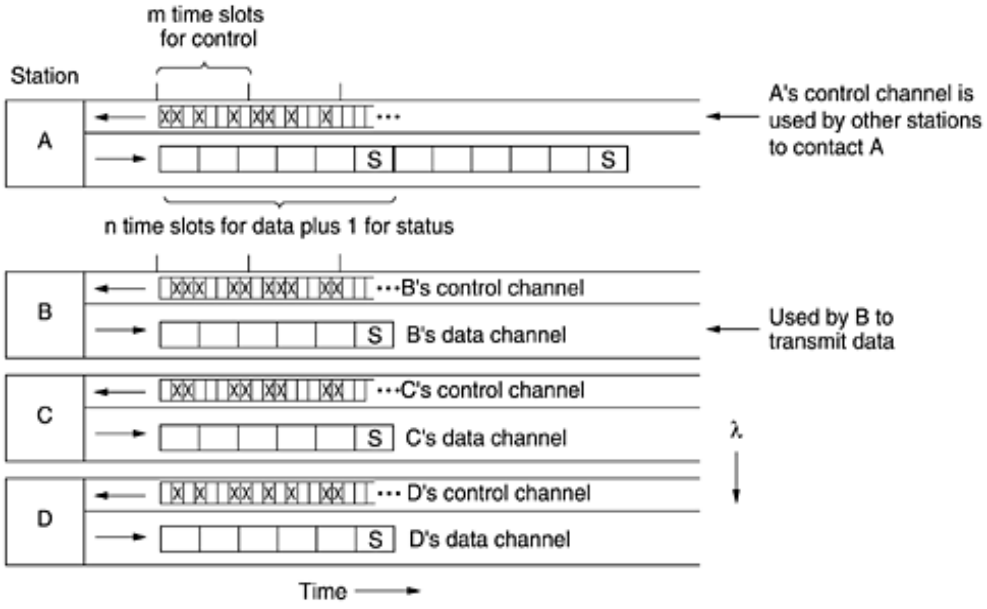


Figure 3.6. Wavelength division multiple access

3.3. ETHERNET:

- **IEEE 802.3 Local Area Network (LAN) Protocols:** Ethernet protocols refer to the family of local-area network (LAN) covered by the IEEE 802.3. In the Ethernet standard, there are two modes of operation: half-duplex and full-duplex modes. In the half duplex mode, data are transmitted using the popular Carrier-Sense Multiple Access/Collision Detection (CSMA/CD) protocol on a shared medium.

- The main disadvantages of the half-duplex are the efficiency and distance limitation, in which the link distances, is limited by the minimum MAC frame size. This restriction reduces the efficiency drastically for high-rate transmission. Therefore, the carrier extension technique is used to ensure the minimum frame size of 512 bytes in Gigabit Ethernet to achieve a reasonable link distance. Four data rates are currently defined for operation over optical fiber and twisted-pair cables :

10 Mbps - 10Base-T Ethernet (IEEE 802.3)

100 Mbps - Fast Ethernet (IEEE 802.3u)

1000 Mbps - Gigabit Ethernet (IEEE 802.3z)

10-Gigabit - 10 Gbps Ethernet (IEEE 802.3ae).

The **Ethernet System** consists of three basic elements:

(1) The physical medium used to carry Ethernet signals between computers,

(2) a set of medium access control rules embedded in each Ethernet interface that allow multiple computers to fairly arbitrate access to the shared Ethernet channel, and

(3) an Ethernet frame that consists of a standardized set of bits used to carry data over the system.

As with all IEEE 802 protocols, the ISO data link layer is divided into two IEEE 802 sub-layers, the Media Access Control (MAC) sub-layer and the MAC-client sub-layer. The IEEE 802.3 physical layer corresponds to the ISO physical layer.

Each Ethernet-equipped computer operates independently of all other stations on the network: there is no central controller. All stations attached to an Ethernet are connected to a shared signaling system, also called the medium. To send data a station first listens to the channel, and when the channel is idle the station transmits its data in the form of an Ethernet frame, or packet.

After each frame transmission, all stations on the network must contend equally for the next frame transmission opportunity. Access to the shared channel is determined by the medium access control (MAC) mechanism embedded in the Ethernet interface located in each station. The medium access control mechanism is based on a system called Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

As each Ethernet frame is sent onto the shared signal channel, all Ethernet interfaces look at the destination address. If the destination address of the frame matches with the interface address, the frame will be read entirely and be delivered to the networking software running on that computer. All other network interfaces will stop reading the frame when they discover that the destination address does not match their own address.

IEEE 802.4 Token Bus: In token bus network station must have possession of a token before it can transmit on the network. The IEEE 802.4 Committee has defined token bus standards as broadband networks, as opposed to Ethernet's baseband transmission technique. The topology of the network can include groups of workstations connected by long trunk cables.

These workstations branch from hubs in a star configuration, so the network has both a bus and star topology. Token bus topology is well suited to groups of users that are separated by some distance. IEEE 802.4 token bus networks are constructed with 75-ohm coaxial cable using a bus topology. The broadband characteristics of the 802.4 standard support transmission over several different channels simultaneously.

The token and frames of data are passed from one station to another following the numeric sequence of the station addresses. Thus, the token follows a logical ring rather than a physical ring. The last station in numeric order passes the token back to the first station. The token does not follow the physical ordering of workstation attachment to the cable. Station 1 might be at one end of the cable and station 2 might be at the other, with station 3 in the middle.

While token bus is used in some manufacturing environments, Ethernet and token ring standards have become more prominent in the office environment.

IEEE 802.5 Token Ring: Token ring is the IEEE 802.5 standard for a token-passing ring network with a star-configured physical topology. Internally, signals travel around the network from one station to the next in a ring. Physically, each station connects to a central hub called a MAU (multistation access unit). The MAU contains a "collapsed ring," but the physical configuration is a star topology. When a station is attached, the ring is extended out to the station and then back to the MAU.

If a station goes offline, the ring is reestablished with a bypass at the station connector. Token ring was popular for an extended period in the late 1980s and 1990s, especially in IBM legacy system environments. IBM developed the technology and provided extensive support for connections to SNA systems. More recently, Ethernet, Fast Ethernet, and Gigabit Ethernet technologies have pushed token ring and other LAN technologies to the sidelines.

Historically, **10Base5** cabling, popularly called **thick Ethernet,** came first. It resembles a yellow garden hose, with markings every 2.5 meters to show where the taps go. (The 802.3

standard does not actually *require* the cable to be yellow, but it does *suggest* it.) Connections to it are generally made using **vampire taps**, in which a pin is *very* carefully forced halfway into the coaxial cable's core. The notation 10Base5 means that it operates at 10 Mbps, uses baseband signaling, and can support segments of up to 500 meters. The first number is the speed in Mbps. Then comes the word "Base" (or sometimes "BASE") to indicate baseband transmission. There used to be a broadband variant, 10Broad36, but it never caught on in the marketplace and has since vanished. Finally, if the medium is coax, its length is given rounded to units of 100 m after "Base."

Historically, the second cable type was **10Base2**, or **thin Ethernet,** which, in contrast to the garden-hose-like thick Ethernet, bends easily. Connections to it are made using industry-standard BNC connectors to form T junctions, rather than using vampire taps. BNC connectors are easier to use and more reliable. Thin Ethernet is much cheaper and easier to install, but it can run for only 185 meters per segment, each of which can handle only 30 machines.

Detecting cable breaks, excessive length, bad taps, or loose connectors can be a major problem with both media. For this reason, techniques have been developed to track them down. Basically, a pulse of known shape is injected into the cable. If the pulse hits an obstacle or the end of the cable, an echo will be generated and sent back. By carefully timing the interval between sending the pulse and receiving the echo, it is possible to localize the origin of the echo. This technique is called **time domain reflectometry**.

The problems associated with finding cable breaks drove systems toward a different kind of wiring pattern, in which all stations have a cable running to a central **hub** in which they are all connected electrically (as if they were soldered together). Usually, these wires are telephone company twisted pairs, since most office buildings are already wired this way, and normally plenty of spare pairs are available. This scheme is called **10Base-T**. Hubs do not buffer incoming traffic. We will discuss an improved version of this idea (switches), which do buffer incoming traffic later.

For 10Base5, a **transceiver** is clamped securely around the cable so that its tap makes contact with the inner core. The transceiver contains the electronics that handle carrier detection and collision detection. When a collision is detected, the transceiver also puts a special invalid signal on the cable to ensure that all other transceivers also realize that a collision has occurred.

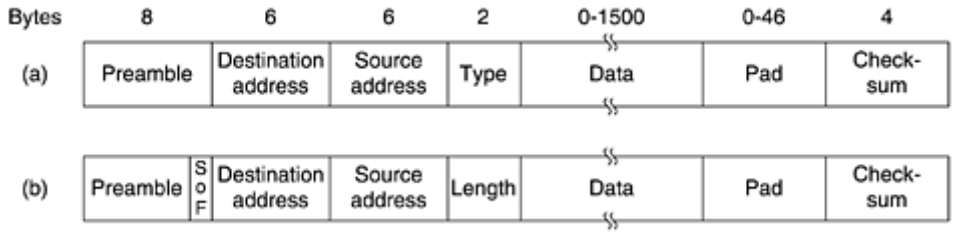


Figure 3.7. Frame formats. (a) DIX Ethernet. (b) IEEE 802.3

Types:

- Fast Ethernet

- Gigabit Ethernet
- Ten- Gigabit Ethernet

3.4. Wireless LANs

- IEEE 802.11, the Working Group Setting the Standards for Wireless LANs.
- WiFi Alliance
- IEEE 802.11x and 802.11aa IEEE standards for authentication
- Wi-Fi Planet News and hype about IEEE 802.11 wireless LANs
- IEEE 802.11 Wikipedia article
- ZigBee versus other wireless networking standards A comparison with Bluetooth, 802.11, etc.

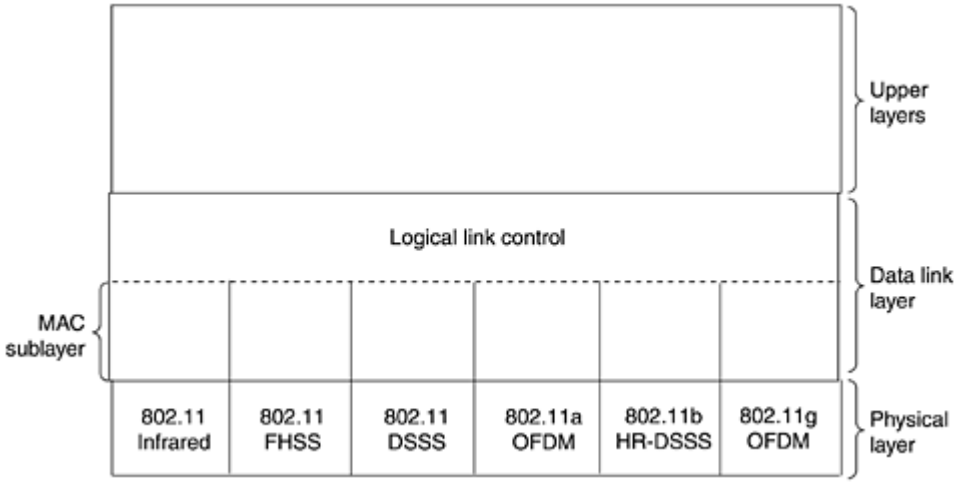


Figure 3.8. The 802.11 Protocol Stack

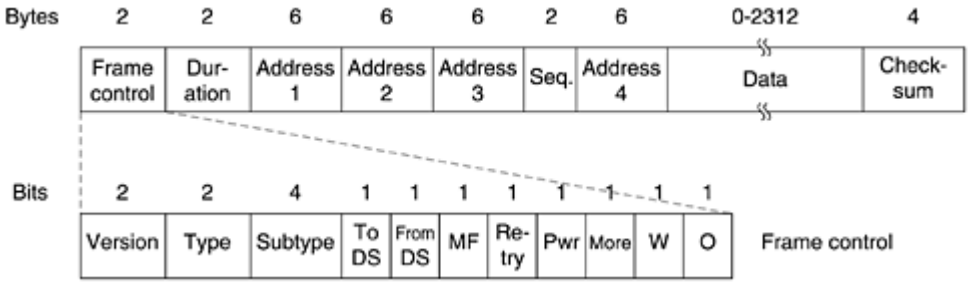


Figure 3.9. Frame Format

The protocol starts when *A* decides it wants to send data to *B*. It begins by sending an RTS frame to *B* to request permission to send it a frame. When *B* receives this request, it may decide to grant permission, in which case it sends a CTS frame back. Upon receipt of the CTS, *A* now sends its frame and starts an ACK timer. Upon correct receipt of the data frame, *B* responds with an ACK frame, terminating the exchange. If *A*'s ACK timer expires before the ACK gets back to it, the whole protocol is run again.

Now let us consider this exchange from the viewpoints of *C* and *D*. *C* is within range of *A*, so it may receive the RTS frame. If it does, it realizes that someone is going to send data soon, so for the good of all it desists from transmitting anything until the exchange is completed. From the information provided in the RTS request, it can estimate how long the sequence will take, including the final ACK, so it asserts a kind of virtual channel busy for itself, indicated by **NAV** (**Network Allocation Vector**)

3.5. BROADBAND WIRELESS

- IEEE 802.16 (Wireless MAN) Fixed broadband (MMDS and LMDS)
- Wireless broadband Wikipedia article
- IEEE 802.16 Wikipedia article
- WiMAX Wikipedia article
- WiMAX Forum Industry news, press releases, white papers, etc.

Running fiber, coax, or even category 5 twisted pair to millions of homes and businesses is prohibitively expensive.

The answer is broadband wireless. Erecting a big antenna on a hill just outside of town and installing antennas directed at it on customers' roofs is much easier and cheaper than digging trenches and stringing cables. Thus, competing telecommunication companies have a great interest in providing a multimegabit wireless communication service for voice, Internet, movies on demand, etc.

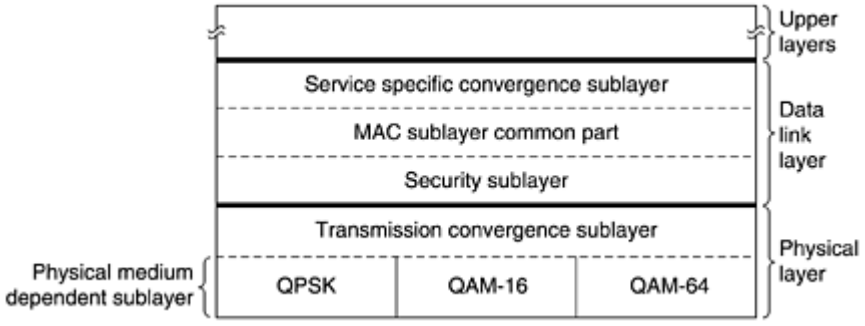


Figure. 3.10. Protocol stack

Many people in the industry realized that having a broadband wireless standard was the key element missing, so IEEE was asked to form a committee composed of people from key companies and academia to draw up the standard. The next number available in the 802 numbering space was **802.16**, so the standard got this number. Work was started in July 1999, and the final standard was approved in April 2002. Officially the standard is called "Air Interface for Fixed Broadband Wireless Access Systems." However, some people prefer to call it a **wireless MAN (Metropolitan Area Network)** or a **wireless local loop**. We regard all these terms as interchangeable.

Like some of the other 802 standards, 802.16 was heavily influenced by the OSI model, including the (sub) layers, terminology, service primitives, and more. Unfortunately, also like OSI, it is fairly complicated. In the following sections we will give a brief description of some of the highlights of 802.16, but this treatment is far from complete and leaves out many details. For additional information about broadband wireless in general, see (Bolcskei et al., 2001; and Webb, 2001). For information about 802.16 in particular, see (Eklund et al., 2002).

3.6. BLUETOOTH

- Bluetooth is to allow very different (portable and fixed) devices located in each other's proximity to exchange information:
  - Let very different portable devices (PDA, cellular phone, notebook) set up connections
  - Replace many of the existing cables (headset, keyboard, mouse, printer) Provide better wireless connection (handsfree solutions)
  - Provide wireless access to Internet entry points Relatively high bandwidth: 1 Mbit/second
- Also referred to as IEEE 802.15.1
- It's named after a Viking king who unified Denmark and Norway (940-981) Paolo Costa 04 - MAC Sublayer Bluetooth 53

Bluetooth Architecture

Piconet: Group of devices with one master and multiple slaves. There can as much as 7 active slaves, but a total of 255 parked ones (i.e., in a power-saving state).

Scatternet: An interconnected collection of piconets A piconet is a centralized TDM system with the master determining which device gets to communicate the connection procedure for a non-existent piconet is initiated by any of the devices, which then becomes the master The master-

slave design facilitates the implementation of Bluetooth chips for under 5$ Paolo Costa 04 - MAC Sublayer Bluetooth

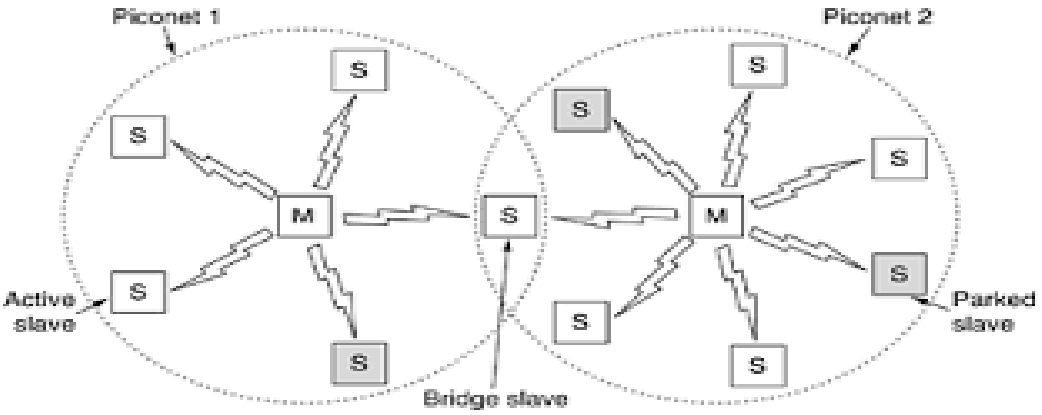


Figure 3.11.Two piconets can be connected to form a scatternet

Bluetooth Protocol Stack (1/2)

Radio: it uses frequency hopping (2.4 GHz band):

- Take data signal and modulate it with a carrier signal that changes frequency in hops.
- Good to minimize interference from other devices (microwave ovens!) hops for Bluetooth: fixed at 2402 + k MHz, k = 0, 1. . . 78.
- Modulation is frequency shift keying with 1 bit / Hertz ⇒ 1Mbps data rate but much of this is consumed as overhead
- Baseband: Core of the data link layer.
  - Determines timing, framing, packets, and flow control.
  - Provides synchronous and asynchronous data communication.
  - Error correction can be used to provide higher reliability

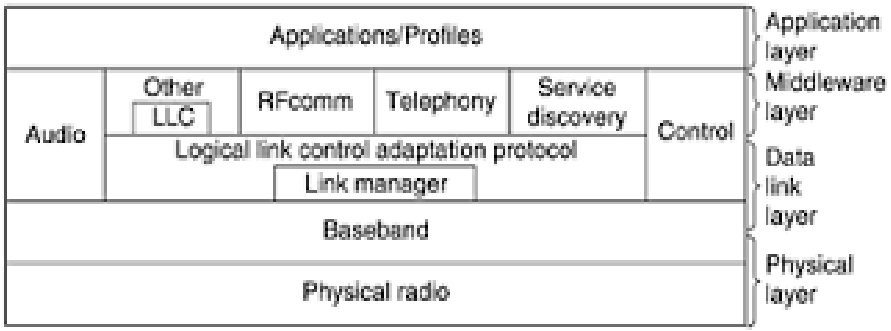


Figure 3.12. 802.15 version of the Bluetooth protocol architecture

Bluetooth Protocol Stack (2/2)

Link manager: Manages connections, power management

Logical link control: Multiplexing of higher-level protocols, segmentation and reassembly of large packets, device discovery

Audio: Handles streaming for voice-related applications

RFCOMM: Emulate serial cable based on GSM protocol

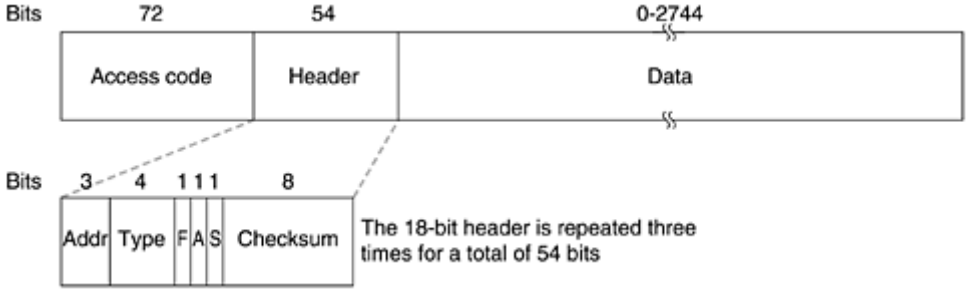


Figure 3.13. Typical Bluetooth data frame

3.7. Data Link Layer Switching

- LAN Switching LAN switching tutorial from Cisco
- Multiprotocol Label Switching (MPLS) Wikipedia article
- Virtual LANs
  - IEEE 802.1Q Home Page Many 802.1Q links
  - Virtual LAN s Links to articles about VLANs
  - VLAN Tutorial everything about VLANs, from Computer-Network.net
  - VLAN Basics Tutorial A brief tutorial on VLANs.
  - Virtual LAN Wikipedia Article
  - Multiprotocol Label Switching article at Cisco
- Many organizations have multiple LANs and wish to connect them. LANs can be connected by devices called **bridges**, which operate in the data link layer. Bridges examine the data layer link addresses to do routing. Since they are not supposed to examine the payload field of the frames they route, they can transport IPv4 (used in the Internet now), IPv6 (will be used in the Internet in the future), AppleTalk, ATM, OSI, or any other kinds of packets. In contrast, *routers* examine the addresses in packets and route based on them. Although this seems like a clear division between bridges and routers, some modern developments, such as the advent of switched Ethernet, have muddied the waters, as we will see later. In the following sections we will look at bridges and switches, especially for connecting different 802 LANs. For a comprehensive treatment of bridges, switches, and related topics, see (Perlman, 2000).
- Before getting into the technology of bridges, it is worthwhile taking a look at some common situations in which bridges are used. We will mention six reasons why a single organization may end up with multiple LANs.

- First, many university and corporate departments have their own LANs, primarily to connect their own personal computers, workstations, and servers. Since the goals of the various departments differ, different departments choose different LANs, without regard to what other departments are doing. Sooner or later, there is a need for interaction, so bridges are needed. In this example, multiple LANs came into existence due to the autonomy of their owners.
- Second, the organization may be geographically spread over several buildings separated by considerable distances. It may be cheaper to have separate LANs in each building and connect them with bridges and laser links than to run a single cable over the entire site.
- Third, it may be necessary to split what is logically a single LAN into separate LANs to accommodate the load. At many universities, for example, thousands of workstations are available for student and faculty computing. Files are normally kept on file server machines and are downloaded to users' machines upon request. The enormous scale of this system precludes putting all the workstations on a single LAN—the total bandwidth needed is far too high. Instead, multiple LANs connected by bridges are used, as shown in Fig 3.14. Each LAN contains a cluster of workstations with its own file server so that most traffic is restricted to a single LAN and does not add load to the backbone.

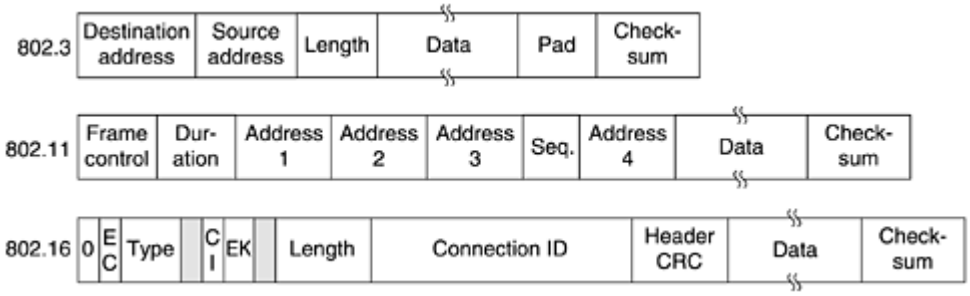


Figure. 3.14. The IEEE 802 frame formats

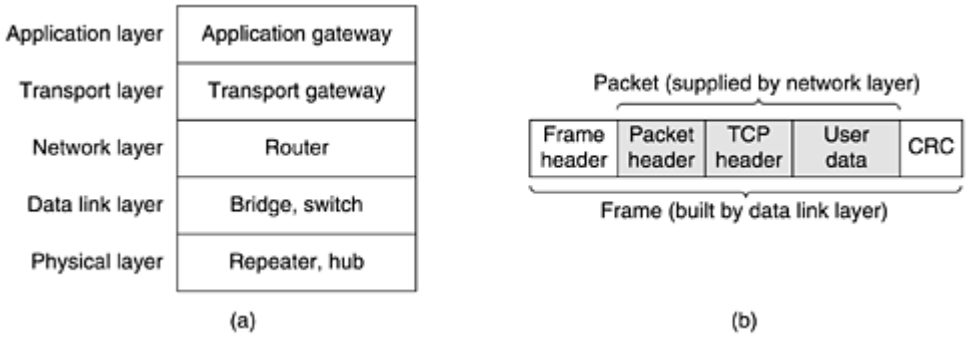


Figure. 3.15. (a) Which device is in which layer. (b) Frames, packets, and headers.

Now let us look at the switching devices and see how they relate to the packets and frames. At the bottom, in the physical layer, we find the repeaters. These are analog devices that are connected to two cable segments. A signal appearing on one of them is amplified and put out on the other. Repeaters do not understand frames, packets, or headers. They understand volts. Classic Ethernet, for example, was designed to allow four repeaters, in order to extend the maximum cable length from 500 meters to 2500 meters.

Next we come to the hubs. A hub has a number of input lines that it joins electrically. Frames arriving on any of the lines are sent out on all the others. If two frames arrive at the same time, they will collide, just as on a coaxial cable. In other words, the entire hub forms a single collision domain. All the lines coming into a hub must operate at the same speed. Hubs differ from repeaters in that they do not (usually) amplify the incoming signals and are designed to hold multiple line cards each with multiple inputs, but the differences are slight. Like repeaters, hubs do not examine the 802 addresses or use them in any way.

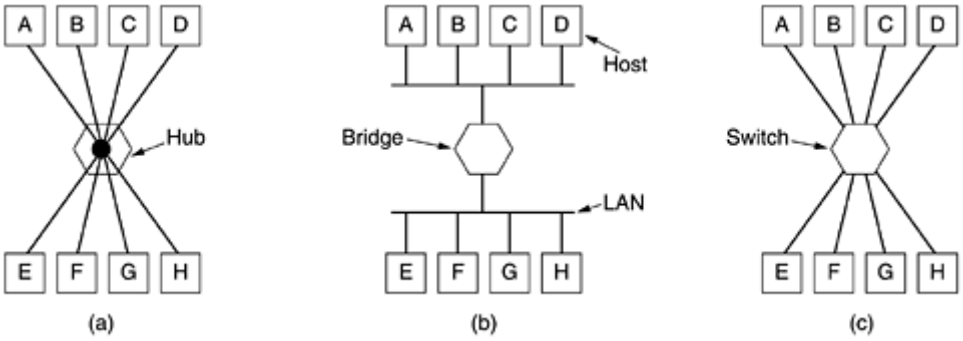


Figure 3.16. (a) A hub. (b) A bridge. (c) A switch

3.8 Network Layer

3.8.1 Network Layer Design Issues

3.8.1.1 Store-and-Forward Packet Switching

The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval.

Host H1 is directly connected to one of the carrier's routers, A, by a leased line. In contrast, H2 is on a LAN with a router, F, owned and operated by the customer. This router also has a leased line to the carrier's equipment.

· We have shown F as being outside the oval because it does not belong to the carrier, but in

terms of construction, software, and protocols, it is probably no different from the carrier's routers.

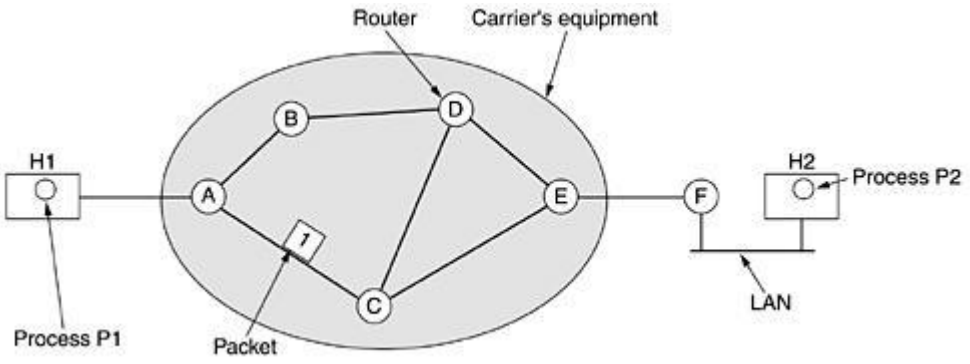


Figure 3.17 The environment of the network layer protocols.

This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier. The packet is stored there until it has fully arrived so the checksum can be verified.

·Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.

3.8.1.2. Services provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is what kind of services the network layer provides to the transport layer.

The network layer services have been designed with the following goals in mind.

1. The services should be independent of the router technology.

2. The transport layer should be shielded from the number, type, and topology of the routers present.

3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer. This freedom often degenerates into a raging battle between two warring factions.

The other camp argues that the subnet should provide a reliable, connection-oriented service. They claim that 100 years of successful experience with the worldwide telephone system is an

excellent guide. In this view, quality of service is the dominant factor, and without connections in the subnet, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video.

These two camps are best exemplified by the Internet and ATM. The Internet offers connectionless network-layer service; ATM networks offer connection-oriented network-layer service. However, it is interesting to note that as quality-of-service guarantees are becoming more and more important, the Internet is evolving.

3.8.1.3Implementation of Connectionless Service

Two different organizations are possible, depending on the type of service offered. If connectionless service is offered, packets are injected into the subnet individually and routed independently of each other. No advance setup is needed.

In this context, the packets are frequently called datagrams (in analogy with telegrams) and the subnet is called a datagram subnet. If connection-oriented service is used, a path from the source router to the destination router must be established before any data packets can be sent.

This connection is called a VC (virtual circuit), in analogy with the physical circuits set up by the telephone system, and the subnet is called a virtual-circuit subnet. In this section we will examine datagram subnets; in the next one we will examine virtual-circuit subnets.

Let us now see how a datagram subnet works. Suppose that the process P1 in Fig. 3-18 has a long message for P2. It hands the message to the transport layer with instructions to deliver it to process P2 on host H2.

The transport layer code runs on H1, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer, probably just another procedure within the operating system.

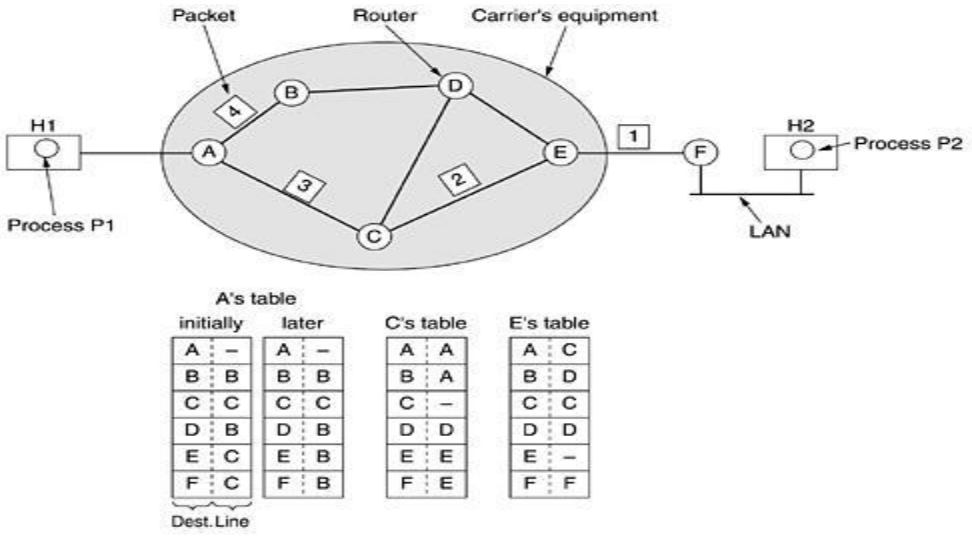


Figure 3.18 Routing within a datagram subnet.

Let us assume that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4 and sends each of them in turn to router A using some point-to-point protocol, for example, PPP.

At this point the carrier takes over. Every router has an internal table telling it where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination.

Only directly-connected lines can be used. A has only two outgoing lines—to B and C—so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router. A's initial routing table is shown in the figure under the label ''initially.''

However, something different happened to packet 4. When it got to A it was sent to router B, even though it is also destined for F. For some reason, A decided to send packet 4 via a different route than that of the first three.

Perhaps it learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label ''later.'' The algorithm that manages the tables and makes the routing decisions is called the routing algorithm.

3.8.1.4. Implementation of Connection-Oriented Service

For connection-oriented service, we need a virtual-circuit subnet. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in Fig.

Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works.

When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to. As an example, consider the situation of Fig. 3-3. Here, host H1 has established connection 1 with host H2.

It is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.

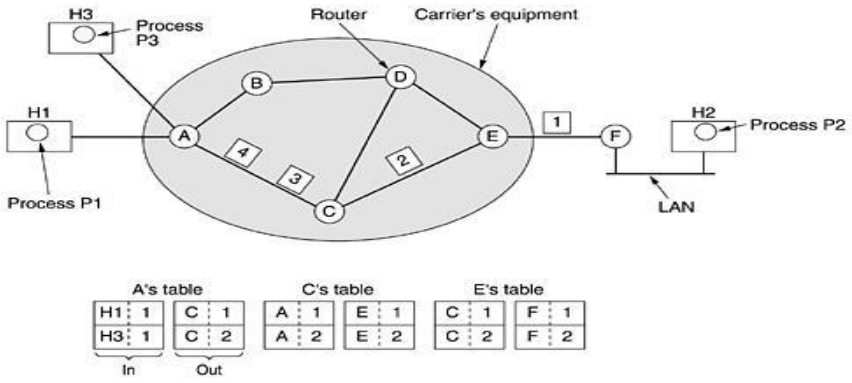


Figure 3-19. Routing within a virtual-circuit subnet.

Now let us consider what happens if H3 also wants to establish a connection to H2. It chooses connection identifier 1 and tells the subnet to establish the virtual circuit. This leads to the second row in the tables.

Note that we have a conflict here because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this. For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection.

Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets. In some contexts, this is called label switching.

3.8.1.5. Comparison of Virtual-Circuit and Datagram Subnets

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize the arguments both ways. The major issues are listed in Fig. 3-4, although purists could probably find a counterexample for everything in the figure.

Table 3.1

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Inside the subnet, several trade-offs exist between virtual circuits and datagrams. One trade-off is between router memory space and bandwidth. Virtual circuits allow packets to contain circuit numbers instead of full destination addresses. If the packets tend to be fairly short, a full destination address in every packet may represent a significant amount of overhead and hence, wasted bandwidth. The price paid for using virtual circuits internally is the table space within the routers.

Depending upon the relative cost of communication circuits versus router memory, one or the other may be cheaper. Another trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources.

However, figuring out what to do with a data packet in a virtual-circuit subnet is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram subnet, a more complicated lookup procedure is required to locate the entry for the destination.

For transaction processing systems (e.g., stores calling up to verify credit card purchases), the overhead required to set up and clear a virtual circuit may easily dwarf the use of the circuit. If the majority of the traffic is expected to be of this kind, the use of virtual circuits inside the subnet makes little sense.

On the other hand, permanent virtual circuits, which are set up manually and last for months or years, may be useful here. Virtual circuits also have a vulnerability problem. If a router crashes and loses its memory, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted.

In contrast, if a datagram router goes down, only those users whose packets were queued in the router at the time will suffer, and maybe not even all those, depending upon whether they have already been acknowledged.

The loss of a communication line is fatal to virtual circuits using it but can be easily compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the subnet, since routes can be changed partway through a long sequence of packet transmissions.

3.9 What is Network Layer?

The network layer is concerned with getting packets from the source all the way to the destination. The packets may require to make many hops at the intermediate routers while reaching the destination. This is the lowest layer that deals with end to end transmission. In order to achieve its goals, the network layer must know about the topology of the communication network. It must also take care to choose routes to avoid overloading of some of the communication lines while leaving others idle. The network layer-transport layer interface frequently is the interface between the carrier and the customer, that is the boundary of the subnet. The functions of this layer include :

1. Routing - The process of transferring packets received from the Data Link Layer of the source network to the Data Link Layer of the correct destination network is called routing. Involves decision making at each intermediate node on where to send the

packet next so that it eventually reaches its destination. The node which makes this choice is called a router. For routing we require some mode of addressing which is recognized by the Network Layer. This addressing is different from the MAC layer addressing.

2. Inter-networking - The network layer is the same across all physical networks (such as Token-Ring and Ethernet). Thus, if two physically different networks have to communicate, the packets that arrive at the Data Link Layer of the node which connects these two physically different networks, would be stripped of their headers and passed to the Network Layer. The network layer would then pass this data to the Data Link Layer of the other physical network..
3. Congestion Control - If the incoming rate of the packets arriving at any router is more than the outgoing rate, then congestion is said to occur. Congestion may be caused by many factors. If suddenly, packets begin arriving on many input lines and all need the same output line, then a queue will build up. If there is insufficient memory to hold all of them, packets will be lost. But even if routers have an infinite amount of memory, congestion gets worse, because by the time packets reach to the front of the queue, they have already timed out (repeatedly), and duplicates have been sent. All these packets are dutifully forwarded to the next router, increasing the load all the way to the destination. Another reason for congestion are slow processors. If the router's CPUs are slow at performing the bookkeeping tasks required of them, queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion.

The main functions performed by the network layer are as follows:

- Routing
- Congestion Control
- Internetwokring

3.10 Routing

Routing is the process of forwarding of a packet in a network so that it reaches its intended destination. The main goals of routing are:

1. **Correctness:** The routing should be done properly and correctly so that the packets may reach their proper destination.
2. **Simplicity:** The routing should be done in a simple manner so that the overhead is as low as possible. With increasing complexity of the routing algorithms the overhead also increases.
3. **Robustness:** Once a major network becomes operative, it may be expected to run continuously for years without any failures. The algorithms designed for routing should be robust enough to handle hardware and software failures and should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network rebooted every time some router goes down.
4. **Stability:** The routing algorithms should be stable under all possible circumstances.
5. **Fairness:** Every node connected to the network should get a fair chance of transmitting their packets. This is generally done on a first come first serve basis.
6. **Optimality:** The routing algorithms should be optimal in terms of throughput and minimizing mean packet delays. Here there is a trade-off and one has to choose depending on his suitability.

3.10.1Classification of Routing Algorithms

The routing algorithms may be classified as follows:

1. **Adaptive Routing Algorithm:** These algorithms change their routing decisions to reflect changes in the topology and in traffic as well. These get their routing information from adjacent routers or from all routers. The optimization parameters are the distance, number of hops and estimated transit time. This can be further classified as follows:
    1. **Centralized:** In this type some central node in the network gets entire information about the network topology, about the traffic and about other nodes. This then transmits this information to the respective routers. The advantage of this is that only one node is required to keep the information. The disadvantage is that if the central node goes down the entire network is down, i.e. single point of failure.
    2. **Isolated:** In this method the node decides the routing without seeking information from other nodes. The sending node does not know about the status of a particular link. The disadvantage is that the packet may be send through a congested route resulting in a delay. Some examples of this type of algorithm for routing are:
        - **Hot Potato:** When a packet comes to a node, it tries to get rid of it as fast as it can, by putting it on the shortest output queue without regard to where that link leads. A variation of this algorithm is to combine static routing with the hot potato algorithm. When a packet arrives, the routing algorithm takes into account both the static weights of the links and the queue lengths.
        - **Backward Learning:** In this method the routing tables at each node gets modified by information from the incoming packets. One way to implement backward learning is to include the identity of the source node in each packet, together with a hop counter that is incremented on each hop. When a node receives a packet in a particular line, it notes down the number of hops it has taken to reach it from the source node. If the previous value of hop count stored in the node is better than the current one then nothing is done but if the current value is better then the value is updated for future use. The problem with this is that when the best route goes down then it cannot recall the second best route to a particular node. Hence all the nodes have to forget the stored informations periodically and start all over again.
    3. **Distributed:** In this the node receives information from its neighbouring nodes and then takes the decision about which way to send the packet. The disadvantage is that if in between the the interval it receives information and sends the paket something changes then the packet may be delayed.
2. **Non-Adaptive Routing Algorithm:** These algorithms do not base their routing decisions on measurements and estimates of the current traffic and topology. Instead the route to be taken in going from one node to the other is computed in advance, off-line, and downloaded to the routers when the network is booted. This is also known as static routing. This can be further classified as:
    1. **Flooding:** Flooding adapts the technique in which every incoming packet is sent on every outgoing line except the one on which it arrived. One problem with this method is that packets may go in a loop. As a result of this a node may receive several copies of a particular packet which is undesirable. Some techniques adapted to overcome these problems are as follows:

- **Sequence Numbers:** Every packet is given a sequence number. When a node receives the packet it sees its source address and sequence number. If the node finds that it has sent the same packet earlier then it will not transmit the packet and will just discard it.
- **Hop Count:** Every packet has a hop count associated with it. This is decremented(or incremented) by one by each node which sees it. When the hop count becomes zero(or a maximum possible value) the packet is dropped.
- **Spanning Tree:** The packet is sent only on those links that lead to the destination by constructing a spanning tree routed at the source. This avoids loops in transmission but is possible only when all the intermediate nodes have knowledge of the network topology.

Flooding is not practical for general kinds of applications. But in cases where high degree of robustness is desired such as in military applications, flooding is of great help.

2. **Random Walk:** In this method a packet is sent by the node to one of its neighbours randomly. This algorithm is highly robust. When the network is highly interconnected, this algorithm has the property of making excellent use of alternative routes. It is usually implemented by sending the packet onto the least queued link.

Delta Routing

Delta routing is a hybrid of the centralized and isolated routing algorithms. Here each node computes the cost of each line (i.e some functions of the delay, queue length, utilization, bandwidth etc) and periodically sends a packet to the central node giving it these values which then computes the **k** best paths from node **i** to node **j**. Let **Cij1** be the cost of the best **i-j** path, **Cij2** the cost of the next best path and so on.If **Cijn - Cij1 < delta**, (**Cijn** - cost of **n'th** best**i-j** path, **delta** is some constant) then path **n** is regarded equivalent to the best **i-j** path since their cost differ by so little. When **delta -> 0** this algorithm becomes centralized routing and when **delta -> infinity** all the paths become equivalent.

Multipath Routing

In the above algorithms it has been assumed that there is a single best path between any pair of nodes and that all traffic between them should use it. In many networks however there are several paths between pairs of nodes that are almost equally good. Sometimes in order to improve the performance multiple paths between single pair of nodes are used. This technique is called multipath routing or bifurcated routing. In this each node maintains a table with one row for each possible destination node. A row gives the best, second best, third best, etc outgoing line for that destination, together with a relative weight. Before forwarding a packet, the node generates a random number and then chooses among the alternatives, using the weights as probabilities. The tables are worked out manually and loaded into the nodes before the network is brought up and not changed thereafter.

Hierarchical Routing

In this method of routing the nodes are divided into regions based on hierarchy. A particular node can communicate with nodes at the same hierarchial level or the nodes at a lower level and directly under it. Here, the path from any source to a destination is fixed and is exactly one if the heirarchy is a tree.

Non-Hierarchical Routing

In this type of routing, interconnected networks are viewed as a single network, where bridges, routers and gateways are just additional nodes.

- Every node keeps information about every other node in the network
- In case of adaptive routing, the routing calculations are done and updated for all the nodes.

The above two are also the disadvantages of non-hierarchical routing, since the table sizes and the routing calculations become too large as the networks get bigger. So this type of routing is feasible only for small networks.

Hierarchical Routing

This is essentially a 'Divide and Conquer' strategy. The network is divided into different regions and a router for a particular region knows only about its own domain and other routers. Thus, the network is viewed at two levels:

1. The Sub-network level, where each node in a region has information about its peers in the same region and about the region's interface with other regions. Different regions may have different 'local' routing algorithms. Each local algorithm handles the traffic between nodes of the same region and also directs the outgoing packets to the appropriate interface.
2. The Network Level, where each region is considered as a single node connected to its interface nodes. The routing algorithms at this level handle the routing of packets between two interface nodes, and is isolated from intra-regional transfer.

Networks can be organized in hierarchies of many levels; e.g. local networks of a city at one level, the cities of a country at a level above it, and finally the network of all nations.

In Hierarchical routing, the interfaces need to store information about:

- All nodes in its region which are at one level below it.
- Its peer interfaces.
- At least one interface at a level above it, for outgoing packages.

Advantages of Hierarchical Routing :

- Smaller sizes of routing tables.
- Substantially lesser calculations and updates of routing tables.

Disadvantage :

- Once the hierarchy is imposed on the network, it is followed and possibility of direct paths is ignored. This may lead to sub optimal routing.

Source Routing

Source routing is similar in concept to virtual circuit routing. It is implemented as under:

- Initially, a path between nodes wishing to communicate is found out, either by flooding or by any other suitable method.
- This route is then specified in the header of each packet routed between these two nodes. A route may also be specified partially, or in terms of some intermediate hops.

Advantages:

- Bridges do not need to lookup their routing tables since the path is already specified in the packet itself.
- The throughput of the bridges is higher, and this may lead to better utilization of bandwidth, once a route is established.

Disadvantages:

- Establishing the route at first needs an expensive search method like flooding.
- To cope up with dynamic relocation of nodes in a network, frequent updates of tables are required, else all packets would be sent in wrong direction. This too is expensive.

Policy Based Routing

In this type of routing, certain restrictions are put on the type of packets accepted and sent. e.g.. The IIT- K router may decide to handle traffic pertaining to its departments only, and reject packets from other routes. This kind of routing is used for links with very low capacity or for security purposes.

Shortest Path Routing

Here, the central question dealt with is 'How to determine the optimal path for routing ?' Various algorithms are used to determine the optimal routes with respect to some predetermined criteria. A network is represented as a graph, with its terminals as nodes and the links as edges. A 'length' is associated with each edge, which represents the cost of using the link for transmission. Lower the cost, more suitable is the link. The cost is determined depending upon the criteria to be optimized. Some of the important ways of determining the cost are:

- **Minimum number of hops:** If each link is given a unit cost, the shortest path is the one with minimum number of hops. Such a route is easily obtained by a breadth first search method. This is easy to implement but ignores load, link capacity etc.
- **Transmission and Propagation Delays:** If the cost is fixed as a function of transmission and propagation delays, it will reflect the link capacities and the geographical distances. However these costs are essentially static and do not consider the varying load conditions.

- **Queuing Delays:** If the cost of a link is determined through its queuing delays, it takes care of the varying load conditions, but not of the propagation delays.

Ideally, the cost parameter should consider all the above mentioned factors, and it should be updated periodically to reflect the changes in the loading conditions. However, if the routes are changed according to the load, the load changes again. This feedback effect between routing and load can lead to undesirable oscillations and sudden swings.

Routing Algorithms

As mentioned above, the shortest paths are calculated using suitable algorithms on the graph representations of the networks. Let the network be represented by graph G (V, E) and let the number of nodes be 'N'. For all the algorithms discussed below, the costs associated with the links are assumed to be positive. A node has zero cost w.r.t itself. Further, all the links are assumed to be symmetric, i.e. if $d_{i,j}$ = cost of link from node i to node j, then $d_{i,j} = d_{j,i}$. The graph is assumed to be complete. If there exists no edge between two nodes, then a link of infinite cost is assumed. The algorithms given below find costs of the paths from all nodes to a particular node; the problem is equivalent to finding the cost of paths from a source to all destinations.

3.11 Bellman-Ford Algorithm

This algorithm iterates on the number of edges in a path to obtain the shortest path. Since the number of hops possible is limited (cycles are implicitly not allowed), the algorithm terminates giving the shortest path.

Notation:

$d_{i,j}$ = Length of path between nodes i and j, indicating the cost of the link.
h = Number of hops.
D[i,h] = Shortest path length from node i to node 1, with upto 'h' hops.
D[1,h] = 0 for all h .

Algorithm :

Initial condition : D[i, 0] = infinity, for all i (i != 1)

Iteration : $D[i, h+1] = \min\{ d_{i,j} + D[j,h] \}$ over all values of j .

Termination : The algorithm terminates when

D[i, h] = D [i, h+1] for all i .

Principle:

For zero hops, the minimum length path has length of infinity, for every node. For one hop the shortest-path length associated with a node is equal to the length of the edge between that node and node 1. Hereafter, we increment the number of hops allowed, (from h to h+1) and find out whether a shorter path exists through each of the other nodes. If it exists, say through node 'j', then its length must be the sum of the lengths between these two nodes (i.e. $d_{i,j}$) and the shortest path between j and 1 obtainable in upto h paths. If such a path doesn't exist, then

the path length remains the same. The algorithm is guaranteed to terminate, since there are utmost N nodes, and so N-1 paths. It has time complexity of O (N^3) .

3.12 Dijkstra's Algorithm

Notation:
D_i = Length of shortest path from node 'i' to node 1.
$d_{i,j}$ = Length of path between nodes i and j .

Algorithm
Each node j is labeled with Dj, which is an estimate of cost of path from node j to node 1. Initially, let the estimates be infinity, indicating that nothing is known about the paths. We now iterate on the length of paths, each time revising our estimate to lower values, as we obtain them. Actually, we divide the nodes into two groups ; the first one, called set P contains the nodes whose shortest distances have been found, and the other Q containing all the remaining nodes. Initially P contains only the node 1. At each step, we select the node that has minimum cost path to node 1. This node is transferred to set P. At the first step, this corresponds to shifting the node closest to 1 in P. Its minimum cost to node 1 is now known. At the next step, select the next closest node from set Q and update the labels corresponding to each node using :

$$D_j = \min [D_j , D_i + d_{j,i}] \quad (3.4)$$

Finally, after N-1 iterations, the shortest paths for all nodes are known, and the algorithm terminates.

Principle
Let the closest node to 1 at some step be i. Then i is shifted to P. Now, for each node j , the closest path to 1 either passes through i or it doesn't. In the first case Dj remains the same. In the second case, the revised estimate of D_j is the sum $D_i + d_{i,j}$. So we take the minimum of these two cases and update D_j accordingly. As each of the nodes get transferred to set P, the estimates get closer to the lowest possible value. When a node is transferred, its shortest path length is known. So finally all the nodes are in P and the D_j 's represent the minimum costs. The algorithm is guaranteed to terminate in N-1 iterations and its complexity is O(N^2).

3.13 The Floyd Warshall Algorithm

This algorithm iterates on the set of nodes that can be used as intermediate nodes on paths. This set grows from a single node (say node 1) at start to finally all the nodes of the graph. At each iteration, we find the shortest path using given set of nodes as intermediate nodes, so that finally all the shortest paths are obtained.

Notation
$D_{i,j}$ [n] = Length of shortest path between the nodes i and j using only the nodes 1,2,....n as intermediate nodes.

Initial Condition
Di,j[0] = $d_{i,j}$ for all nodes i,j .

Algorithm

Initially, n = 0. At each iteration, add next node to n. i.e. For n = 1,2, ...N-1 ,

$$Di,j[n + 1] = \min \{ D_{i,j}[n] , D_{i,n+1}[n] + D_{n+1,j}[n] \} \quad (3.5)$$

Principle

Suppose the shortest path between i and j using nodes 1,2,...n is known. Now, if node n+1 is allowed to be an intermediate node, then the shortest path under new conditions either passes through node n+1 or it doesn't. If it does not pass through the node n+1, then $D_{i,j}[n+1]$ is same as $D_{i,j}[n]$. Else, we find the cost of the new route, which is obtained from the sum, $D_{i,n+1}[n] + D_{n+1,j}[n]$. So we take the minimum of these two cases at each step. After adding all the nodes to the set of intermediate nodes, we obtain the shortest paths between all pairs of nodes together. The complexity of Floyd-Warshall algorithm is $O (N^3)$.

It is observed that all the three algorithms mentioned above give comparable performance, depending upon the exact topology of the network.

3.14 Address Resolution Protocol

If a machine talks to another machine in the same network, it requires its physical or MAC address. But ,since the application has given the destination's IP address it requires some mechanism to bind the IP address with its MAC address.This is done through Address Resolution protocol (ARP).IP address of the destination node is broadcast and the destination node informs the source of its MAC address.

1. Assume broadcast nature of LAN
2. Broadcast IP address of the destination
3. Destination replies it with its MAC address.
4. Source maintains a cache of IP and MAC address bindings

But this means that every time machine A wants to send packets to machine B, A has to send an ARP packet to resolve the MAC address of B and hence this will increase the traffic load too much, so to reduce the communication cost computers that use ARP maintains a cache of recently acquired IP_to_MAC address bindings, i.e. they dont have to use ARP repeatedly. ARP Refinements Several refinements of ARP are possible: When machine A wants to send packets to macine B, it is possible that machine B is going to send packets to machine A in the near future.So to avoid ARP for machine B, A should put its IP_to_MAC address binding in the special packet while requesting for the MAC address of B. Since A broadcasts its initial request for the MAC address of B, every machine on the network should extract and store in its cache the IP_to_MAC address binding of A When a new machine appears on the network (e.g. when an operating system reboots) it can broadcast its IP_to_MAC address binding so that all other machines can store it in their caches. This will eliminate a lot of ARP packets by all other machines, when they want to communicate with this new machine.

Example displaying the use of Address Resolution Protocol:

Consider a scenario where a computer tries to contact some remote machine using ping program, assuming that there has been no exchange of IP datagrams previously between the two machines and therefore arp packet must be sent to identify the MAC address of the remote machine.

The arp request message (who is A.A.A.A tell B.B.B.B where the two are IP addresses) is broadcast on the local area network with an Ethernet protocol type 0x806. The packet is discarded by all the machines except the target machine which responds with an arp response message (A.A.A.A is hh:hh:hh:hh:hh:hh where hh:hh:hh:hh:hh:hh is the Ethernet source address). This packet is unicast to the machine with IP address B.B.B.B. Since the arp request message included the hardware address (Ethernet source address) of the requesting computer, target machine doesn't require another arp message to figure it out.

3.15 Reverse Address Resolution Protocol

RARP is a protocol by which a physical machine in a local area network can request to learn its IP address from a gateway server's Address Resolution Protocol table or cache. This is needed since the machine may not have permanently attacded disk where it can store its IP address permanently. A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Medium Access Control - MAC) addresses to corresponding Internet Protocol addresses. When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the router table, the RARP server will return the IP address to the machine which can store it for future use.

Both the machine that issues the request and the server that responds use physical network addresses during their brief communication. Usually, the requester does not know the physical address. So, the request is broadcasted to all the machines on the network. Now, the requester must identify istelf uniquely to the server. For this either CPU serial number or the machine's physical network address can be used. But using the physical address as a unique id has two advantages.

- These addresses are always available and do not have to be bound into bootstrap code.
- Because the identifying information depends on the network and not on the CPU vendor, all machines on a given network will supply unique identifiers.

Request:
Like an ARP message, a RARP message is sent from one machine to the another encapsulated in the data portion of a network frame. An ethernet frame carrying a RARP request has the usual preamle, Ethernet source and destination addresses, and packet type fields in front of the frame. The frame conatins the value 8035 (base 16) to identify the contents of the frame as a RARP message. The data portion of the frame contains the 28-octet RARP message. The sender braodcasts a RARP request that specifies itself as both the sender and target machine, and supplies its physical network address in the target hardware address field. All machines on the network receive the request, but only those authorised to supply the RARP services process the request and send a reply, such machines are known informally as RARP servers. For RARP to succeed, the network must contain at least one RARP server.
Reply:
Servers answers request by filling in the target protocol address field, changing the message type from request to reply, and sending the reply back directly to the machine making the request.

Timing RARP Transactions
Since RARP uses the physical network directly, no other protocol software will time the

response or retransmit the request. RARP software must handle these tasks. Some workstations that rely on RARP to boot, choose to retry indefinitely until the receive a response. Other implementations announce failure after only a few tries to avoid flooding the network with unnecessary broadcast.

Mulitple RARP Servers Advantage: More reliability. Diadvantage: Overloading may result when all servers respond. So, to get away with disadvantage we have primary and secondary servers. Each machine that makes RARP request is assigned a primary server. Normally, the primary server responds but if it fails, then requester may time out and rebroadcast the request.Whenever a secondary server receives a second copy of the request within a short time of the first, it responds. But, still there might be a problem that all secondary servers respond, thus overloading the network. So, the solution adopted is to avoid having all secondary servers transmit responses simultaneously. Each secondary server that receives the request computes a random delay and then sends a response.

Drawbacks of RARP

- Since it operates at low level, it requires direct addresss to the network which makes it difficult for an application programmer to build a server.
- It doesn't fully utilizes the capability of a network like ethernet which is enforced to send a minimum packet size since the reply from the server contains only one small piece of information, the 32-bit internet address.

RARP is formally described in RFC903.

3.16. Congestion Control Algorithms

As Internet can be considered as a Queue of packets, where transmitting nodes are constantly adding packets and some of them (receiving nodes) are removing packets from the queue. So, consider a situation where too many packets are present in this queue (or internet or a part of internet), such that constantly transmitting nodes are pouring packets at a higher rate than receiving nodes are removing them. This degrades the performance, and such a situation is termed as Congestion. Main reason of congestion is more number of packets into the network than it can handle. So, the objective of congestion control can be summarized as to maintain the number of packets in the network below the level at which performance falls off dramatically. The nature of a Packet switching network can be summarized in following points:
• A network of queues

• At each node, there is a queue of packets for each outgoing channel

• If packet arrival rate exceeds the packet transmission rate, the queue size grows without bound

• When the line for which packets are queuing becomes more than 80% utilized, the queue length grows alarmingly

When the number of packets dumped into the network is within the carrying capacity, they all are delivered, expect a few that have too be rejected due to transmission errors). And then the

number delivered is proportional to the number of packets sent. However, as traffic increases too far, the routers are no longer able to cope, and they begin to lose packets. This tends to make matter worse. At very high traffic, performance collapse completely, and almost no packet is delivered. In the following sections, the causes of congestion, the effects of congestion and various congestion control techniques are discussed in detail

3.16.1. Causes Of Congestion

Congestion can occur due to several reasons. For example, if all of a sudden a stream of packets arrive on several input lines and need to be out on the same output line, then a long queue will be build up for that output. If there is insufficient memory to hold these packets, then packets will be lost (dropped). Adding more memory also may not help in certain situations. If router have an infinite amount of memory even then instead of congestion being reduced, it gets worse; because by the time packets gets at the head of the queue, to be dispatched out to the output line, they have already timed-out (repeatedly), and duplicates may also be present. All the packets will be forwarded to next router up to the destination, all the way only increasing the load to the network more and more. Finally when it arrives at the destination, the packet will be discarded, due to time out, so instead of been dropped at any intermediate router (in case memory is restricted) such a packet goes all the way up to the destination, increasing the network load throughout and then finally gets dropped there. Slow processors also cause Congestion. If the router CPU is slow at performing the task required for them (Queuing buffers, updating tables, reporting any exceptions etc.), queue can build up even if there is excess of line capacity. Similarly, LowBandwidth lines can also cause congestion. Upgrading lines but not changing slow processors, or vice-versa, often helps a little; these can just shift the bottleneck to some other point. The real problem is the mismatch between different parts of the system. Congestion tends to feed upon itself to get even worse. Routers respond to overloading by dropping packets. When these packets contain TCP segments, the segments don't reach their destination, and they are therefore left unacknowledged, which eventually leads to timeout and retransmission. So, the major cause of congestion is often the bursty nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion because other causes just act as an enzyme which boosts up the congestion when the traffic is bursty (i.e., other causes just add on to make the problem more serious, main cause is the bursty traffic). This means that when a device sends a packet and does not receive an acknowledgment from the receiver, in most the cases it can be assumed that the packets have been dropped by intermediate devices due to congestion. By detecting the rate at which segments are sent and not acknowledged, the source or an intermediate router can infer the level of congestion on the network. In the following section we shall discuss the ill effects of congestion.

3.16.2 Effects of Congestion

Congestion affects two vital parameters of the network performance, namely throughput and delay. In simple terms, the throughput can be defined as the percentage utilization of the network capacity. Figure shows how throughput is affected as offered loadincreases. Initially throughput increases linearly with offered load, because utilization of the network increases.

However, as the offered load increases beyond certain limit, say 60% of the capacity of the network, the throughput drops. If the offered load increases further, a point is reached when not a single packet is delivered to any destination, which is commonly known as deadlock situation. There are three curves in Fig. the ideal one corresponds to the situation when all the packets introduced are delivered to their destination up to the maximum capacity of the network. The second one corresponds to the situation when there is no congestion control. The third one is the case when some congestion control technique is used. This prevents the throughput collapse, but provides lesser throughput than the ideal condition due to overhead of the congestion control technique. The delay also increases with offered load, as shown in Fig.. And no matter what technique is used for congestion control, the delay grows without bound as the load approaches the capacity of the system. It may be noted that initially there is longer delay when congestion control policy is applied. However, the network without any congestion control will saturate at a lower offered load

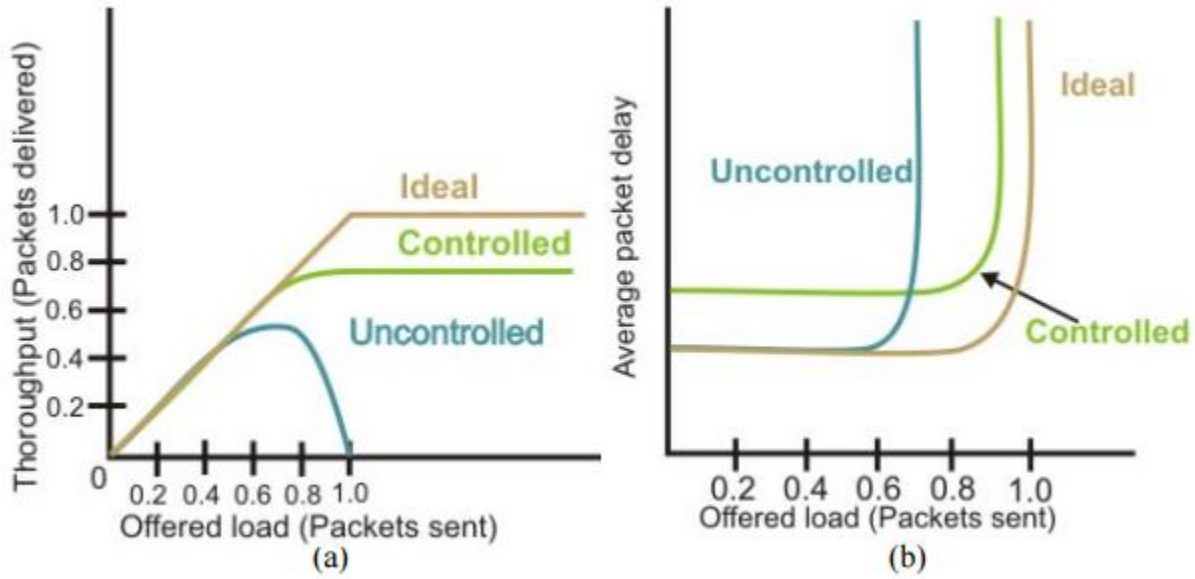


Figure 3.20 (a) Effect of congestion on throughput (b) Effect of congestion on delay

3.17 Congestion Control Techniques

Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. As shown in Fig., the congestion control techniques can be broadly classified two broad categories:

• Open loop: Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State.

• Close loop: Protocols that allow system to enter congested state, detect it, and remove it

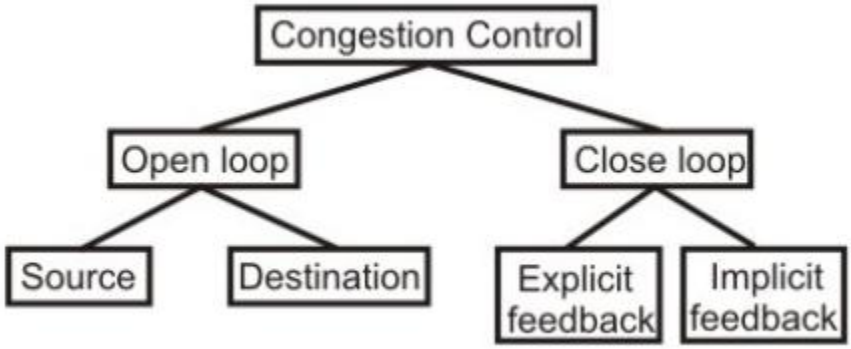


Figure 3.21 Congestion control categories

The first category of solutions or protocols attempt to solve the problem by a good design, at first, to make sure that it doesn't occur at all. Once system is up and running midcourse corrections are not made. These solutions are somewhat static in nature, as the policies to control congestion don't change much according to the current state of the system. Such Protocols are also known as Open Loop solutions. These rules or policies include deciding upon when to accept traffic, when to discard it, making scheduling decisions and so on. Main point here is that they make decision without taking into consideration the current state of the network. The open loop algorithms are further divided on the basis of whether these acts on source versus that act upon destination. The second category is based on the concept of feedback. During operation, some system parameters are measured and feed back to portions of the subnet that can take action to reduce the congestion. This approach can be divided into 3 steps:

• Monitor the system (network) to detect whether the network is congested or not and what's the actual location and devices involved.

• To pass this information to the places where actions can be taken

• Adjust the system operation to correct the problem.

These solutions are known as Closed Loop solutions. Various Metrics can be used to monitor the network for congestion. Some of them are: the average queue length, number of packets that are timed-out, average packet delay, number of packets discarded due to lack of buffer space, etc. A general feedback step would be, say a router, which detects the congestion send special packets to the source (responsible for the congestion) announcing the problem. These extra packets increase the load at that moment of time, but are necessary to bring down the congestion at a later time. Other approaches are also used at times to curtail down the congestion. For example, hosts or routers send out probe packets at regular intervals to explicitly ask about the congestion and source itself regulate its transmission rate, if congestion is detected in the network. This kind of approach is a pro-active one, as source tries to get knowledge about congestion in the network and act accordingly.

Yet another approach may be where instead of sending information back to the source an intermediate router which detects the congestion send the information about the congestion to rest of the network, piggy backed to the outgoing packets. This approach will in no way put an extra load on the network (by not sending any kind of special packet for feedback). Once the congestion has been detected and this information has been passed to a place where the action needed to be done, then there are two basic approaches that can overcome the problem. These are: either to increase the resources or to decrease the load. For example, separate dial-up lines or alternate links can be used to increase the bandwidth between two points, where congestion occurs. Another example could be to decrease the rate at which a particular sender in transmitting packets out into the network. The closed loop algorithms can also be divided into two categories, namely explicit feedback and implicit feedback algorithms. In the explicit approach, special packets are sent back to the sources to curtail down the congestion. While in implicit approach, the source itself acts pro-actively and tries to deduce the existence of congestion by making local observations. In the following sections we shall discuss about some of the popular algorithms from the above categories.

3.17.1 Leaky Bucket Algorithm

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. This scenario is depicted in figure Once the bucket is full, any additional water entering it spills over the sides and is lost (i.e. it doesn't appear in the output stream through the hole underneath). The same idea of leaky bucket can be applied to packets, as shown in Fig. Conceptually each network interface contains a leaky bucket. And the following steps are performed:

• When the host has to send a packet, the packet is thrown into the bucket.

• The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.

• Bursty traffic is converted to a uniform traffic by the leaky bucket.

• In practice the bucket is a finite queue that outputs at a finite rate. This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.

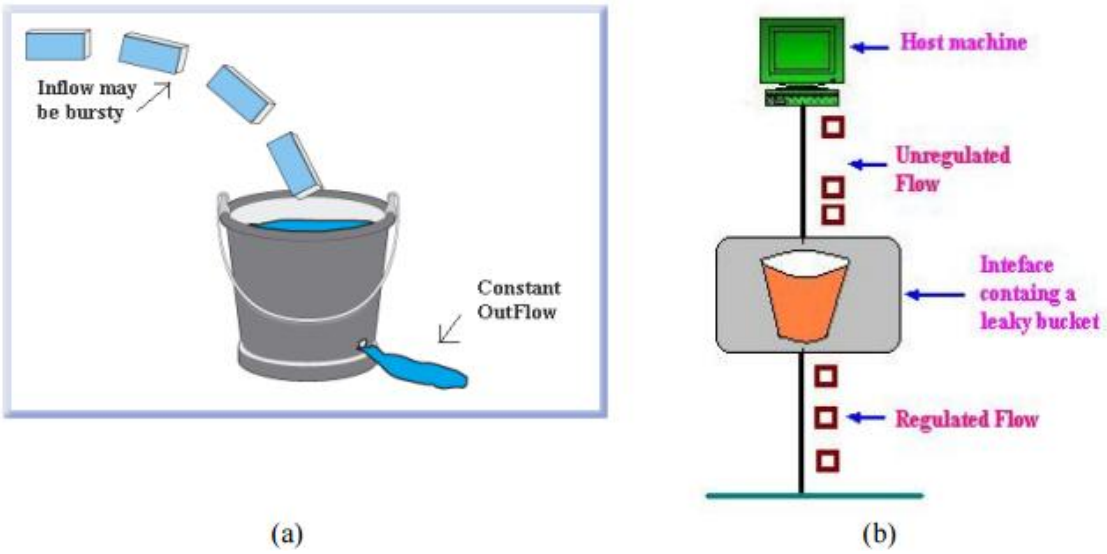


Figure 3.22 (a) Leaky bucket (b) Leaky bucket implementation

3.17.2 Token Bucket Algorithm

The leaky bucket algorithm described above, enforces a rigid pattern at the output stream, irrespective of the pattern of the input. For many applications it is better to allow the output to speed up somewhat when a larger burst arrives than to loose the data. Token Bucket algorithm provides such a solution. In this algorithm leaky bucket holds token, generated at regular intervals. Main steps of this algorithm can be described as follows: ƒ In regular intervals tokens are thrown into the bucket. ƒ The bucket has a maximum capacity. ƒ If there is a ready packet, a token is removed from the bucket, and the packet is send. ƒ If there is no token in the bucket, the packet cannot be send. Figure shows the two scenarios before and after the tokens present in the bucket have been consumed. In Fig.3.17.2 the bucket holds two tokens, and three packets are waiting to be sent out of the interface, in Fig. two packets have been sent out by consuming two tokens, and 1 packet is still left. The token bucket algorithm is less restrictive than the leaky bucket algorithm, in a sense that it allows bursty traffic. However, the limit of burst is restricted by the number of tokens available in the bucket at a particular instant of time. The implementation of basic token bucket algorithm is simple; a variable is used just to count the tokens. This counter is incremented every t seconds and is decremented whenever a packet is sent. Whenever this counter reaches zero, no further packet is sent out as shown in Fig.3.17.2.

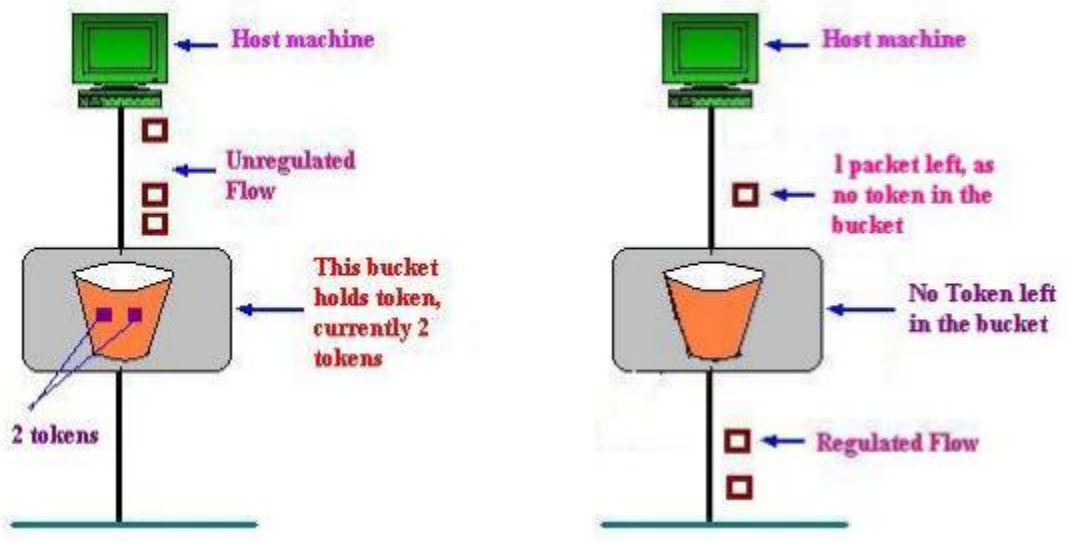


Fig 3.23 Token bucket holding two tokens, before packets are send out, (b) Token bucket after two packets are send, one packet still remains as no token is left

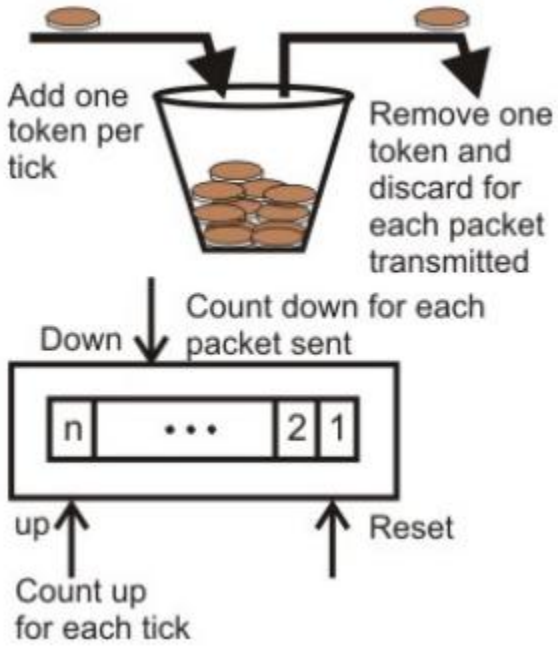


Figure 3.24 Implementation of the Token bucket algorithm

3.Congestion control in virtual Circuit

Till now we have discussed two open loop algorithms, where the policy decisions are made in the beginning, irrespective of the current state. Both leaky bucket algorithm and token bucket algorithm are open loop algorithms.

In this section we shall have a look at how the congestion is tackled in a virtualcircuit network. Admission control is one such closed-loop technique, where action is taken once congestion is detected in the network. Different approaches can be followed:

• Simpler one being: do not set-up new connections, once the congestion is signaled. This type of approach is often used in normal telephone networks. When the exchange is overloaded, then no new calls are established.

• Another approach, which can be followed is: to allow new virtual connections, but route these carefully so that none of the congested router (or none of the problem area) is a part of this route.

• Yet another approach can be: To negotiate different parameters between the host and the network, when the connection is setup. During the setup time itself, Host specifies the volume and shape of traffic, quality of service, maximum delay and other parameters, related to the traffic it would be offering to the network. Once the host specifies its requirement, the resources needed are reserved along the path, before the actual packet follows.

Choke Packet Technique

The choke packet technique, a closed loop control technique, can be applied in both virtual circuit and datagram subnets. Each router monitors its resources and the utilization at each of its output line. There is a threshold set by the administrator, and whenever any of the resource utilization crosses this threshold and action is taken to curtail down this. Actually each output line has a utilization associated with it, and whenever this utilization crosses the threshold, the output line enters a “warning” state. If so, the router sends a choke packet back to the source, giving it a feedback to reduce the traffic. And the original packet is tagged (a bit is manipulated in the header field) so that it will not generate other choke packets by other intermediate router, which comes in place and is forwarded in usual way. It means that the first router (along the way of a packet), which detects any kind of congestion, is the only one that sends the choke packets. When the source host gets the choke packet, it is required to reduce down the traffic send out to that particular destination (choke packet contains the destination to which the original packet was send out). After receiving the choke packet the source reduces the traffic by a particular fixed percentage, and this percentage decreases as the subsequent choke packets are received. Figure depicts the functioning of choke packets. For Example, when source A receives a choke packet with destination B at first, it will curtail down the traffic to destination B by 50%, and if again after affixed duration of time interval it receives the choke packet again for the same destination, it will further curtail down the traffic by 25% more and so on. As stated above that a source will entertain another subsequent choke packet only after a fixed interval of time, not before that. The reason for this is that when the first choke packet arrives at that point of time other packets destined to the same destination would also be there in the network and they will generate other choke packets too, the host should ignore these choke packets which refer to the same destination for a fixed time interval.

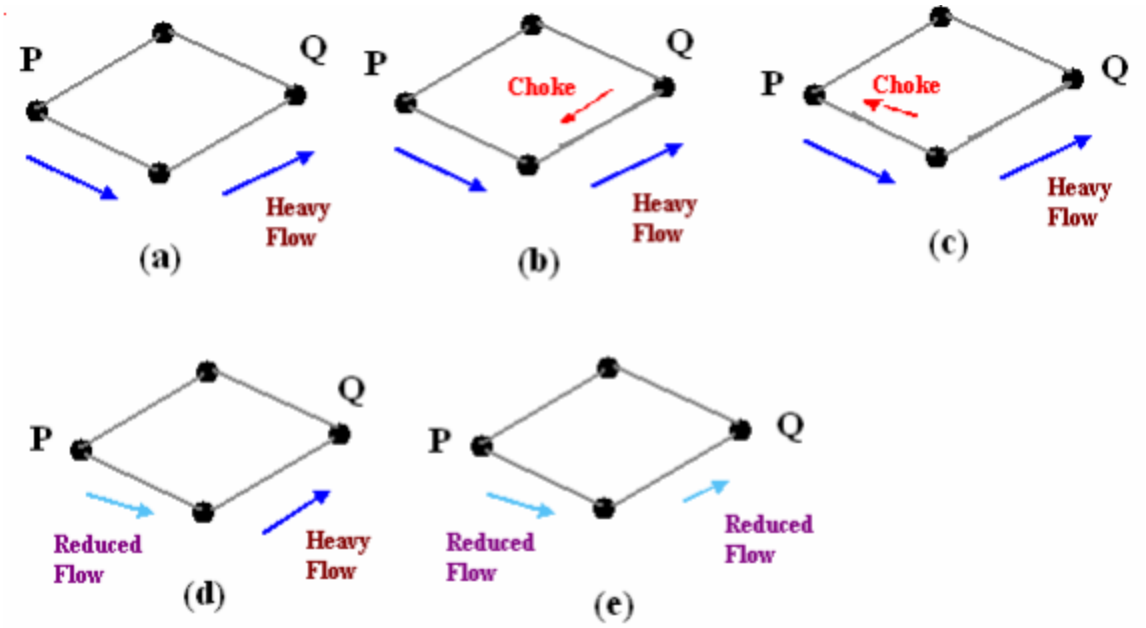


Figure 3.25 Depicts the functioning of choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to P, (c) Choke packet reaches P, (d) P reduces the flow and send a reduced flow out, (e) Reduced flow reaches node Q

Hop-by Hop Choke Packets

This technique is an advancement over Choked packet method. At high speed over long distances, sending a packet all the way back to the source doesn't help much, because by the time choke packet reach the source, already a lot of packets destined to the same original destination would be out from the source. So to help this, Hop-by-Hop Choke packets are used. In this approach, the choke packet affects each and every intermediate router through which it passes by. Here, as soon as choke packet reaches a router back to its path to the source, it curtails down the traffic between those intermediate routers. In this scenario, intermediate nodes must dedicate few more buffers for the incoming traffic as the outflow through that node will be curtailed down immediately as choke packet arrives it, but the input traffic flow will only be curtailed down when choke packet reaches the node which is before it in the original path. This method is illustrated in Fig.. As compared to choke packet technique, hop-by-hop choke packet algorithm is able to restrict the flow rapidly. As can been seen from Figures and one-step reduction is seen in controlling the traffic, this single step advantage is because in our example there is only one intermediate router. Hence, in a more complicated network, one can achieve a significant advantage by using hop-by-hop choke packet method.

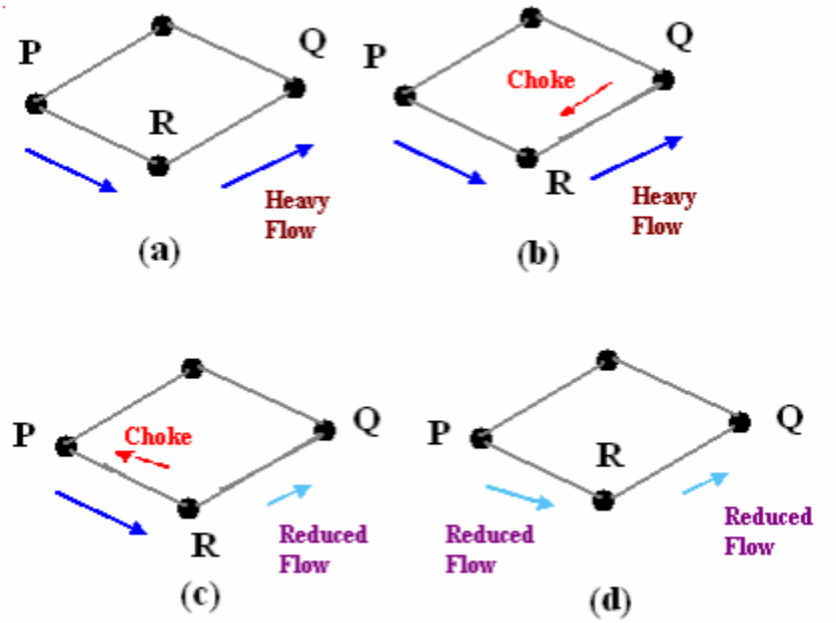


Figure 3.26 Depicts the functioning of Hop-by-Hop choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to P, (c) Choke packet reaches R, and the flow between R and Q is curtail down, Choke packer reaches P, and P reduces the flow out

Load Shedding

Another simple closed loop technique is Load Shedding; it is one of the simplest and more effective techniques. In this method, whenever a router finds that there is congestion in the network, it simply starts dropping out the packets. There are different methods by which a host can find out which packets to drop. Simplest way can be just choose the packets randomly which has to be dropped. More effective ways are there but they require some kind of cooperation from the sender too. For many applications, some packets are more important than others. So, sender can mark the packets in priority classes to indicate how important they are. If such a priority policy is implemented than intermediate nodes can drop packets from the lower priority classes and use the available bandwidth for the more important packets.

Slow Start - a Pro-active technique

This is one of the pro-active techniques, which is used to avoid congestion. In the original implementation of TCP, as soon as a connection was established between two devices, they could each go “hog wild”, sending segments as fast as they liked as long as there was room in the other devices receive window. In a busy internet, the sudden appearance of a large amount of new traffic could aggravate any existing congestion. To alleviate this, modern TCP devices are restrained in the rate at which they initially send segments. Each sender is at first restricted to sending only an amount of data equal to one “full-sized” segment—that is, equal to the MSS (maximum segment size) value for the connection. Each time an acknowledgment is received, the amount of data the device can send is increased by the size of another full-sized segment. Thus, the device “starts slow” in terms of how much data it can send, with the amount it sends

increasing until either the full window size is reached or congestion is detected on the link. In the latter case, the congestion avoidance feature is used. When potential congestion is detected on a TCP link, a device responds by throttling back the rate at which it sends segments. A special algorithm is used that allows the device to drop the rate at which segments are sent quickly when congestion occurs. The device then uses the Slow Start algorithm just above to gradually increase the transmission rate back up again to try to maximize throughput without congestion occurring again.

Flow Control versus Congestion control

Flow control is a very important part of regulating the transmission of data between devices, but it is limited in a way that it only considers what is going on within each of the devices on the connection, and not what is happening in devices between them. It relates to the point-point traffic between a given sender and a receiver. Flow control always involves some kind of feedback from receiver to sender to tell sender how things are at other end of the network. Since we are dealing with how TCP works between a typical server and client at layer four, we don't worry about how data gets between them; that's the job of the Internet Protocol at layer three. In practice, what is going on at layer three can be quite important. Considered from an abstract point of view, our server and client may be connected “directly” using TCP, but all the packets we transmit are carried across an internet and routers between different networks. These networks and routers are also carrying data from many other connections and higher-layer protocols. If the internet becomes very busy, the speed at which segments are carried between the endpoints of our connection will be reduced, and they could even be dropped. This is called congestion control. Congestion control has to do with making sure that subnet carry the offered traffic. It is the global issue, involving the behavior of all the hosts, router, link, store and forward mechanism between them in the entire subnet or internet

Quality of Service

Requirements • Techniques for Achieving Good Quality of Service • Integrated Services • Differentiated Services • Label Switching and MPLS

Requirements

How stringent the quality-of-service requirements are

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Techniques for Good QoS

- Over provisioning
- Buffering
- Traffic shaping
- The leak bucket algorithm
- Token bucket algorithm
- Resource reservation
- Admission control
- Proportional routing
- Packet scheduling

3.16 Transport protocol

- In computer networking, the transport layer is a conceptual division of methods in the layered architecture of protocols in the network stack in the Internet Protocol Suite and the Open Systems Interconnection (OSI). The protocols of the layer provide host-to-host communication services for applications.[1] It provides services such as connection-oriented data stream support, reliability, flow control, and multiplexing.
- The details of implementation and semantics of the Transport Layer of the TCP/IP model (RFC 1122),[2] which is the foundation of the Internet, and the Open Systems Interconnection (OSI) model of general networking, are different. In the OSI model the transport layer is most often referred to as Layer 4 or L4, while numbered layers are not used in TCP/IP.
- The best-known transport protocol of TCP/IP is the Transmission Control Protocol (TCP), and lent its name to the title of the entire suite. It is used for connection-oriented transmissions, whereas the connectionless User Datagram Protocol (UDP) is used for simpler messaging transmissions. TCP is the more complex protocol, due to its stateful design incorporating reliable transmission and data stream services. Other prominent protocols in this group are the Datagram Congestion Control Protocol (DCCP) and the Stream Control Transmission Protocol (SCTP).

3.16.1 Transport Service

Transport layer services are conveyed to an application via a programming interface to the transport layer protocols. The services may include the following features:

- Connection-oriented communication: It is normally easier for an application to interpret a connection as a data stream rather than having to deal with the underlying connection-less models, such as the datagram model of the User Datagram Protocol (UDP) and of the Internet Protocol (IP).
- Same order delivery: The network layer doesn't generally guarantee that packets of data will arrive in the same order that they were sent, but often this is a desirable feature. This is usually done through the use of segment numbering, with the receiver passing them to the application in order. This can cause head-of-line blocking.
- Reliability: Packets may be lost during transport due to network congestion and errors. By means of an error detection code, such as a checksum, the transport protocol may check that the data is not corrupted, and verify correct receipt by sending an ACK or NACK message to the sender. Automatic repeat request schemes may be used to retransmit lost or corrupted data.
- Flow control: The rate of data transmission between two nodes must sometimes be managed to prevent a fast sender from transmitting more data than can be supported by the receiving data buffer, causing a buffer overrun. This can also be used to improve efficiency by reducing buffer under run.
- Congestion avoidance: Congestion control can control traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. For example, automatic repeat requests may keep the network in a congested state; this situation can be avoided by adding congestion avoidance to the flow control, including slow-start. This keeps the bandwidth consumption at a low level in the beginning of the transmission, or after packet retransmission.
- Multiplexing: Ports can provide multiple endpoints on a single node. For example, the name on a postal address is a kind of multiplexing, and distinguishes between different recipients of the same location. Computer applications will each listen for information on their own ports, which enables the use of more than one network service at the same time. It is part of the transport layer in the TCP/IP model, but of the session layer in the OSI model.

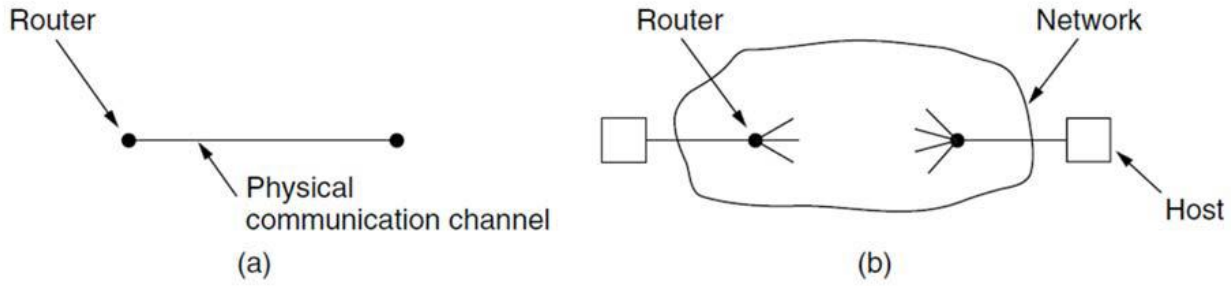
- The transport layer is responsible for delivering data to the appropriate application process on the host computers. This involves statistical multiplexing of data from different application processes, i.e. forming data packets, and adding source and destination port numbers in the header of each transport layer data packet. Together with the source and destination IP address, the port numbers constitutes a network socket, i.e. an identification address of the process-to-process communication. In the OSI model, this function is supported by the session layer.
- Some transport layer protocols, for example TCP, but not UDP, support virtual circuits, i.e. provide connection oriented communication over an underlying packet oriented datagram network. A byte-stream is delivered while hiding the packet mode communication for the application processes. This involves connection establishment, dividing of the data stream into packets called segments, segment numbering and reordering of out-of order data.

Finally, some transport layer protocols, for example TCP, but not UDP, provide end-to-end reliable communication, i.e. error recovery by means of error detecting code and automatic repeat request (ARQ) protocol. The ARQ protocol also provides flow control, which may be combined with congestion avoidance.

- UDP is a very simple protocol, and does not provide virtual circuits, nor reliable communication, delegating these functions to the application program. UDP packets are called datagrams, rather than segments.
- TCP is used for many protocols, including HTTP web browsing and email transfer. UDP may be used for multicasting and broadcasting, since retransmissions are not possible to a large amount of hosts. UDP typically gives higher throughput and shorter latency, and is therefore often used for real-time multimedia communication where packet loss occasionally can be accepted, for example IP-TV and IP-telephony, and for online computer games.
- Many non-IP-based networks, such as X.25, Frame Relay and ATM, implement the connection-oriented communication at the network or data link layer rather than the transport layer. In X.25, in telephone network modems and in wireless communication systems, reliable node-to-node communication is implemented at lower protocol layers.
- The OSI connection-mode transport layer protocol specification defines five classes of transport protocols: TP0, providing the least error recovery, to TP4, which is designed for less reliable networks.

3.16.2 Elements of transport protocol

- Transport protocol similar to data link protocols
- Both do error control and flow control
- However, significant differences exist

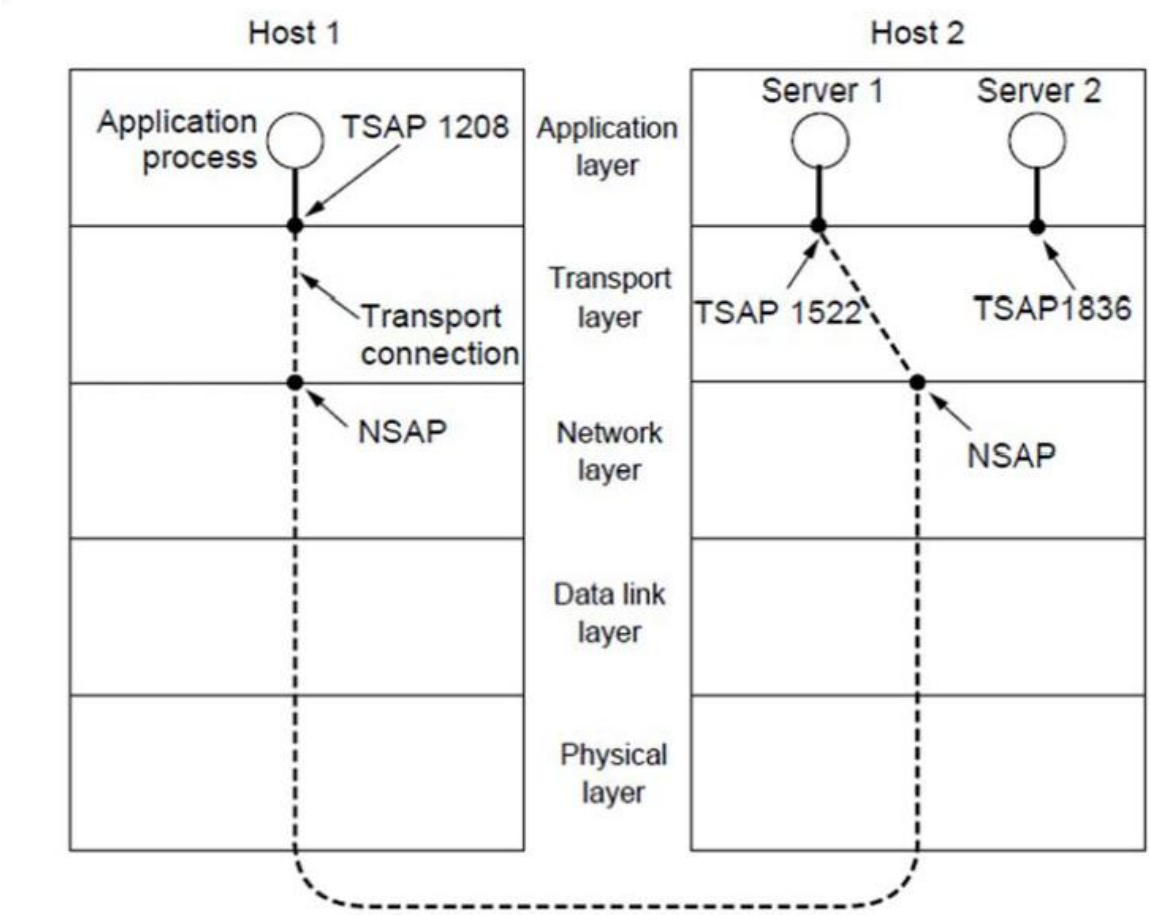


Environment of the data link layer Environment of the transport layer

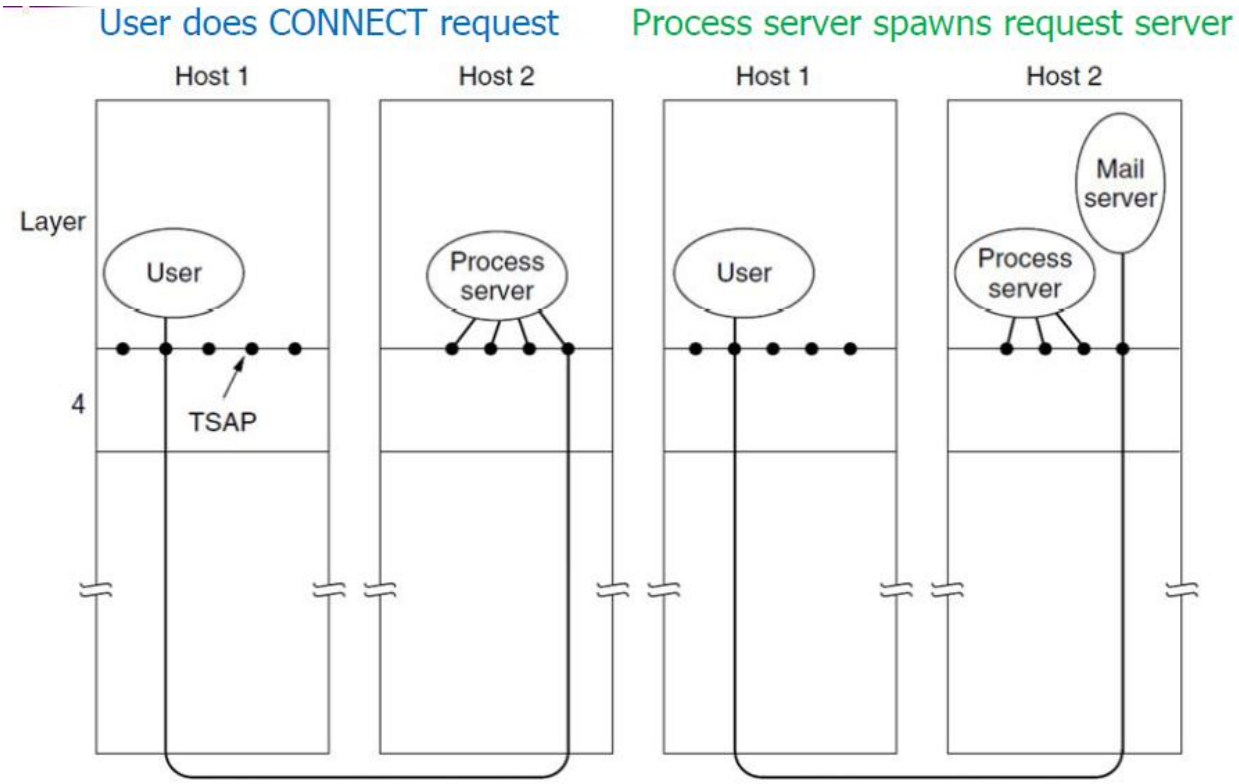
13.16.2.1 Addressing

- Specify which host process to connect to
- TSAP: Transport Service Access Point
- In TCP, UDP, called ports

- Analogy: NSAP. Example: IP address
- Client or server app attaches to TSAP
- Connections run through NSAP
- TSAP to distinguish endpoints sharing NSAP



- Many server processes used only rarely
- Waste if each process listen to TSAP all day
- Instead, use initial connection protocol
- Spec process server listens all known TSAP
- Act as proxy for lightly used servers
- e.g. inted, xinetd on UNIX



Connection Establishment
- Sounds easy; surprisingly tricky!
- Just send REQUEST, wait for ACCEPTED?
- Can lose, delay, corrupt, duplicate packets
- Duplicate may transfer bank money again!
- Protocols must work correct all cases
- Implemented efficiently in common cases
- Main problem is delayed duplicates
- Cannot prevent; must deal with (reject)

Solutions for delayed duplicates
- Not reuse transport address (TSAP)
- difficult to connect to process
- Give each connection unique ID
- seq # chosen by initiating party
- update table listing obsolete connections
- check new connections against table
- requires maintain certain amount of history
- if machine crashes, no longer identify old con

- To simplify problem, restrict packet lifetime
- restricted network design: prevent looping
- hop counter in each packet: - 1 at each hop
- timestamp in each packet: clock must be synced

 Must also guarantee ACKs are dead
 Assume a value T of max packet lifetime
 T sec after packet sent, sure traces are gone
 In the Internet, T is usually 120 seconds.

New method with packet lifetime bounded
 Label segments with seq # not reused in T
 T and packet rate determine size of seq #s
 1 packet w given seq # may be outstanding
 Duplicates may still occur, but discarded dst
 Not possible to have delayed duplicate old packet with same seq # accepted at dest

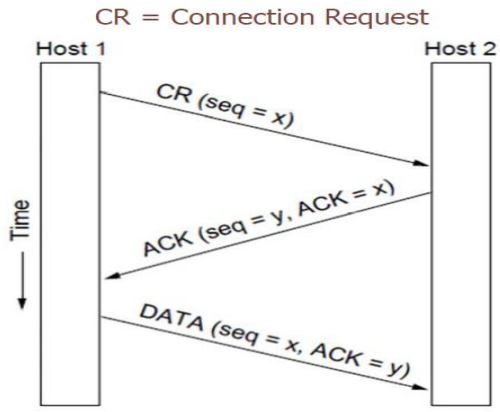
How to deal with losing memory after crash?

 Each host has time- of- day clock
 clocks at different host need not be synced
 binary counter increments uniform intervals
 no. of bits must be ≥ of seq #
 clock must be running even if host goes down
 Initial seq # (k- bits) ← low k- bits of clock
 Seq space must be so large
 by time # wrap, old pkts w same # are long gone

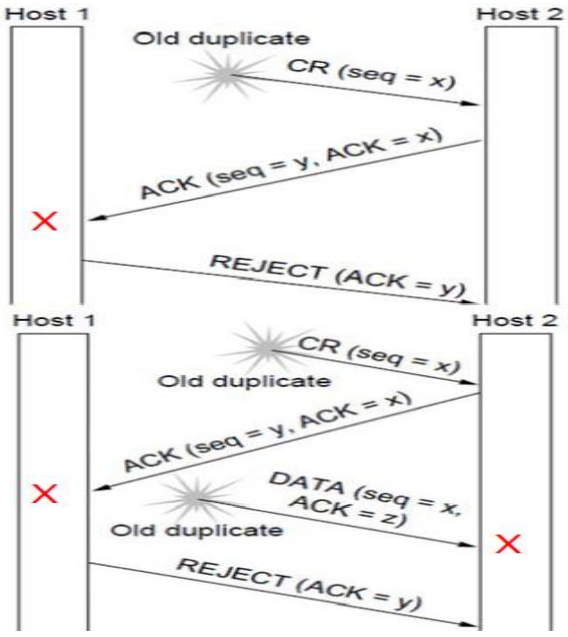
Clock method work within connection
 Host don't remember # across connections
 Can't know if CONN REQUEST with initial seq # is a duplicate of a recent connection
 To solve this, use three- way handshake
 Check with other peer that con req is current
 Used in TCP, with 32- bit seq #
 Clock not used in TCP; attacker can predict

Normal Procedure

 H1 choses initial s# x
 H2 replies
 ACKs x
 announce own s# y
 H1replies
 ACKs y
 with 1st data segment

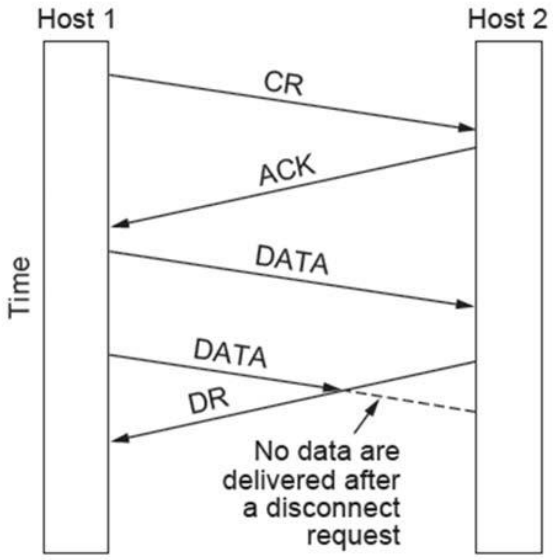


Abnormal situations
 Delayed duplicate CR
 H2 sends ACK to H1
 H1 rejects
 H2 knows it was tricked
16
 Worst case
DD CR, old ACK floating
 H2 gets delayed CR, replies
 H1 rejects
 H2 gets old DATA, discards
(z received instead of y)



Connection Release

- Easier than establish
- However, some pitfalls
- Asymmetric release
- each con term separately
- abrupt; may cause data loss
- better protocol needed



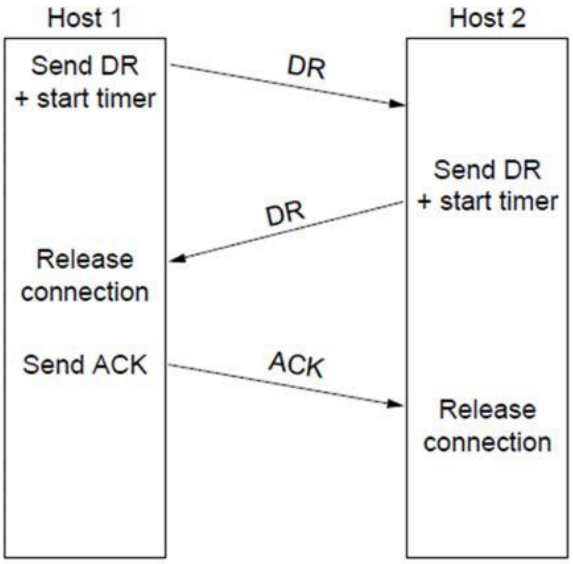
Symmetric release
- Each direction is released independently
- Can receive data after sending DISCONNECT
- H1: I am done, are you done too?
- H2: I am done too, goodbye
- Two- army problem: unreliable channel

- **Two army problem**

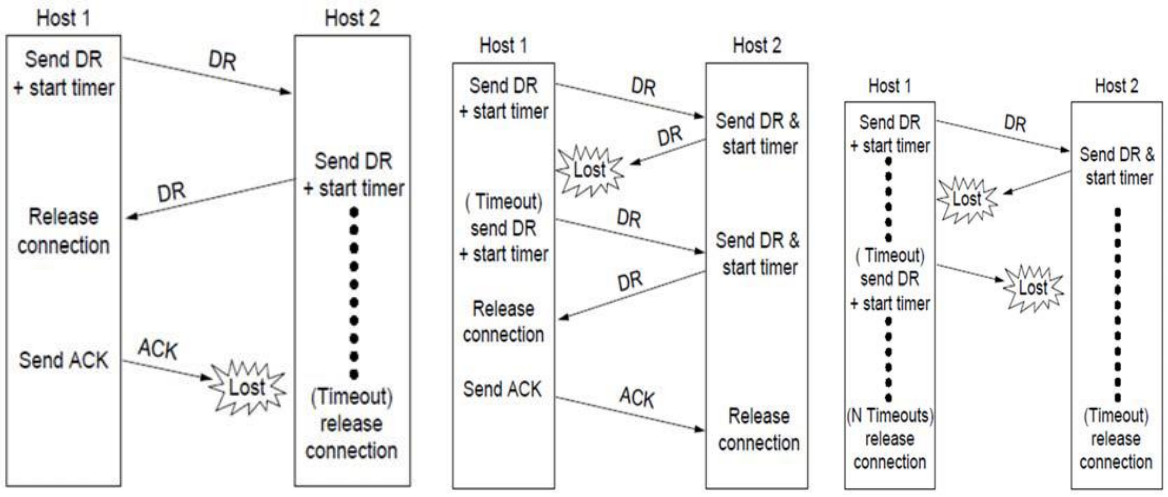
- each blue army < white army, but together are larger
- need to sync attack
- however, only com channel is the valley (unreliable)
- 3- way handshake? B1 can't know ACK arrived
- making 4- way handshake doesn't help either
- Let each side independently decide its done
- Easier to solve

Normal release sequence
- H1 send DR, start timer
- H2 responds with DR
- when H1 recv DR, release

 when H2 recv ACK, release



Error cases, handled by timers, retransmissions



Final ACK lost:
Many lost DRs
Host 2 times out

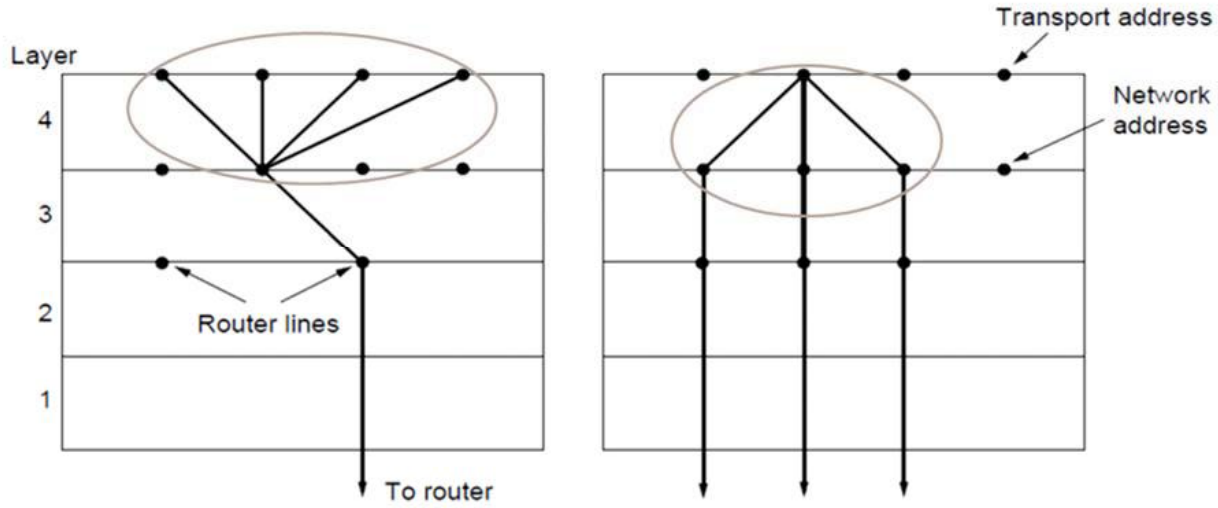
Lost DR:H1 starts over

Extreme:
both release after N

 Protocol usually suffices; can fail in theory
 after N lost attempts; half open connection
 Not allowing give up, can go on forever
 To kill half open connections, automatically

disconnect if no received segments in X sec
 Must have timer reset after each segment
 Send dummy segments to keep con alive
 TCP normally does symmetric close, with each side independently close ½ con w FIN

Multiplexing
 Transport, network sharing can either be:
 Multiplexing: connections share a network address
 Inverse multiplexing: addresses share a connection



PART – A
1. Write down the design issue of network layer.
2. What is meant by Routing?
3. What is congestion?
4. What is Static and Dynamic Routing?
5. What is Centralized routing and isolated routing?
6. What is flooding?
7. Write the concept behind flow based routing.
8. What are the Different Broadcast routing techniques?
9. Give the advantages of multipath routing?
10. What is reverse path forwarding and state its advantage.
11. What are the various congestion control techniques?
12. What is leaky bucket algorithm?
13. What is Token bucket algorithm?
14. What is choke packet, How congestion is controlled over here?
15. What is reliability?
16. What is vulnerable time in CSMA, CSMA/CD,CSMA/CA? Differentiate

PART – B

1. Explain about the various multiple access protocols.
2. Give short notes on Bluetooth.
3. Describe in detail about Ethernet.
4. Discuss the architecture of IEEE 802.11 in detail.
5. What is Routing, Explain Shortest Path Routing algorithm with example.
6. Explain Distance Vector & Link state routing algorithm.
   Explain the following routing algorithm with example.
   a. Flooding routing algorithm.
   b. Hierarchical routing algorithm.
7. What is congestion? What are the different methods for controlling congestion?