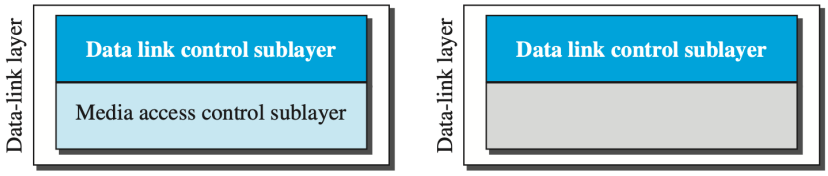
Short notes on Computer Networks and Data Communication

Topics:
Data Link Layer
Error Control: Error Detection, Parity Checks, and CRC,
Coding techniques
Framing Issues: Byte, Bit, and Length-oriented framing
Media Access Control (MAC) and LANs ALOHA, Slotted ALOHA
CSMA, CSMA/CD CSMA/CA, HDLC Protocol, and Operation
Ethernet Protocol, Frame Structure, Fast Ethernet Gigabit Ethernet
IEEE 802.Y LAN Standard, devices

DATA LINK SUBLAYERS

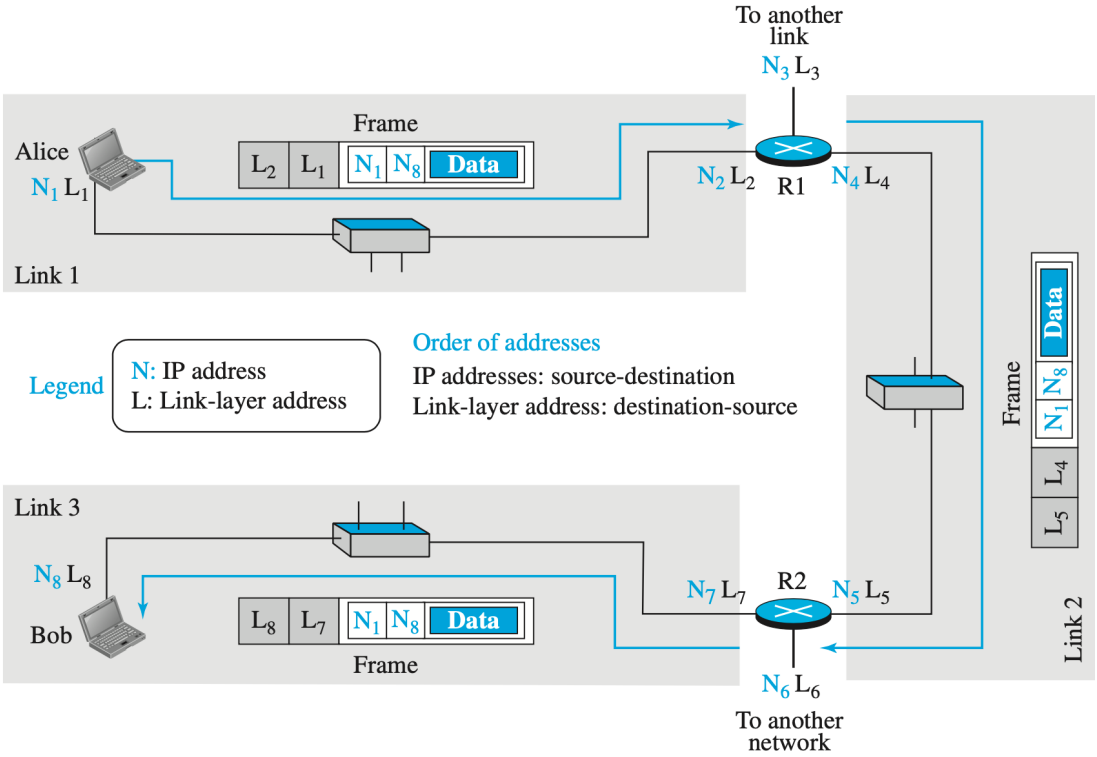
To better understand the link layer's functionality and services, we can divide the data-link layer into two sublayers: data link control (DLC) and media access control (MAC).



a. Data-link layer of a broadcast link b. Data-link layer of a point-to-point link

LINK-LAYER ADDRESSING

We need to remember that the IP addresses in a datagram should not be changed. If the destination IP address in a datagram change, the packet never reaches its destination; if the source IP address in a datagram change, the destination host or a router can never communicate with the source if a response needs to be sent back or an error needs to be reported back to the source. Therefore, for a link, more importantly, within a link domain, we need another addressing mechanism in a connectionless **internetwork**: the link-layer addresses of the two nodes. A link-layer address is sometimes called a link address, sometimes a physical address, and sometimes a MAC address.



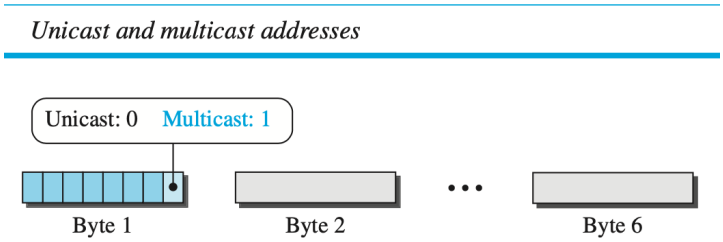
In the figure above, we have three links and two routers. We also have shown only two hosts: Alice (source) and Bob (destination). For each host, we have shown two addresses: the IP addresses (N) and the link-layer addresses (L). Note that a router has as many pairs of addresses as the number of links the router is connected to. We have shown three frames, one in each link. Each frame carries the same datagram with the same source

and destination addresses (N1 and N8), but the link-layer addresses of the frame change from link to link. In link 1, the link-layer addresses are L1 and L2. In link 2, they are L4 and L5. In link 3, they are L7 and L8.

THREE TYPES OF ADDRESSES

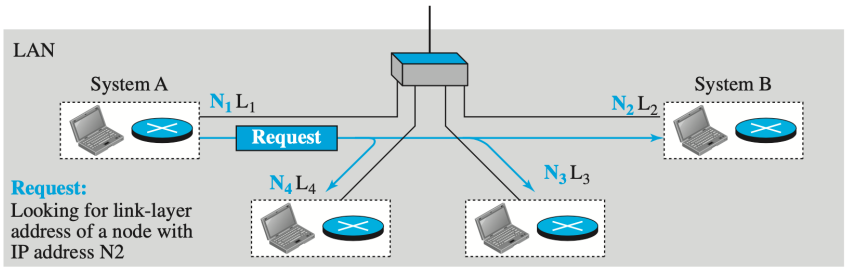
Some link-layer protocols define three types of addresses: unicast, multicast, and broadcast.

1. Unicast Address
   Each host or interface of a router is assigned a unicast address. Unicasting means one-to-one communication. A frame with a unicast address destination is destined only for one entity in the link.
   Example: A2:34:45:11:92:F1
2. Multicast Address
   Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link).
   Example: A3:34:45:11:92:F1
3. Broadcast Address
   Some link-layer protocols define a broadcast address. Broadcasting means one-to-all communication. A frame with a destination broadcast address is sent to all entities in the link.
   Example: FF:FF:FF:FF:FF:FF

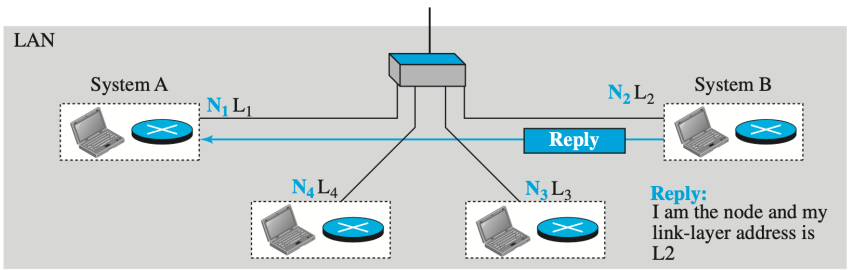


ADDRESS RESOLUTION PROTOCOL (ARP)

Anytime a node has an IP datagram to send to another node in a link; it has the IP address of the receiving node. The source host knows the IP address of the default router. Each router except the last one in the path gets the IP address of the next router by using its for-warding table. The last router knows the IP address of the destination host. However, the IP address of the next node does not help move a frame through a link; we need the link-layer address of the next node. This is the time when the Address Resolution Protocol (ARP) becomes helpful. ARP accepts an IP address from the IP protocol, maps it to the corresponding link-layer address, and passes it to the data-link layer. Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the sender's link layer, IP addresses, and the receiver's IP address. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer **broadcast** address. Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and link-layer addresses. The packet is **unicast** directly to the node that sent the request packet.

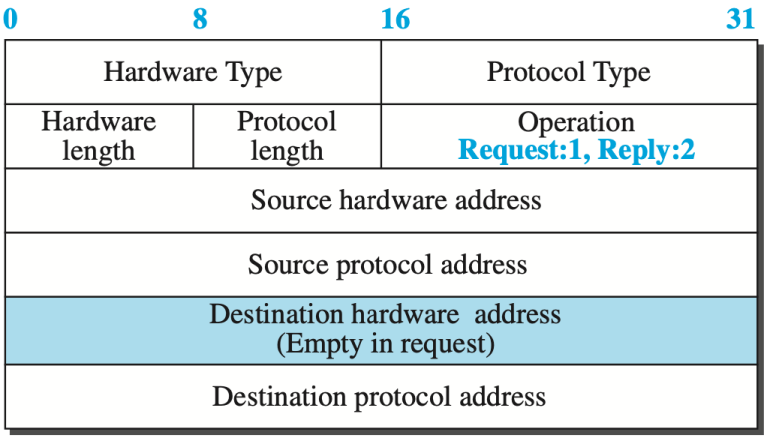


a. ARP request is broadcast



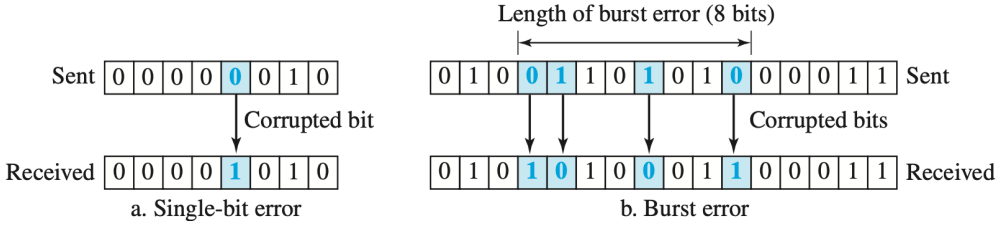
b. ARP reply is unicast

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



TYPES OF ERRORS

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

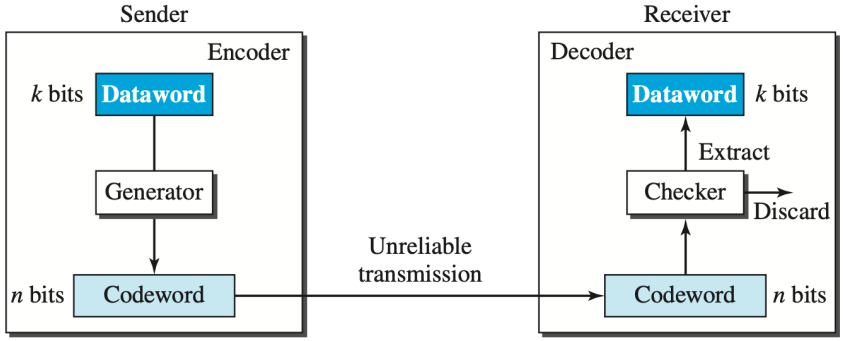


Redundancy

The central concept in detecting or correcting errors is redundancy. To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits. Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits.

BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords. The generic process of error detection using coding is shown below:



ERROR DETECTION CODES

Hamming Distance

One of the **central concepts** in coding for error control is the idea of the Hamming design. The Hamming distance between two words (of the same size) *is the number of differences between the corresponding bits*. We show the Hamming distance between two words x and y as d(x, y). We may wonder why Hamming distance is important for error detection. The reason is that the Hamming distance between the received codeword and the sent codeword is the number of corrupted bits during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error, and the Hamming distance between the two is d(00000, 01101) = 3. The Hamming distance can easily be found if we apply the XOR operation ($\oplus$) on the two words and count the number of 1s in the result.

Example: The Hamming distance d(000, 011) is 2 because (000 $\oplus$ 011) is 011 (two 1s).

Minimum Hamming Distance for Error Detection

In a set of codewords, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of codewords. If *s* errors occur during transmission, the Hamming distance between the sent and received codeword is *s*. If our system is to detect up to *s* errors, the minimum distance between the valid codes must be (s + 1). To **guarantee** the detection of up to *s* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

1. Parity-Check Code

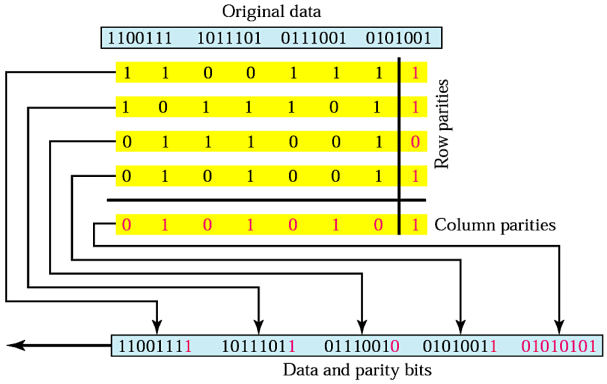
   Perhaps the most familiar error-detecting code is the parity-check code. This code is a linear block code. This code changes a k-bit data word to an n-bit codeword where n = k + 1. The least significant extra bit, the parity bit, is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of 1s, we discuss the even case. The minimum Hamming distance for this category is $d_{min} = 2$, meaning the code is a single-bit error-detecting code. Our code below is a parity-check code with k = 4 and n = 5.

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	**00000**	1000	**10001**
0001	**00011**	1001	**10010**
0010	**00101**	1010	**10100**
0011	**00110**	1011	**10111**
0100	**01001**	1100	**11000**
0101	**01010**	1101	**11011**
0110	**01100**	1110	**11101**
0111	**01111**	1111	**11110**

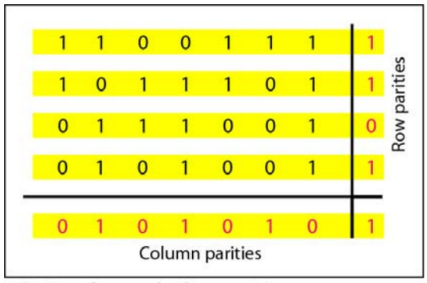
   Interestingly, A parity-check code can detect an odd number of errors. (Try yourself.)

2. Two-Dimensional Parity Check

   A better approach is the two-dimensional parity checks. This method organizes a block of bits in a table (rows and columns). First, we calculate the *parity bit* for each data unit. Then, we organize them into a table. For example, in the Figure below, we have four data units shown in four rows and eight columns (the last column is the parity computed for each row). We calculate the parity bit for each column and create a new row of 8 bits, the parity bits for the whole block. Note that the first parity bit in the fifth row is calculated based on all first bits, the second parity bit is calculated based on all second bits, and so on. We then attach the 8 parity bits to the original data and send them to the receiver.

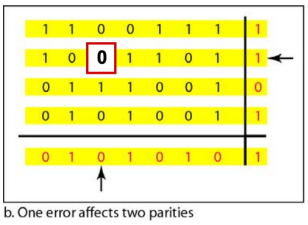


Two-dimensional parity check increases the likelihood of detecting burst errors. As shown in the previous example, a redundancy of n bits can easily detect a burst error of n bits. A burst error of more than n bits is also detected by this method with a very high probability.

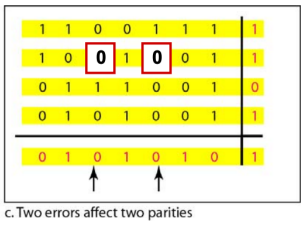


a. Design of row and column parities

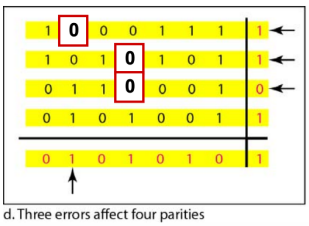
There is, however, one pattern of errors that remains elusive. If 2 bits in one data unit are damaged and two bits in exactly the same positions in another data unit are also damaged, the checker will not detect an error. Consider, for example, below



b. One error affects two parities



c. Two errors affect two parities



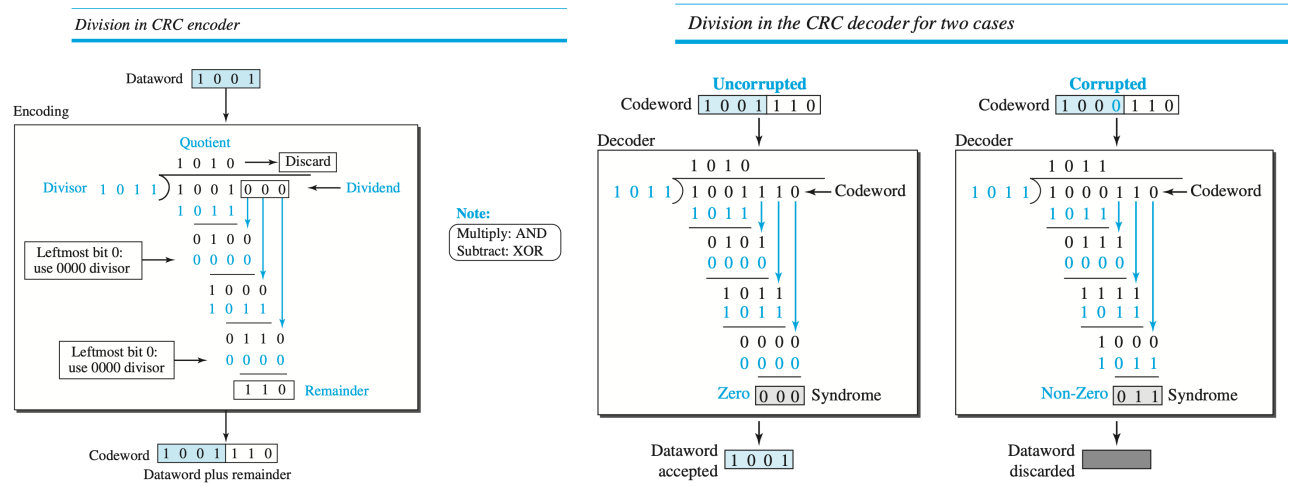
d. Three errors affect four parities

e. Four errors cannot be detected

Can we correct the error in this method? (Try yourself).

3. Cyclic Redundancy Check

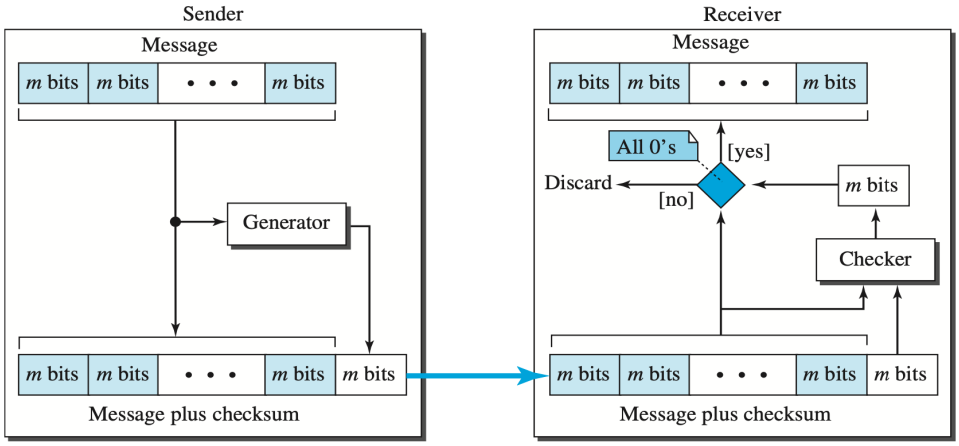
We can create cyclic codes to correct errors. In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding n − k (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size n − k + 1 (4 here), which is predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder is appended to the dataword to create the codeword. The decoder receives the codeword (possibly corrupted in transition). A copy of all n bits is fed to the checker, which is a replica of the generator. The remainder produced by the checker is a syndrome of n − k (3 here) bits fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 left- most bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).



What if the extra redundant bit gets corrupted? (Try yourself)

4. Checksum

Checksum is an error-detecting technique that can be applied to a message of any length. In the Internet, the checksum technique is mostly used at the network and trans-port layer rather than the data-link layer. At the source, the message is first divided into m-bit units. The generator then creates an extra m-bit unit called the checksum, which is sent with the message. At the destination, the checker creates a new checksum from the combination of the message and sent checksum. If the new checksum is all 0s, the message is accepted; otherwise, the message is discarded.



For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, **36**), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere, and the message will not be accepted. In this example, each number can be written as a 4-bit word (each is less than 15) except for the sum. One solution is to use one's complement arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and $2^m - 1$ using only m bits. If the number has more than m bits, the extra leftmost bits must be added to the m rightmost bits (wrapping).

For example, the decimal number 36 in binary is $(100100)_2$. To change it to a 4-bit number we add the extra leftmost bit to the right four bits as shown below.

$$(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}$$

Instead of sending 36 as the sum, we can send 6 as the sum (7, 11, 12, 0, 6, **6**). The receiver can add the first five numbers in one's complement arithmetic. If the result is 6, the numbers are accepted; otherwise, they are rejected.

Interestingly, We can make the receiver's job easier if we send the complement of the sum, the checksum. In one's complement arithmetic, the complement of a number is found by completing all bits (changing all 1s to 0s and all 0s to 1s). This is the same as subtracting the number from $2^m - 1$.

Using Hamming Distance

Earlier, we discussed the Hamming distance for error detection. We said that to detect s errors, the minimum Hamming distance should be $d_{min} = s + 1$. For error detection, we definitely need more distance. It can be shown that to detect *t* errors, we need to have $d_{min} = 2t + 1$. In other words, if we want to correct 10 bits in a packet, we need to make the minimum hamming distance 21 bits, meaning many redundant bits must be sent with the data.

1. Hamming code

Hamming Codes are a general method for constructing error-correcting codes by using a minimum distance of three.

Every integer m has a $(2^m - 1)$-bit Hamming code containing m parity bits and $2^m - 1 - m$ information bits. The parity bits are intermixed with the information bits as follows:

If we number the bit positions from 1 to $2^m - 1$, the bits in position 2^k, where $0 \leq k \leq m - 1$, are the parity bits, and the bits in the remaining positions are information bits.

The value of each parity bit (for even parity) is chosen so that the total number of 1s in a specific group of bit positions is even.

For m = 3, we have a $(2^m-1 =)$ 7-bit Hamming code as shown below:

7	6	5	4	3	2	1	Bit positions
1	1	1	1	0	0	0	Parity group for parity bit 4
1	1	0	0	1	1	0	Parity group for parity bit 2
1	0	1	0	1	0	1	Parity group for parity bit 1
			↑		↑	↑	
			Parity bits				

Parity bits are in position 2^k, where $0 \leq k \leq m - 1$: $\Rightarrow 2^0, 2^1, 2^2 \Rightarrow 1, 2, 4$.
Information bits in the remaining positions: 3, 5, 6, 7.
For each parity bit in position 2^k, its corresponding group of information bits includes all those bits in the position whose binary representation has a 1 in position 2^k.
e.g.

- For parity bit 4=100, therefore positions 5, 6, & 7 are in the group
- For parity bit 2=010, therefore, positions 3, 6, & 7 are in the group
- For parity bit 1=001, therefore, positions 3, 5, & 7 are in the group

The information bits in position 7, whose binary representation is 111, is used to compute the value of the parity bits in positions 4, 2 and 1.
The information bits in position 6, whose binary representation is 110, are used only for computing the value of the parity bits in positions 4 and 2.
The information bits in position 5, whose binary representation is 110, are used only for computing the value of the parity bits in positions 4 and 1.

Example: To send the information bits 1001, we would send the code word 1001100 because the parity bits are 100.

7	6	5	4	3	2	1	parity bit
1	**0**	**0**	?	**1**	?	?	
1	**0**	**0**	*1*	**1**	?	?	*1*
1	**0**	**0**	*1*	**1**	0	?	*0*
1	**0**	**0**	*1*	**1**	0	*0*	*0*
			↑		↑	↑	

Example: transmit 1010101 but received a single-bit error in position 6, i.e. 1110101. Count the number of 1 bit in each of the parity groups for 1110101:

7	6	5	4	3	2	1	# 1 bits in group
1	1	1	0	1	0	1	3
1	1	1	0	1	0	1	3
1	1	1	0	1	0	1	4
			↑		↑	↑	
			wrong parity		wrong parity		

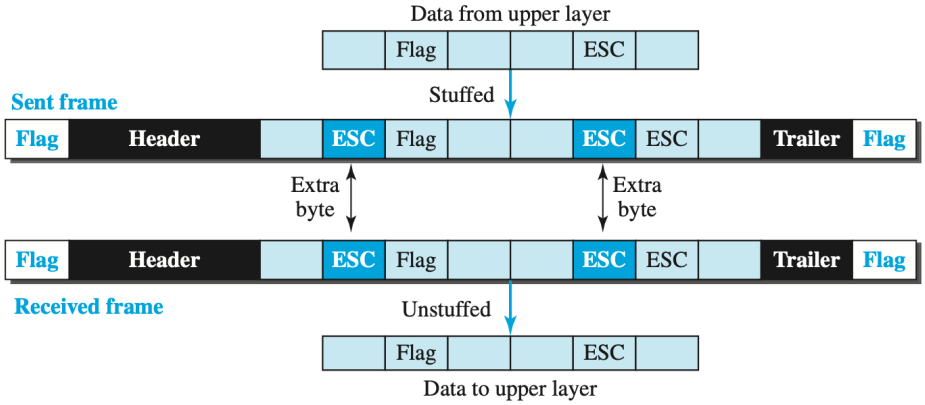
Odd number of 1 bits in group 4 and 2. Therefore, the bit error is in position 4+2 = 6.

DLC SERVICES
The data link control (DLC) deals with procedures for communication between two adjacent nodes—node-to-node communication—no matter whether the link is dedicated or broadcast. Data link control functions include framing, flow & error control.

Framing
Frames can be of fixed or variable size. In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter.

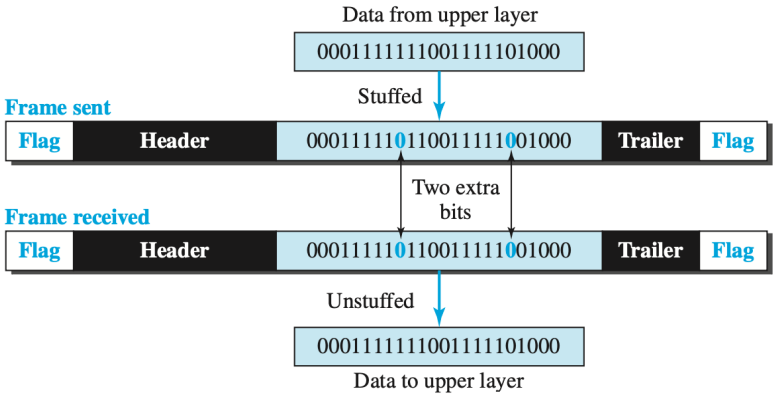
Character-Oriented Framing
In character-oriented (or byte-oriented) framing, data to be carried are 8-bit characters from a coding system such as ASCII. The header, which usually carries the source and destination addresses and other control information, and the trailer, which carries error detection redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (1-byte) flag is added at the frame's beginning and end. Any character used for the flag could also be part of the information. If this happens, the receiver, when encountering this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a **byte-stuffing** strategy was added to character-oriented framing. In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern

as the flag. This byte is usually called the escape character (ESC) and has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting flag. The idea is shown below:



Bit-Oriented Framing

In bit-oriented framing, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphics, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag, 01111110, as the delimiter to define the frame's beginning and end, as shown below.



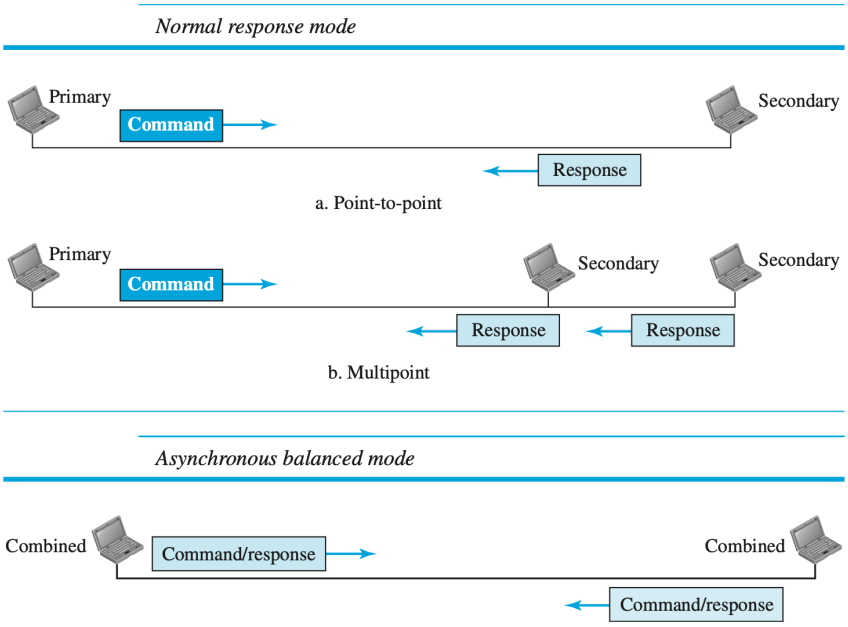
This flag can create the same type of problem we saw in the character-oriented protocols. If the flag pattern appears in the data, we need to inform the receiver somehow that this is not the end of the frame. We do this by stuffing one single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called bit stuffing. If a 0 and five consecutive 1 bits are encountered in bit stuffing, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The receiver will remove the 0.

HDLC

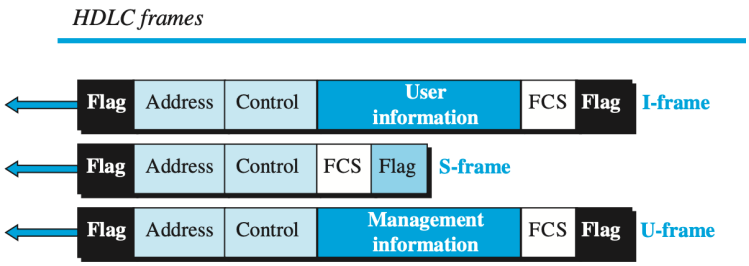
High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. Although this protocol is more a theoretical issue than practical, most of the concepts defined in this protocol are the basis for other practical protocols such as PPP, or the Ethernet protocol.

HDLC provides two common transfer modes that can be used in different configurations: normal response mode (NRM) and asynchronous balanced mode (ABM). In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A primary station can send commands; a secondary station can only respond. The NRM is used for both point-to-point and multipoint links, as shown in Figure below.

In ABM, the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers)



To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: information frames (I-frames), supervisory frames (S-frames), and unnumbered frames (U-frames). Each type of frame serves as an envelope for transmitting a different message type. I- frames are used to data-link user data and control information relating to user data (piggy-backing). S-frames are used only to transport control information. U-frames are reserved for system management.



Flag field. This field contains synchronization pattern 01111110, which identifies both the beginning and the end of a frame.

1. Address field. This field contains the address of the secondary station. If a primary station created the frame, it contains a to address. If a secondary station creates the frame, it contains a from address. The address field can be one byte or several bytes long, depending on the needs of the network.
2. Control field. The control field is one or two bytes used for flow and error control.
3. Information field. The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
4. FCS field. The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte CRC.

The control field determines the type of frame and defines its functionality.

MEDIA ACCESS CONTROL (MAC)

When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link.

RANDOM ACCESS

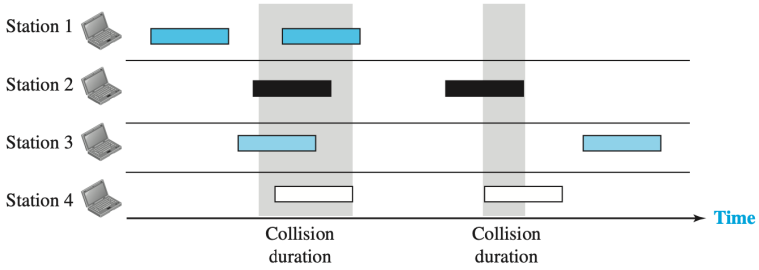
In random-access or *contention* methods, no station is superior to another station, and none is assigned control over another. At each instance, a station with data to send uses a procedure defined by the protocol to decide whether to send. This decision depends on the state of the medium (idle or busy). We will discuss, ALOHA and CSMA in this category.

ALOHA

ALOHA, the earliest *random access* method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN but can be used on any shared medium.

1. Pure ALOHA

The original ALOHA protocol is called pure ALOHA. This is a simple but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send (multiple access). However, since there is only one channel to share, there is the possibility of collision between frames from different stations. A collision involves two or more stations.

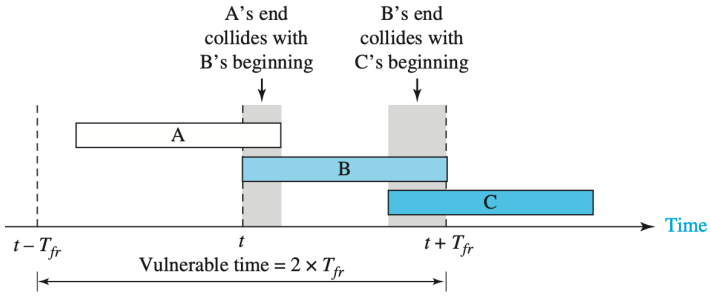


If all these stations try to resend their frames after a time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the back-off time T_B.

The backoff time T_B is a random value that usually depends on K (the number of attempted unsuccessful transmissions). The formula for T_B depends on the implementation. One common formula is the binary exponential backoff. In this method, for each retransmission, a multiplier R = 0 to $2^K - 1$ is randomly chosen and multiplied by T_p (maximum propagation time) or T_{fr} (the average time required to send out a frame) to find TB. Note that in this procedure, the range of the random numbers increases after each collision.

Vulnerable time

Let us find the vulnerable time, the length of time in which collision is possible. We assume that the stations send fixed-length frames, each taking T_{fr} seconds to send. The figure below shows the vulnerable time for station B.



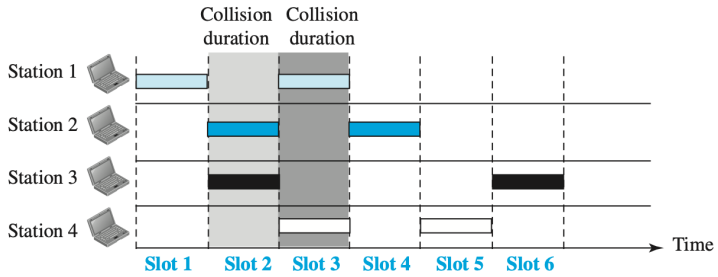
Station B starts to send a frame at a time, t. Now imagine station A has started to send its frame after $t - T_{fr}$. This leads to a collision between Station B's and Station A's frames. On the other hand, suppose that station C starts to send a frame before time $t + T_{fr}$. Here, there is also a collision between frames from station B and station C.

Therefore, we see that the **vulnerable time** during which a collision may occur in pure ALOHA is 2 times the frame transmission time.

Throughput

Let us call G the average number of frames the system generates during one frame transmission time. Then, it can be proven that the average number of successfully transmitted frames for pure ALOHA is $S = G \times e^{-2G}$.

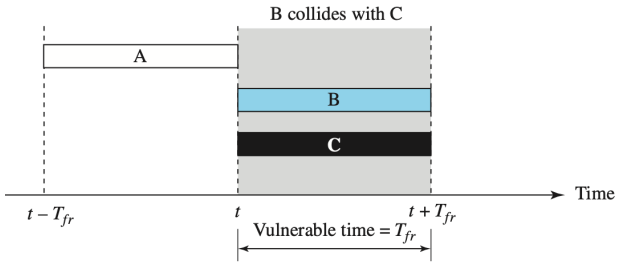
2. Slotted ALOHA



In slotted ALOHA, we divide the time into slots of T_{fr} seconds and force the station to send only at the beginning of the time slot. Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station that started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot.

Vulnerable time

The vulnerable time is now reduced to one-half, equal to T_{fr}.

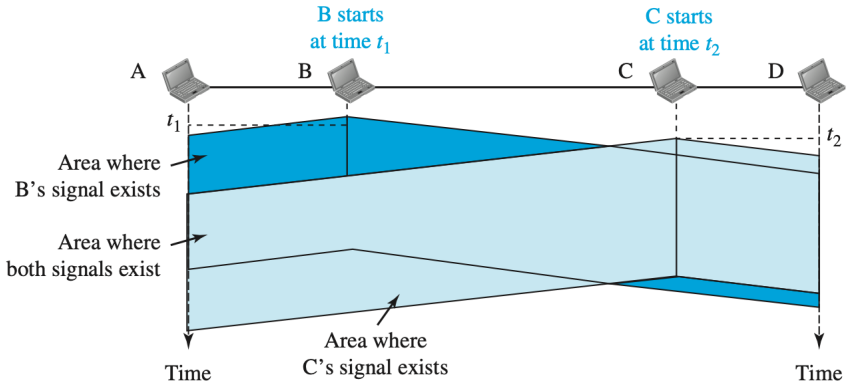


Throughput

It can be proven that the average number of successful transmissions for slotted ALOHA is $S = G \times e^{-G}$.

CSMA

Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk." The possibility of collision still exists because of propagation delay. Consider the below example.



At time t_1, station B senses the medium and finds it idle, so it sends a frame. At time t_2 ($t_2 > t_1$), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the propagation time T_p (maximum propagation time).

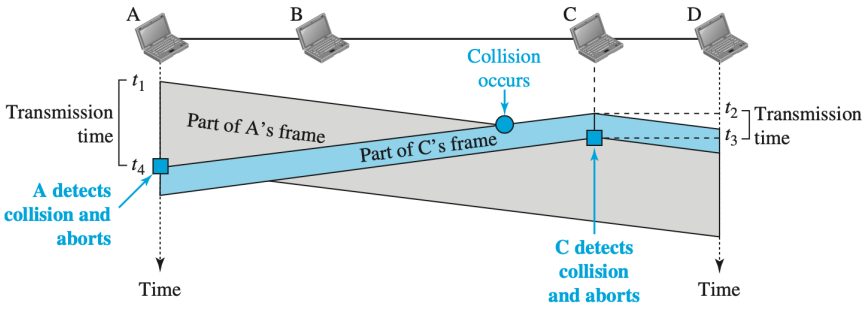
After the station finds the line idle it follows these steps:

1. With probability p, the station sends its frame.
2. With probability $q = 1 - p$, the station waits for the next time slot and checks the line again.
    a. If the line is idle, it goes to step 1.
    b. If the line is busy, it acts as though a collision has occurred and uses the back-off procedure.

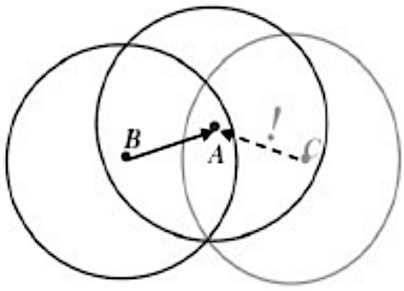
CSMA/CD

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

In the scenario below, At time t_1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t_2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time t_2. Station C detects a collision at time t_3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time t_4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration $t_4 - t_1$; C transmits for the duration $t_3 - t_2$.

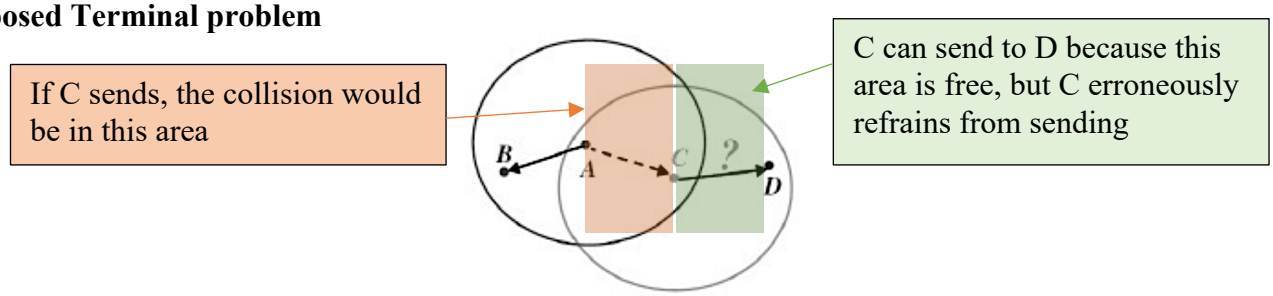


Hidden Terminal problem



The notorious hidden node problem deals with a configuration of three nodes, like A, B, and C, in Figure above, whereby A is within the transmission range of B and C, while C is outside the range of B. In a situation like this, C cannot detect the ongoing transmission of B to A by carrier sensing. Consequently, it can inadvertently interfere with A's reception of B's packet. The transmission range of node B is defined as the area inside which other nodes can receive B's packets correctly. On the other hand, the carrier sense range of B is the area encompassing those nodes whose transmission B can perceive (carrier sense) while not necessarily being able to receive the transmitted packets. Generally, assuming that the two areas are always the same is unreasonable, e.g., the carrier sense range can be twice the transmission range. Suppose that every node in the figure has the same transmission range (represented by a solid circle). Node C is out of the transmission range of node B and thus would appear as a hidden node to B. However, if the carrier sense range of C is larger than the transmission range of B, C is no longer hidden because it can sense the transmission of B and thus avoid interfering with it.

Exposed Terminal problem

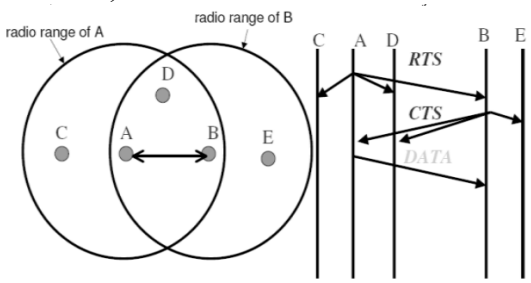


The exposed Node Problem occurs when a node is prevented from sending packets to other nodes due to a neighboring transmitter. Consider an example of 4 nodes labeled A, B, C, and D, where the two receivers (B, D) are out of range of each other, yet the two transmitters in the middle are in range of each other, as shown in the Figure. Here, if a transmission between node A and node B occurs, node C is prevented from transmitting to node D as C concludes after the *carrier sense* that it will interfere with the transmission by its neighbor node A. However, note that node D could still receive the transmission from node C without interference because it is out of range from node A.

RTS/CTS (Request-to-send, Clear-to-send)

The IEEE 802.11 RTS/CTS mechanism helps to solve either of these problems only if the nodes are synchronized. When a node hears an RTS from a neighboring node but not the corresponding CTS, it can be deduced that it is an exposed node and can transmit to other neighboring nodes. If the nodes are not synchronized, the problem may occur that the sender will not hear the CTS during the second sender's data transmission. RTS/CTS works as below:

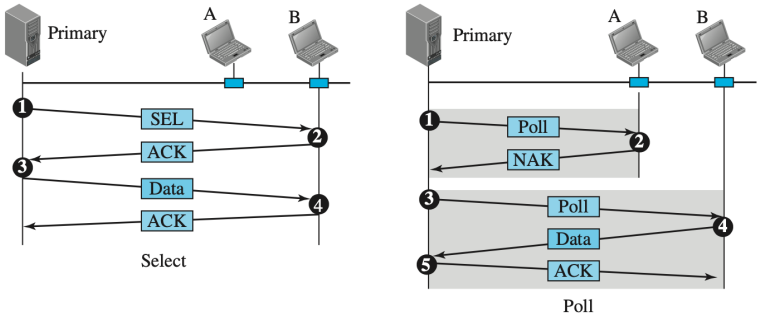
- The transmitter sends an RTS
- If it hears a CTS, it sends data
- If not, retry RTS sometime later
- If you hear a CTS for someone else, don't transmit



CONTROLLED ACCESS

Polling

Polling works with topologies in which one device is designated as a primary station, and the others are secondary. All data exchanges must be made through the primary device, even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device can use the channel at a given time. The primary device, therefore, is always the initiator of a session. This method uses poll and select functions to prevent collisions. However, the drawback is that the system goes down if the primary station fails.



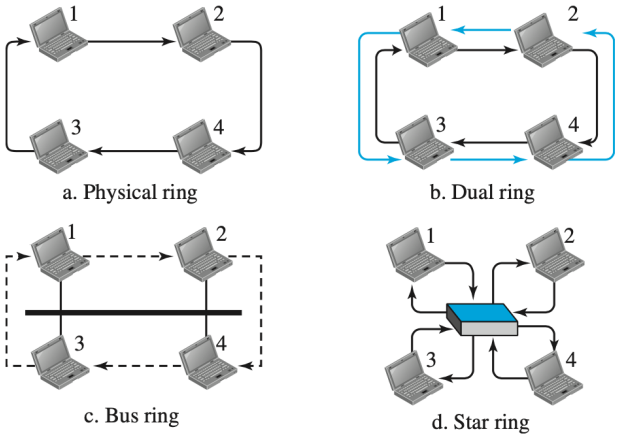
The *select* function is used whenever the primary device has something to send. The primary device uses the *poll* function to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does.

Token Passing

In the token-passing method, the stations in a network are organized in a logical ring. In other words, each station has a predecessor and a successor. The predecessor is the station logically before the station in the ring; the successor is the station after the station in the ring.

In this method, a *special token packet* circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring.

Logical ring and physical topology in token-passing access method

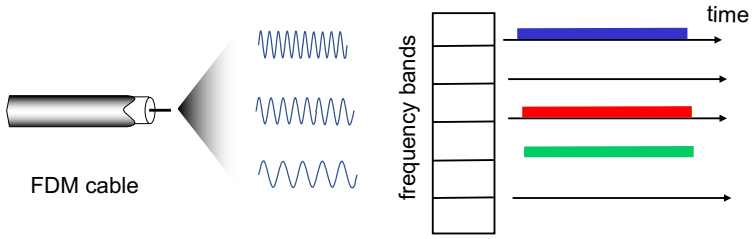


CHANNEL PARTITIONING

channel partition is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code among different stations. This section contains three channelization protocols: FDMA, TDMA, and CDMA (already discussed in the earlier unit).

FDMA

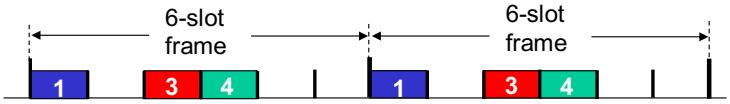
In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it always belongs to the station.



We need to emphasize that although FDMA and frequency-division multiplexing (FDM) conceptually seem similar, they have differences. FDM is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel. A physical multiplexer combines the signals. On the other hand, FDMA is an access method in the data-link layer. The data-link layer in each station tells its physical layer to make a signal from the data passed to it. The signal must be created in the allocated frequency band. The signals created at each station are mixed when they are sent to the common channel. There is no physical multiplexer here.

TDMA

In time-division multiple access (TDMA), the stations share the channel's bandwidth in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot. Each station needs to know the beginning of its slot and the position of its slot in time.



We also need to emphasize that although TDMA and time-division multiplexing (TDM) conceptually seem the same, they have differences. TDM is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.
TDMA, on the other hand, is an access method in the data-link layer. The data-link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer here.

IEEE Project 802

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

Ethernet Evolution

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

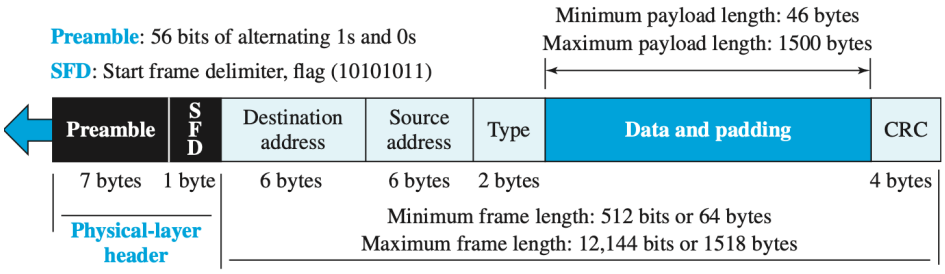
STANDARD ETHERNET

We refer to the original Ethernet technology with a data rate of 10 Mbps as the Standard Ethernet. Although most implementations have moved to other technologies in the Ethernet evolution, some features of the Standard Ethernet have remained the same during the evolution. Ethernet provides a connectionless service, which means each frame sent is independent of the previous or next frame. The sender sends a frame whenever

it has it; the receiver may or may not be ready for it. The sender may overwhelm the receiver with frames, resulting in dropping frames. If a frame drops, the sender will not know about it.

Frame Format

The Ethernet frame contains seven fields, as shown:



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

Start frame delimiter (SFD). This field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization.

Destination address (DA). This field is six bytes (48 bits) and contains the link-layer address of the destination station or stations to receive the packet.

Source address (SA). This field is also six bytes and contains the link-layer address of the packet's sender.

Type. This field defines the upper-layer protocol whose packet is encapsulated in the frame. This protocol can be IP, ARP, OSPF, and so on.

Data. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes.

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

LINK-LAYER SWITCHES

The role of the switch is to receive incoming link-layer frames and forward them to outgoing links;

A switch has the wonderful property that its table is built automatically, dynamically, and autonomously—without any intervention from a network administrator or a configuration protocol. In other words, switches are self-learning. This capability is accomplished as follows:

1. The switch table is initially empty.
2. For each incoming frame received on an interface, the switch stores in its table
   (1) the MAC address in the frame's source address field,
   (2) the interface from which the frame arrived, and
   (3) the current time or something called Time to Live (TTL).

   In this manner, the switch records in its table the LAN segment on which the sender resides. If every host in the LAN eventually sends a frame, then every host will eventually get recorded in the table.
3. The switch deletes an address in the table if no frames are received with that address as the source address after some period of time (the aging time or TTL). In this manner, if a PC is replaced by another PC (with a different adapter), the MAC address of the original PC will eventually be purged from the switch table.

For example, a switch, when it makes an entry in its table, learns about an address as shown:

Address	Interface	Time
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....	....	....

Figure 6.23 ♦ Switch learns about the location of an adapter with address 01-12-23-34-45-56

Switches Versus Routers

routers are store-and-forward packet switches that forward packets using network-layer addresses. Although a switch is also a store-and-forward packet switch, it fundamentally differs from a router in the way of forwarding packets using MAC addresses. Whereas a router decides routes for layer-3 packets, a switch is a layer-2 packet switch.

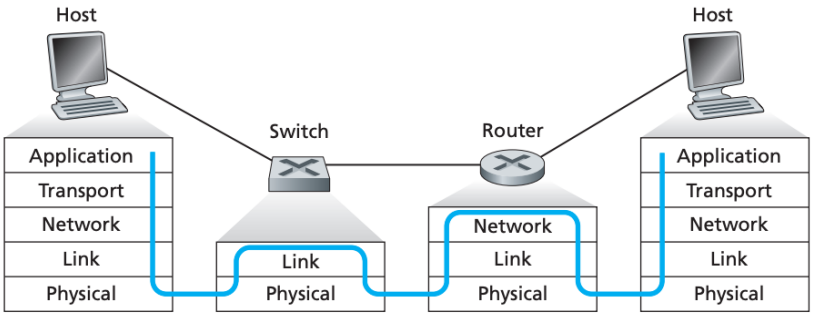


Figure 6.24 ♦ Packet processing in switches, routers, and hosts

References:

1. Andrew S. Tanenbaum, Computer Networks 5th Edition, Pearson, 2013, 978- 9332518742
2. Behrouz Forouzan, Data Communications & Networking Tata McGraw Hill, 2006, ISBN 978-0070584082
3. Ross, Keith W., and James F. Kurose. Computer Networking : A Top-Down Approach. Eighth edition. Pearson; 2021.