Module 2

Error Detection and Correction

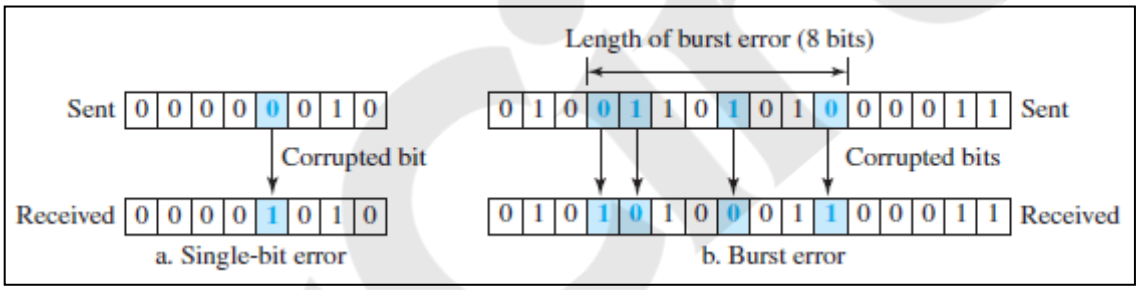
The data link layer transforms the physical layer data into a form suitable for node-to-node (hop-to-hop) communication. Specific responsibilities of the data link layer include ***framing, addressing, flow control, error control,*** **and** ***media access control.***

INTRODUCTION

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.

1. Single-Bit Error

The term *single-bit error* means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.



2. Burst Error

The term *burst error* means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Redundancy

- ➔ The central concept in detecting or correcting errors is redundancy.
- ➔ Some extra bits (redundant bits) are added with data to detect and correct errors.
- ➔ These redundant bits are added by the sender and removed by the receiver.

Detection versus Correction

- ➔ The correction of errors is more difficult than the detection.

➔ In error detection, we are looking only to see if any error has occurred.

➔ In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message.

Coding

Redundancy is achieved through various coding schemes. Coding schemes can be divided into two broad categories: block coding and convolution coding.

BLOCK CODING

➔ In block coding, message is divided into blocks, each of *k* bits, called **datawords**.

➔ Add *r* redundant bits to each block to make the length $n = k + r$. The resulting *n-bit* blocks are called **codewords**.

➔ With *k* bits, 2^k datawords can be created; with *n* bits, 2^n codewords can be created.

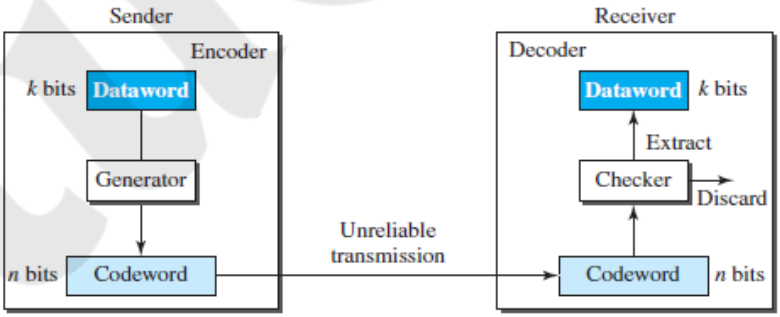
➔ Since $n > k$, the number of possible codewords is larger than the number of possible datawords.

Error Detection

If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.

2. The original codeword has changed to an invalid one.



Hamming Distance

➔ The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits.

➔ The Hamming distance between two words *x* and *y is shown* as *d(x, y).*

➔ The Hamming distance can easily be found by applying the XOR operation on the two words and count the number of 1s in the result.

Ex The Hamming distance d(10101, 11110) is 3 because

Minimum Hamming Distance

➔ It is the measurement that is used for designing a code is the minimum Hamming distance of two.

➔ It is the smallest Hamming distance (d_{min}) between all possible pairs.

$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

d_{min} in this case is 3

Hamming Distance and Error

When a codeword is corrupted during transmission, the Hamming distance between the sent and received codewords is the number of bits affected by the error.

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(OOOOO,* 01101) =3.

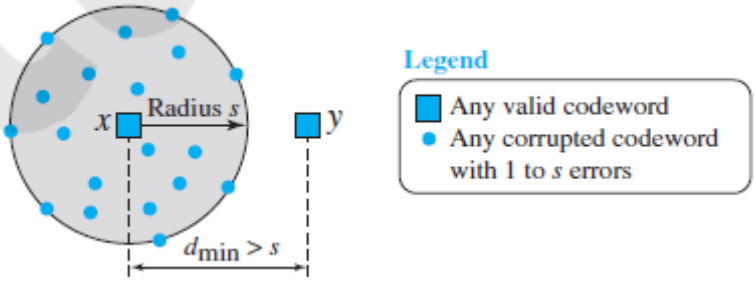
Minimum Distance for Error Detection

If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If the code is to detect up to s errors, the minimum distance between the valid codes must be s + 1, so that the received codeword does not match a valid codeword.

We can look at this geometrically.

➔ Let us assume that the sent codeword *x* is at the center of a circle with radius *s*. All other received codewords that are created by 1 to *s* are points inside the circle or on the perimeter of the circle

➔ All other valid codewords must be outside the circle, as shown in Figure



LINEAR BLOCK CODES

Minimum Distance for Linear Block Codes

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

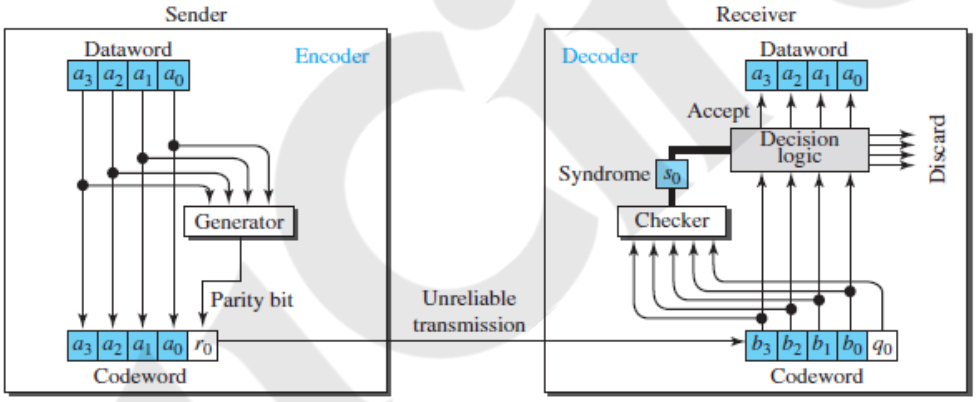
Some Linear Block Codes

1. ***Simple Parity-Check Code(Explain the structure of encoder and decoder S P Code)***

➔ In this code, a *k-bit* dataword is changed to an n-bit codeword where $n = k + 1$.

➔ The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even.

➔ The minimum Hamming distance for this category is $d_{min}=2$, which means that the code is a single-bit error-detecting code; it cannot correct any error.



1. The encoder uses a generator that takes a copy of a 4-bit dataword*(ao,* aI' *a2'* and *a3)* and generates a parity bit *ro.* The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of 1's in the codeword even.

2. The sender sends the codeword which may be corrupted during transmission. The receiver receives a 5-bit word.

3. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the syndrome, is just 1 bit. The

4. The syndrome is passed to the decision logic analyzer. If the syndrome is 0, there is no error in the received codeword; the data portion of the received codeword is accepted as the dataword;

5. If the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

CYCLIC CODES

➔ Cyclic codes are special linear block codes with one extra property.

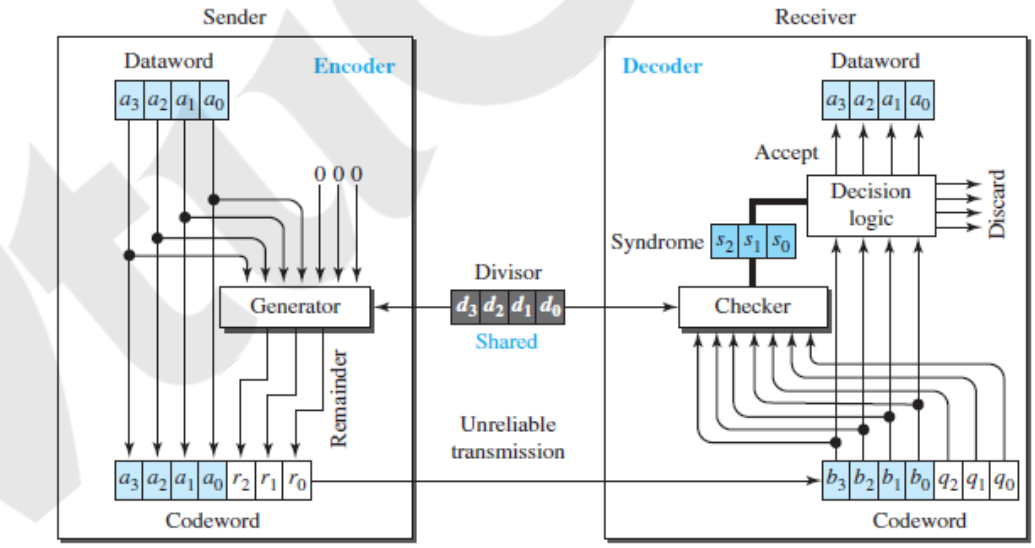
➔ In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

➔ In this case, if we call the bits in the first word a_o to a_6 and the bits in the second word b_o to b_6, we can shift the bits by using the following:

$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

Cyclic Redundancy Check

It is a subset of cyclic codes called the cyclic redundancy check (CRC), which is used in networks such as LANs and WANs.



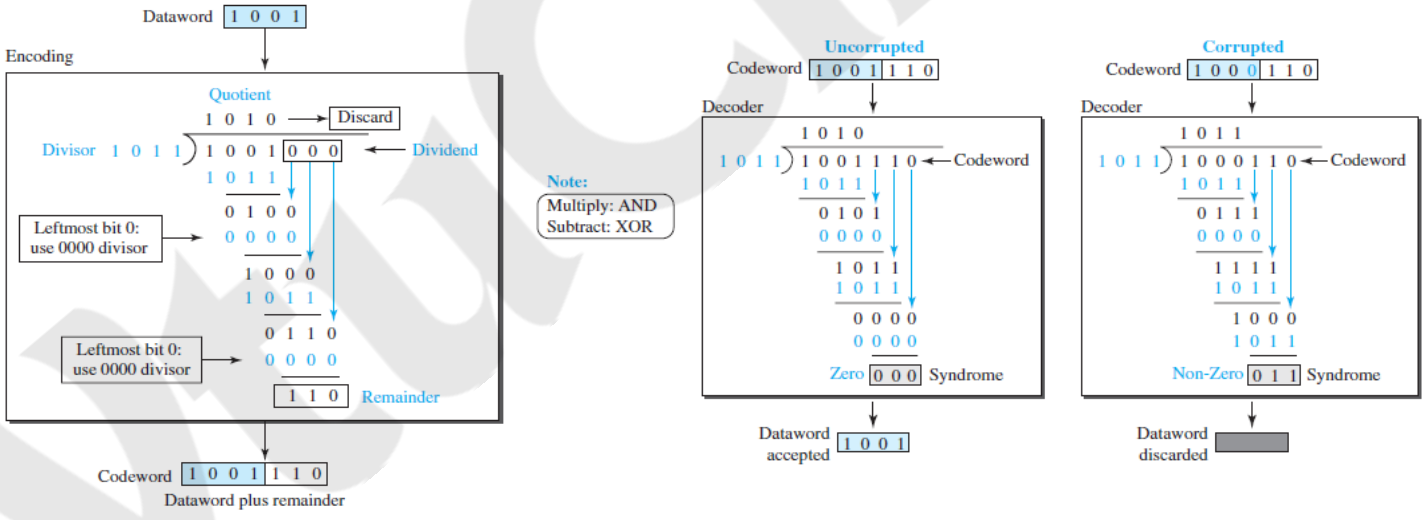
.

In the **encoder**,

1. The dataword has *k* bits (4 here); the codeword has *n* bits (7 here).
2. The size of the dataword is augmented by adding *n* - *k* (3 here) 0s to the right-hand sideof the word. The n-bit result is fed into the generator.
3. The generator uses a divisor of size *n* - *k* + I (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division).
4. The quotient of the division is discarded; the remainder *($r_2r_1r_0$)* is appended to the dataword to create the codeword.

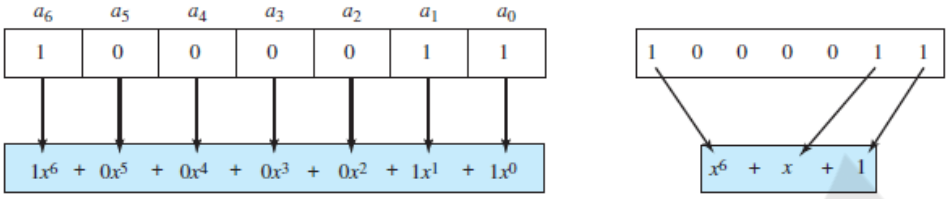
The decoder,

1. Receives the possibly corrupted codeword.
2. A copy of all *n* bits is fed to the checker which is a replica of the generator.
3. The remainder produced by the checker is a syndrome of *n* - *k* (3 here) bits, which is fed to the decision logic analyzer.
4. The analyzer has a simple function. If the syndrome bits are all as, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).



Polynomials

A pattern of Os and 1s can be represented as a **polynomial** with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit.



a. Binary pattern and polynomial

b. Short form

Figure shows one immediate benefit; a 7-bit pattern can be replaced by three terms. The benefit is even more conspicuous when we have a polynomial such as $x^{23}+x^3+1$. Here the bit pattern is 24 bits in length (three 1s and twenty-one 0s) while the polynomial is just three terms.

Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial x^6+x+1 is 6. Note that the degree of a polynomial is 1 less that the number of bits in the pattern. The bit pattern in this case has 7 bits.

Adding and Subtracting Polynomials

Adding and subtracting polynomials in mathematics are done by adding or subtracting the coefficients of terms with the same power. For example, adding $x^5 + x^4 + x^2$ and $x^6 + x^4 + x^2$ gives just $x^6 + x^5$. The terms x^4 and x^2 are deleted.

Multiplying or Dividing Terms

In this arithmetic, multiplying a term by another term is very simple; we just add the powers. For example, x^3 x x^4 is x^7 ,

For dividing, we just subtract the power of the second term from the power of the first. For example, $x^5 1 x^2$ is x^3.

Multiplying Two Polynomials

Multiplying a polynomial by another is done term by term. Each term of the first polynomial must be multiplied by all terms of the second. The result, of course, is then simplified, and pairs of equal terms are deleted. The following is an example:

$$(x^5 + x^3 + x^2 + x)(x^2 + x + 1)$$
$$= x^7 + x^6 + x^5 + x^5 + x^4 + x^3 + x^4 + x^3 + x^2 + x^3 + x^2 + x$$
$$= x^7 + x^6 + x^3 + x$$

Dividing One Polynomial by Another

- ➔ . We divide the first term of the dividend by the first term of the divisor to get the first term of the quotient.
- ➔ We multiply the term in the quotient by the divisor and subtract the result from the dividend.
- ➔ We repeat the process until the dividend degree is less than the divisor degree.

Shifting

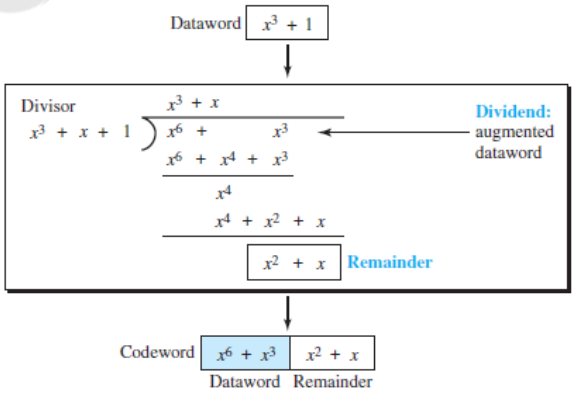
Shifting to the left is accomplished by multiplying each term of the polynomial by x^m, where m is the number of shifted bits; shifting to the right is accomplished by dividing each term of the polynomial by x^m..

Shifting left 3 bits: 10011 becomes 10011000 $x^4 + x + 1$ becomes $x^7 + x^4 + x^3$

Shifting right 3 bits: 10011 becomes 10 $x^4 + x + 1$ becomes x

Cyclic Code Encoder Using Polynomials

- ➔ The dataword 1001 is represented as $x^3 + 1$.
- ➔ The divisor1011 is represented as $x^3 + x + 1$.
- ➔ Find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by x) The result is $x^6 + x^3$.



Cyclic Code Analysis

We can analyze a cyclic code to find its capabilities by using polynomials. We define the following, *wheref(x)* is a polynomial with binary coefficients.

Dataword: *d(x)* Codeword: *c(x)* Generator: *g(x)*
Syndrome: *sex)* Error: *e(x)*

.In a cyclic code,

I. If *s(x)!*=0, one or more bits is corrupted.

2. If *s(x)* =0, either

a. No bit is corrupted. or

b. Some bits are corrupted, but the decoder failed to detect them.

In our analysis we want to find the criteria that must be imposed on the generator, *g(x)* to detect the type of error we especially want to be detected. Let us first find the relationship among the sent codeword, error, received codeword, and the generator.

We can say

$$\text{Received codeword} = c(x) + e(x)$$

In other words, the received codeword is the sum of the sent codeword and the error. The receiver divides the received codeword by *g(x)* to get the syndrome. We can write this as

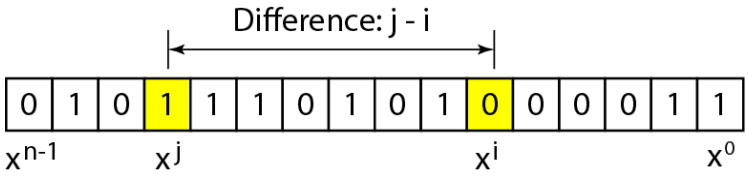
$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

In a cyclic code, those *e(x)* errors that are divisible by *g(x)* are not caught.

Single-Bit Error

If the generator has more than one term and the coefficient of $x^O$is 1, all single errors can be caught.

Two Isolated Single-Bit Errors



If a generator cannot divide $r + 1$ (*t* between 0 and *n* - 1), then all isolated double errors can be detected.

Odd Numbers of Errors

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

Burst Errors

1. All burst errors with $L <= r$ will be detected.
2. All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$
3. All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^r$

A good polynomial generator needs to have the following characteristics:

1. It should have at least two terms.
2. The coefficient of the term x^0 should be 1.
3. It should not divide $X^t + 1$, for *t* between 2 and *n* - 1.
4. It should have the factor $x + 1$.

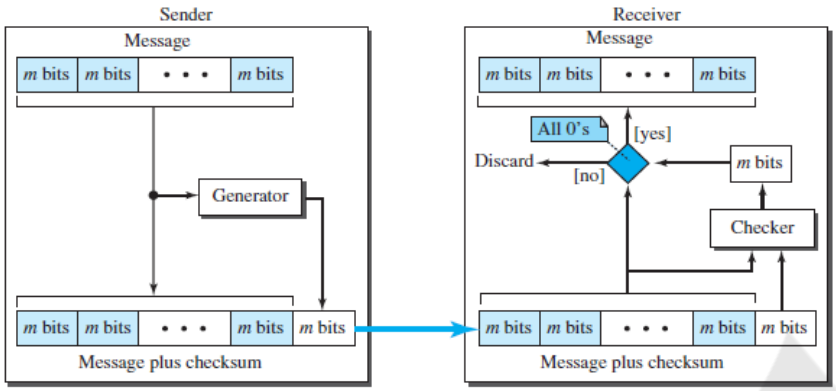
Advantages of Cyclic Codes

1. Cyclic codes have a very good performance in detecting single-biterrors, double errors, an odd number of errors, and burst errors.
2. They can easily be implemented in hardware and software. They are especially fast when implemented in hardware.

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 transport 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.



Concept

The concept of the checksum is not difficult.

- ➔ Suppose our data is a list of five 4-bit numbers that we want to send to a destination.
- ➔ In addition to sending these numbers, we send the sum of the numbers.
- ➔ 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 data are not accepted.
- ➔ We can make the job of the receiver easier if we send the negative (complement) of the sum, called the ***checksum***.
- ➔ In this case, we send (7, 11, 12,0,6, -36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

One's Complement

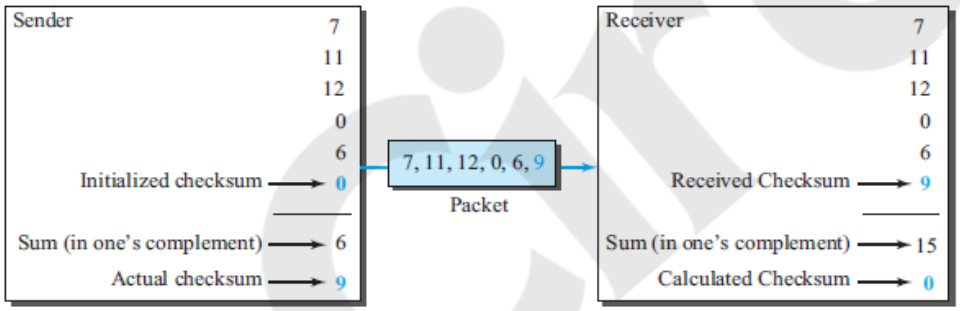
- ➔ The previous example has one major drawback. All of our data can be written as a 4-bit word (they are less than 15) except for the checksum.
- ➔ One solution is to use one's complement arithmetic.
- ➔ If the number has more than *n* bits, the extra leftmost bits need to be added to the *n* rightmost bits (wrapping).
- ➔ In one's complement arithmetic, a negative number can be represented by inverting all bits (changing a 0 to a 1 and a 1 to a 0).

Sender

1. The sender initializes the checksum to 0 and adds all data items and the checksum.
2. The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6.
3. The sum is then complemented, resulting in the checksum value 9 (15 - 6 = 9). The sender now sends six data items to the receiver including the checksum 9.

Receiver

1. The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45.
2. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes O. Since the value of the checksum is 0, this means that the data is not corrupted.
3. The receiver drops the checksum and keeps the other data items.
4. If the checksum is not zero, the entire packet is dropped



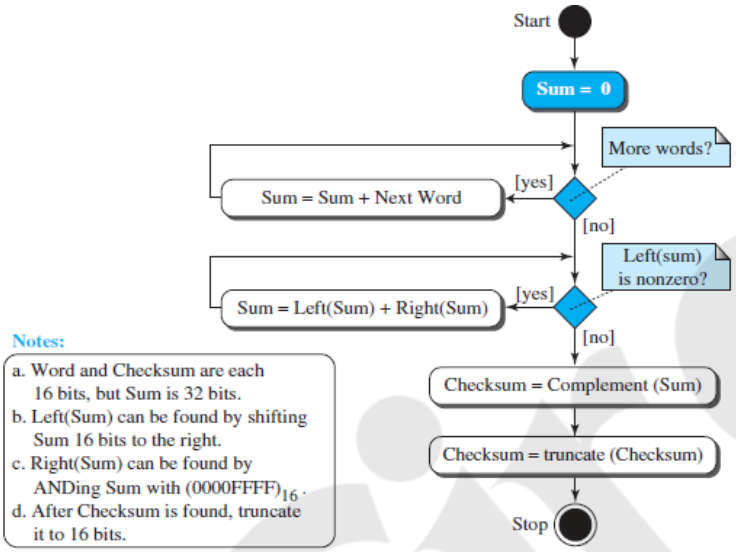
Internet Checksum

Traditionally, the Internet has been using a 16-bit checksum. The sender calculates the checksum by following these steps.

Sender	*Receiver*
1. The message is divided into 16-bit words.	1. The message and the checksum are received.
2. The value of the checksum word is initially set to zero.	2. The message is divided into 16-bit words.
3. All words including the checksum are added using one's complement addition.	3. All words are added using one's complement addition.
4. The sum is complemented and becomes the checksum.	4. The sum is complemented and becomes the new checksum.
5. The checksum is sent with the data.	5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

Algorithm

We can use the flow diagram of Figure below to show the algorithm for calculation of the checksum. A program in any language can easily be written based on the algorithm. Note that the first loop just calculates the sum of the data units in two's complement; the second loop wraps the extra bits created from the two's complement calculation to simulate the calculations in one's complement.



Other Approaches to the Checksum

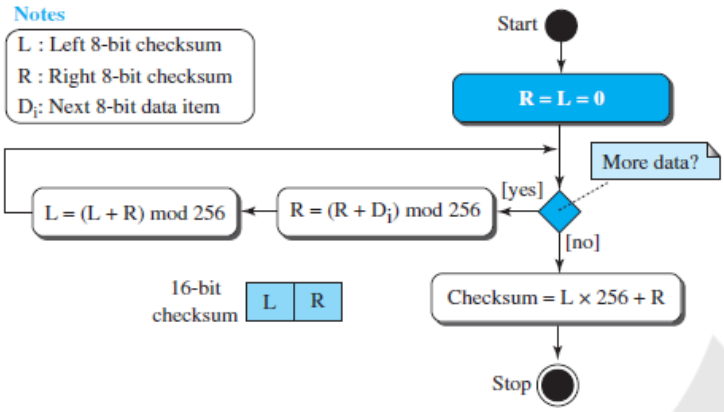
As mentioned before, there is one major problem with the traditional checksum calculation.

If two 16-bit items are transposed in transmission, the checksum cannot catch this error.

This can be overcome by: Fletcher and Adler.

Fletcher Checksum

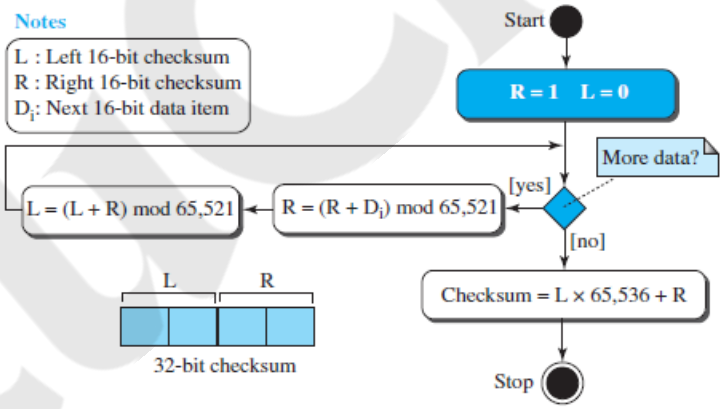
- ➔ The Fletcher checksum was devised to weight each data item according to its position. Fletcher has proposed two algorithms: 8-bit and 16-bit.
- ➔ The 8-bit Fletcher is calculated over data octets (bytes) and creates a 16-bit checksum. The calculation is done modulo 256 (28), which means the intermediate results are divided by 256 and the remainder is kept.
- ➔ The algorithm uses two accumulators, L and R. The first simply adds data items together; the second adds a weight to the calculation.
- ➔ The 16-bit Fletcher checksum is similar to the 8-bit Fletcher checksum, but it is calculated over 16-bit data items and creates a 32-bit checksum. The calculation is done modulo 65,536.



Adler Checksum

➔ The Adler checksum is a 32-bit checksum. Figure shows a simple algorithm in flowchart form. It is similar to the 16-bit Fletcher with three differences.

1. Calculation is done on single bytes instead of 2 bytes at a time.
2. The modulus is a prime number (65,521) instead of 65,536.
3. L is initialized to 1 instead of 0. It has been proved that a prime modulo has a better detecting capability in some combinations of data.



Data Link Control

DLC SERVICES

The two main functions of the data link layer are data link control and media access control. The first, data link control, deals with the design and procedures for communication between two adjacent nodes: node-to-node communication.

Data link control functions include framing, flow and error control, and software implemented

Protocols, that provides smooth and reliable transmission of frames between nodes.

Framing

- ➔ Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination.
- ➔ The data link layer needs to pack bits into frames, so that each frame is distinguishable from another.
- ➔ Framing adds a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.
- ➔ Frames can be of fixed or variable size.

Fixed-Size Framing

In fixed-size framing, the size itself can be used as a delimiter.

Variable-Size Framing(differentiate b/n Character-oriented & Bit-oriented approach)
In variable-size framing, we need a way to define the end of the frame and the beginning of the next. Historically, two approaches were used for this purpose:

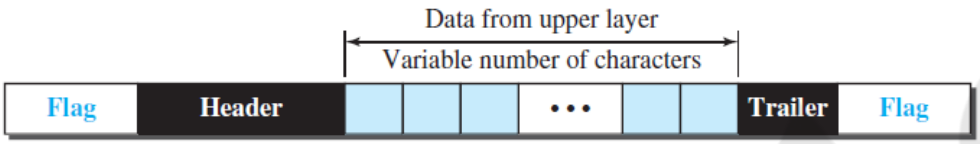
1. Character-oriented approach.
2. Bit-oriented approach

Character-Oriented Protocols

- ➔ In a character-oriented protocol, data to be carried are 8-bit characters I .
- ➔ The header, carries the source and destination addresses and other control information
- ➔ The trailer, carries error detection or error correction redundant bits.

➔ To separate one frame from the next, an 8-bit (I-byte) **flag** is added at the beginning and the end of a frame.

➔ The flag, composed of protocol-dependent special characters, signals the start or end of a frame.



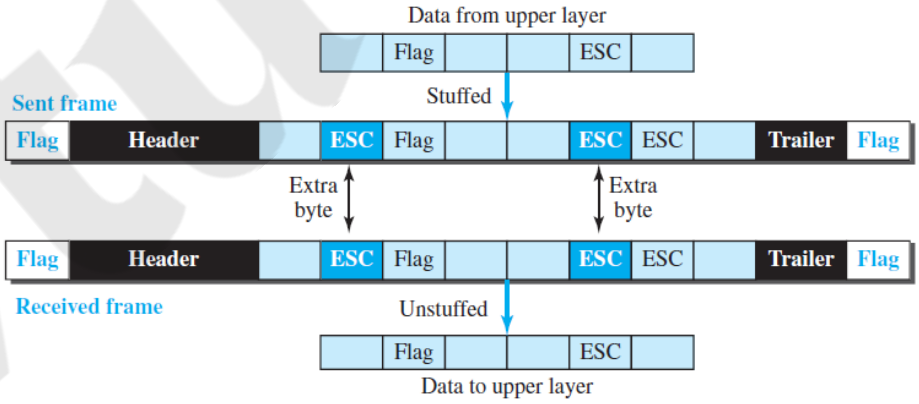
➔ If the receiver encounters the flag pattern in the middle of the data, thinks it has reached the end of the frame.

➔ To fix this problem, a **byte-stuffing** strategy was added to character-oriented framing.

➔ The data section is stuffed with an extra byte if it has data similar to flag.. This byte is usually called the escape character (ESC), which has a predefined bit pattern.

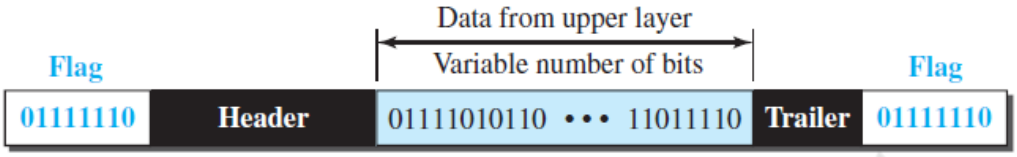
➔ The receiver removes ESC from the data section and treats the next character as data, not a delimiting flag.

➔ If the escape character is part of the text, an extra one is added to show that the second one is part of the text.



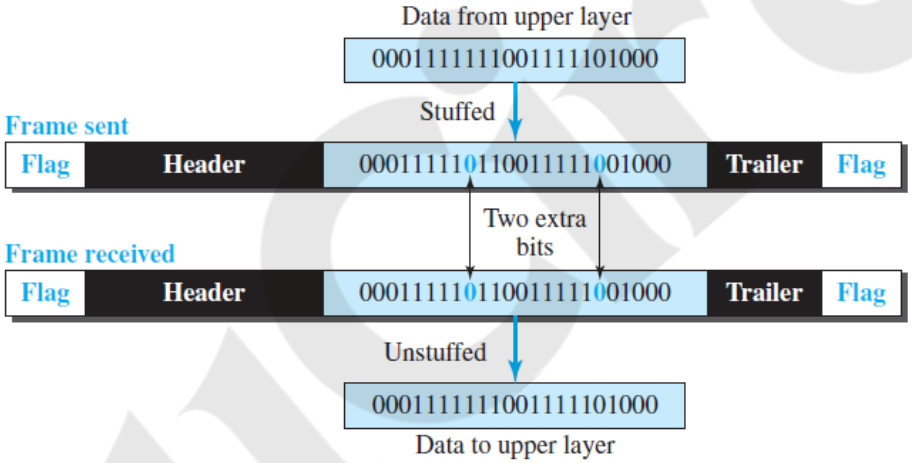
Bit-Oriented Protocols

➔ Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame.



➔ If the flag pattern appears in the data, a single 1bit is stuffed to prevent the pattern from looking like a flag. The strategy is called bit stuffing.

➔ In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added which is removed from the data by the receiver.

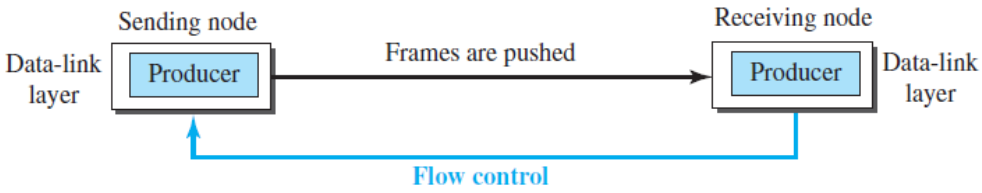


FLOW AND ERROR CONTROL

Flow Control

➔ Flow control coordinates the amount of data that can be sent before receiving an acknowledgment.

➔ In communication at the data-link layer, we are dealing with four entities: network and data-link layers at the sending node and network and data-link layers at the receiving node.



➔ The data-link layer at the sending node tries to push frames toward the data-link layer at the receiving node.

➔ If the receiving node cannot process and deliver the packet to its network at the same rate that the frames arrive, it becomes overwhelmed with frames.

➔ Flow control in this case can be feedback from the receiving node to the sending node to stop or slow down pushing frames.

Buffers

➔ A buffer is a set of memory locations that can hold packets at the sender and receiver. It is usually used to implement flow control.

➔ When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Error Control

Error control at the data-link layer is implemented using one of the following two methods.

➔ If the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer.

➔ if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

Combination of Flow and Error Control

Flow and error control can be combined. In a simple situation, the acknowledgment that is sent for flow control can also be used for error control to tell the sender the packet has arrived uncorrupted.

Connectionless and Connection-Oriented

A DLC protocol can be either connectionless or connection-oriented.

Connectionless Protocol

In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. The frames are not numbered and there is no sense of ordering.

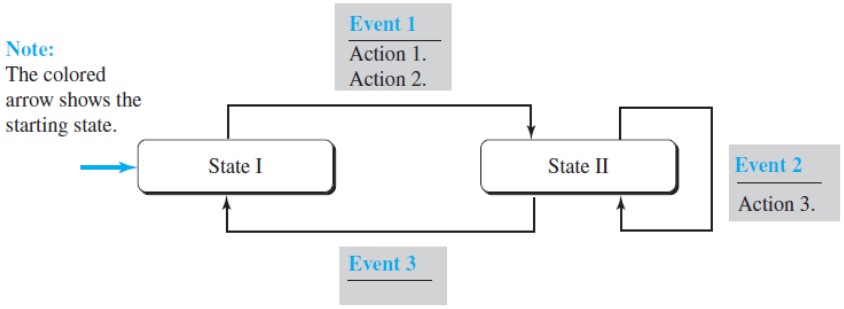
Connection-Oriented Protocol

In a connection-oriented protocol, a logical connection should first be established between the two nodes (setup phase). After all frames that are somehow related to each other are transmitted (transfer phase), the logical connection is terminated (teardown phase). In this type of communication, the frames are numbered and sent in order

DATA-LINK LAYER PROTOCOLS

Traditionally four protocols have been defined for the data-link layer to deal with flow and error control: Simple, Stop-and-Wait, Go-Back-N, and Selective-Repeat. We discuss the first two.

- ➔ The behavior of a data-link-layer protocol can be better shown as a **finite state machine (FSM).**
- ➔ An FSM is thought of as a machine with a finite number of states. The machine is always in one of the states until an *event* occurs.
- ➔ Each event is associated with two reactions: defining the list (possibly empty) of actions to be performed and determining the next state (which can be the same as the current state).
- ➔ One of the states must be defined as the initial state, the state in which the machine starts when it turns on.
- ➔ Rounded-corner rectangles to show states, colored text to show events, and regular black text to show actions.
- ➔ A horizontal line is used to separate the event from the actions, although later we replace the horizontal line with a slash.
- ➔ The arrow shows the movement to the next state.



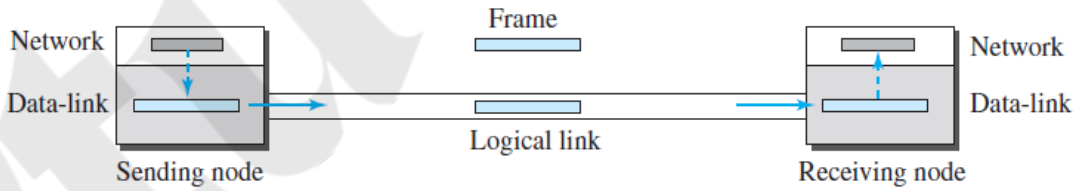
The figure shows a machine with three states. There are only three possible events and three possible actions.

1. The machine starts in state I. If event 1 occurs, the machine performs actions 1 and 2 and moves to state II.
2. When the machine is in state II, two events may occur. If event 1 occurs, the machine performs action 3 and remains in the same state, state II.
3. If event 3 occurs, the machine performs no action, but move to state I.

Simple Protocol

➔ It has no flow or error control. It is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver.

➔ We assume that the receiver can immediately handle any frame it receives.



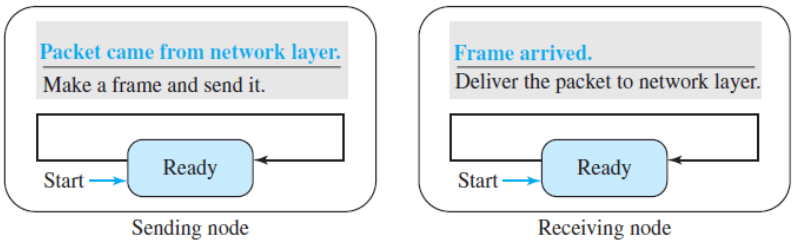
FSMs

➔ Each FSM has only one state, the *ready state*. The sending machine remains in the ready state until a request comes from the process in the network layer.

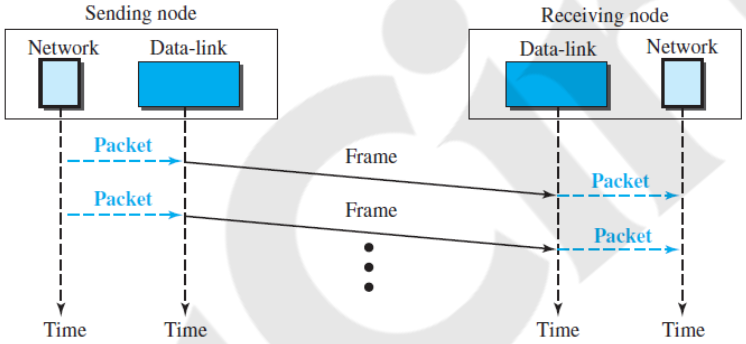
➔ When this event occurs, the sending machine encapsulates the message in a frame and sends it to the receiving machine.

➔ The receiving machine remains in the ready state until a frame arrives from the sending machine.

➔ When this event occurs, the receiving machine decapsulates the message out of the frame and delivers it to the process at the network layer.



An example of communication using this protocol is below. It is very simple. The sender sends frames one after another without even thinking about the receiver.



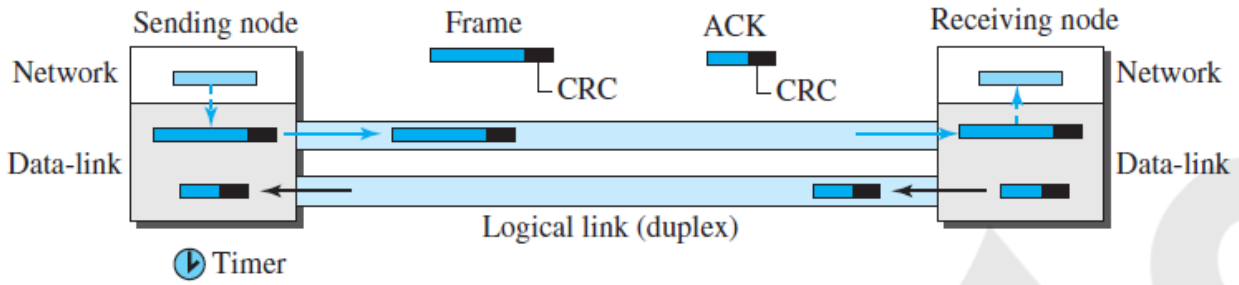
Stop-and-Wait Protocol

➔ Use both flow and error control. The sender sends one frame at a time and waits for an acknowledgment, before sending the next one.

➔ To detect corrupted frames, we need to add a CRC to each data frame. When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.

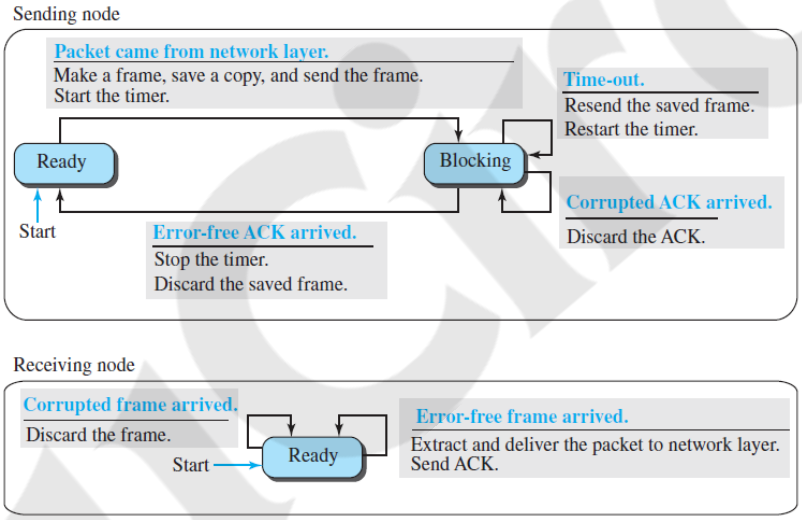
➔ Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame.

➔ If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted. This means that the sender needs to keep a copy of the frame



FSMs

Figure shows the FSMs for our primitive Stop-and-Wait protocol.



Sender States

The sender is initially in the ready state, but it can move between the ready and blocking state ***Ready State.*** When the sender is in this state, it is only waiting for a packet from the network layer. If a packet comes from the network layer, the sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame. The sender then moves to the blocking state. ***Blocking State.*** When the sender is in this state, three events can occur:

a. If a time-out occurs, the sender resends the saved copy of the frame and restarts the timer.
b. If a corrupted ACK arrives, it is discarded.
c. If an error-free ACK arrives, the sender stops the timer and discards the saved copy of the frame. It then moves to the ready state.

Receiver

The receiver is always in the *ready* state. Two events may occur:

a. **If** an error-free frame arrives, the message in the frame is delivered to the network layer and an ACK is sent.

b. If a corrupted f

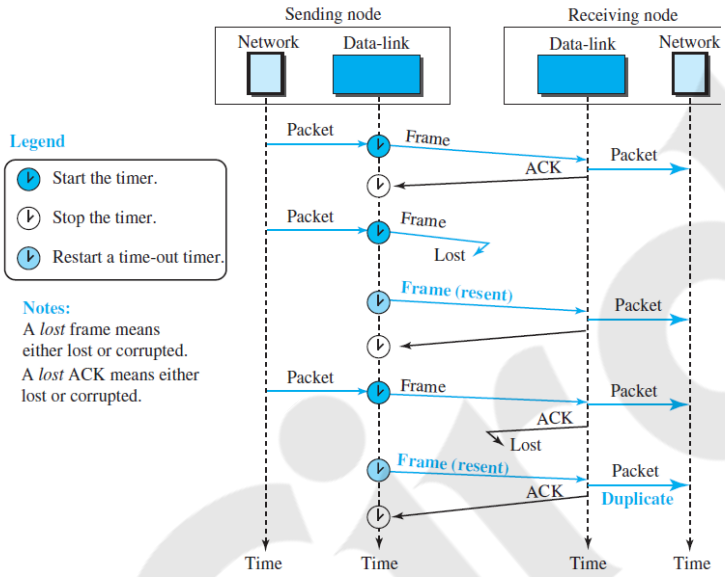
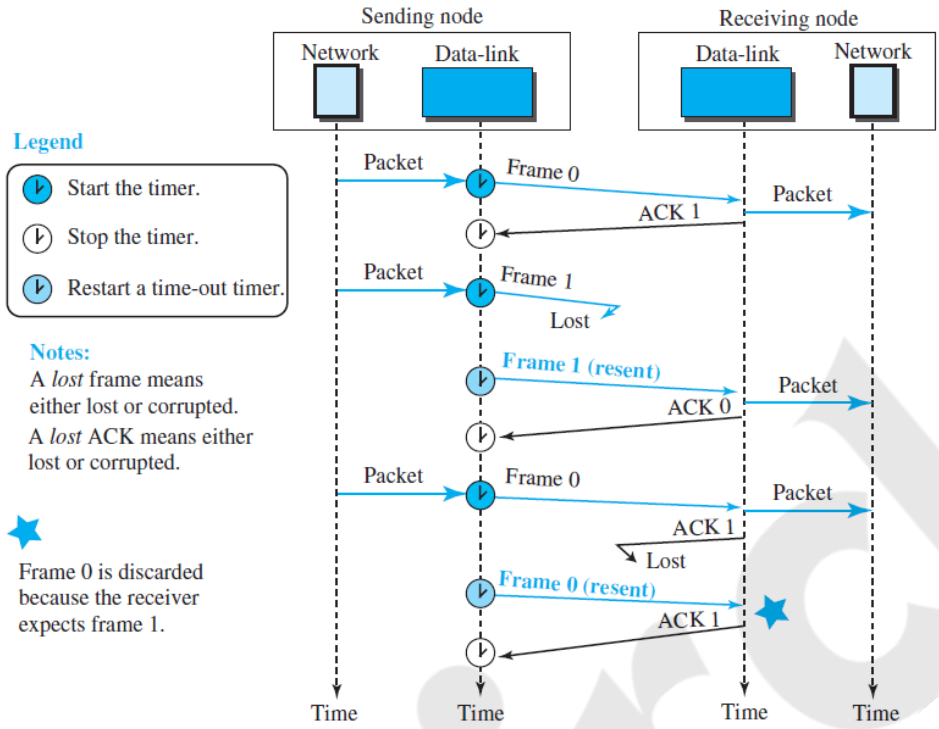


Figure shows an example. The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent. The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent. However, there is a problem with this scheme. The

Network layer at the receiver site receives two copies of the third packet, which is not right.

Sequence and Acknowledgment Numbers

- ➔ Duplicate packets, as much as corrupted packets, need to be avoided. As an example, assume we are ordering some item online.
- ➔ If each packet defines the specification of an item to be ordered, duplicate packets mean ordering an item more than once.
- ➔ o correct the problem in Example, we need to add **sequence numbers** to the data frames and **acknowledgment numbers** to the ACK frames.
- ➔ Numbering in this case is very simple. Sequence numbers are 0, 1, 0, 1, 0, 1, . . . ; the acknowledgment numbers can also be 1, 0, 1, 0, 1, 0, …
- ➔ An acknowledgment number always defines the sequence number of the next frame to receive.



Piggybacking

A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

HDLC

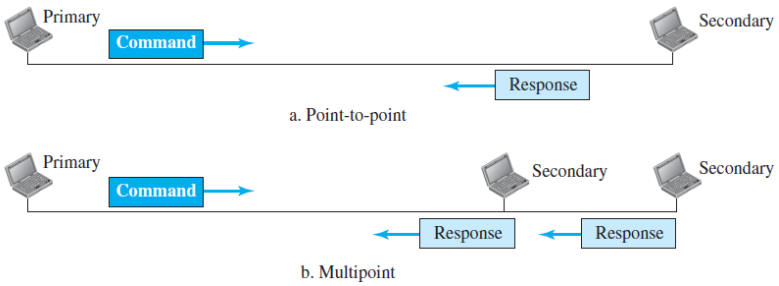
High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the Stop-and-Wait protocol.

Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations:

Normal Response Mode

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 multiple-point links, as shown in Figure.



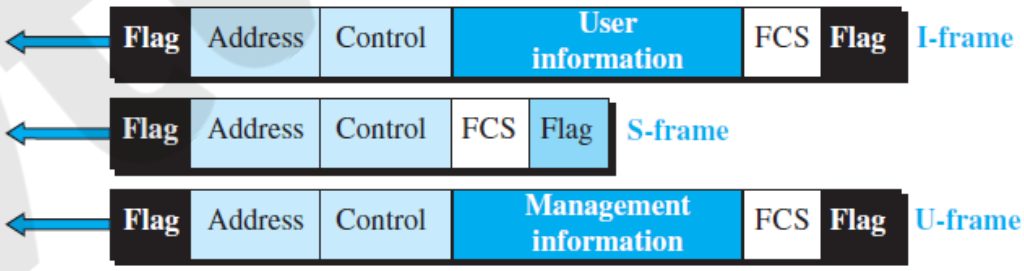
Asynchronous Balanced Mode

In asynchronous balanced mode (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), as shown in Figure. This is the common mode today.

Framing

To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames:

1. I-frames (information frames) are used to transport user data and control information relating to user data (piggybacking).
2. S-frames (supervisory frames) are used only to transport control information. V-frames are reserved for system management.
3. U frames (unnumbered frames) Information carried by U-frames is intended for managing the link itself.



Flag field-The flag field of an HDLC frame is an 8-bit sequence with the bit pattern 01111110 that identifies both the beginning and the end of a frame and serves as a synchronization pattern for the receiver.

Address field- The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a ***to*** address. If a secondary creates the frame, it contains *a* ***from*** address. An address field can be 1 byte or several bytes long, depending on the needs of the network.

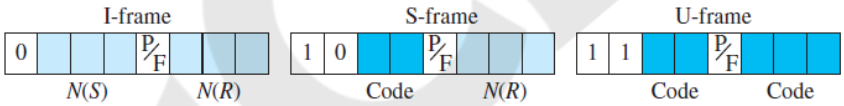
Control field -The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type.

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.

FCS field-The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte ITU-T CRC.

Control Field

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



Control Field for I-Frames

I-frames are designed to carry user data from the network layer. In addition, they can include flow and error control information (piggybacking). The subfields in the control field are used to define these functions.

- The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame.
- The next 3 bits, called *N(S),* define the sequence number of the frame.
- The last 3 bits, called *N(R),* correspond to the acknowledgment number when piggybacking is used.
- The *PIF* field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary. It means *final* when the frame is sent by a secondary to a primary.

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate. S-frames do not have information fields. If the first 2 bits of the control field is 10, this means the frame is an S-frame. The last 3 bits, called *N(R),* corresponds to the acknowledgment number(ACK) or negative acknowledgment number (NAK) depending on the type of S-frame.The 2 bits called code is used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

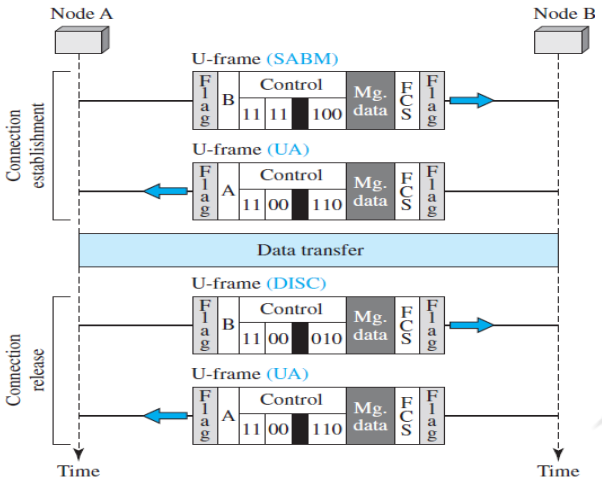
1. Receive ready (RR). If the value of the code subfield is 00, it is an RR S-frame. Receive not ready (RNR). If the value of the code subfield is 10, it is an RNR S-frame. The value of *NCR* is the acknowledgment number.
2. Reject (REJ). If the value of the code subfield is 01, it is a REJ S-frame. The value of *NCR)* is the negative acknowledgment number.
3. Selective reject (SREJ). If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. The value of *N(R)* is the negative acknowledgment number.

Control Fieldfor U-Frames

Unnumbered frames are used to exchange session management and control information between connected devices.

.U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

Figure shows how U-frames can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (UA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a UA (unnumbered acknowledgment).



POINT-TO-POINT PROTOCOL

Although HDLC is a general protocol that can be used for both point-to-point and multipoint configurations, the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP). PPP is by far the most common. PPP provides several services:

1. PPP defines the format of the frame to be exchanged between devices.PPP defines how two devices can negotiate the establishment of the link and the exchange of data.
2. PPP defines how network layer data are encapsulated in the data link frame.
3. PPP defines how two devices can authenticate each other.
4. PPP provides multiple network layer services supporting a variety of network layer protocols.
5. PPP provides connections over multiple links.
6. PPP provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

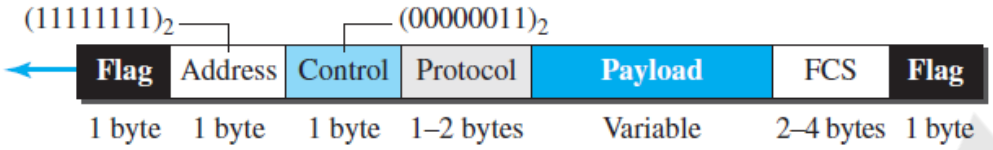
On the other hand, to keep PPP simple, several services are missing:

1. PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver.
2. PPP has a very simple mechanism for error control.
3. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

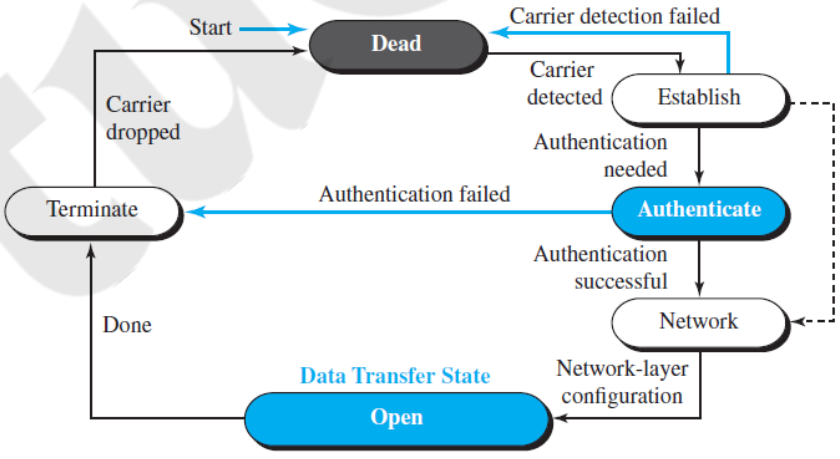
Framing

PPP is a byte-oriented protocol. Framing is done according byte oriented protocols.

Frame Format



Flag.- A PPP frame starts and ends with a I-byte flag with the bit pattern 01111110.
Address-The address field in this protocol is a constant value and set to 11111111 (broadcast address). During negotiation (discussed later), the two parties may agree to omit this byte.
Control.-This field is set to the constant value 11000000.
Protocol-The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only I1 byte.
Payload field- This field carries either the user data or other information. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation.
FCS-The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.
Transition Phases
A PPP connection goes through phases which can be shown in a transition phase diagram.



Dead-In the dead phase the link is not being used. There is no active carrier and the line is quiet.

Establish- When one of the nodes starts the communication, the connection goes into this phase. In this phase, options are negotiated between the two parties..

Authenticate- The authentication phase is optional; the two nodes may decide, during the establishment phase, not to skip this phase.

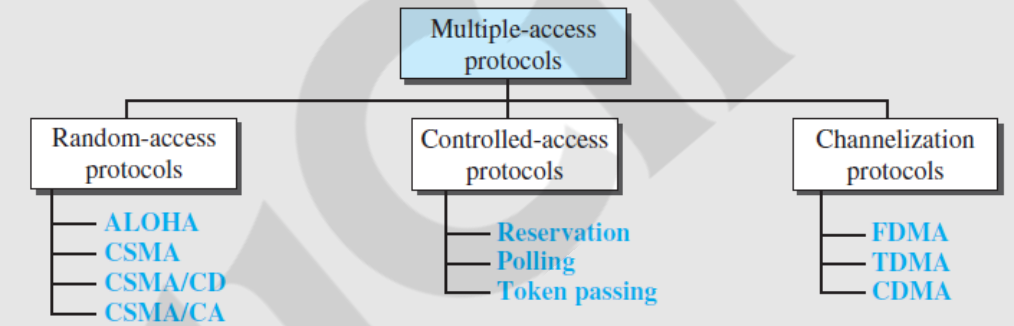
Network- In the network phase, negotiation for the network layer protocols takes place.

Open. In the open phase, data transfer takes place.

Terminate. In the termination phase the connection is terminated.

Media Access control

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.Many protocols have been devised to handle access to a sharedlink. All of these protocols belong to a sublayer in the data-link layer called *media accesscontrol (MAC).* We categorize them into three groups.



RANDOM ACCESS

In random access or contention methods, no station is superior to another station. A station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send.

Two features give this method its name.

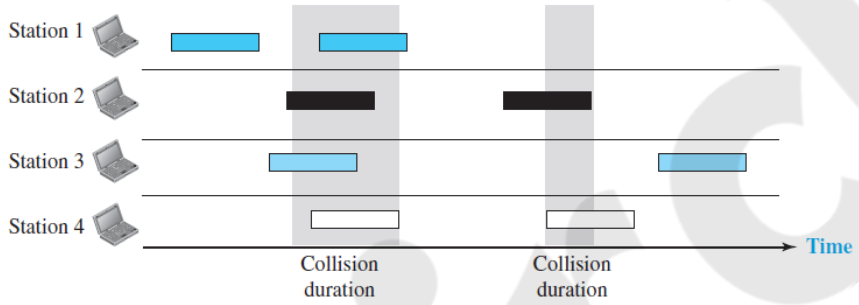
1. There is no scheduled time for a station to transmit. Transmission is random and hence called *random access.*
2. *N*o rules specify which station should send next. Station scompete with one another to access the medium and hence called *contention* methods. If more than one station tries to send, there isan access conflict-collision-and the frames will be either destroyed or modified.

1. ALOHA

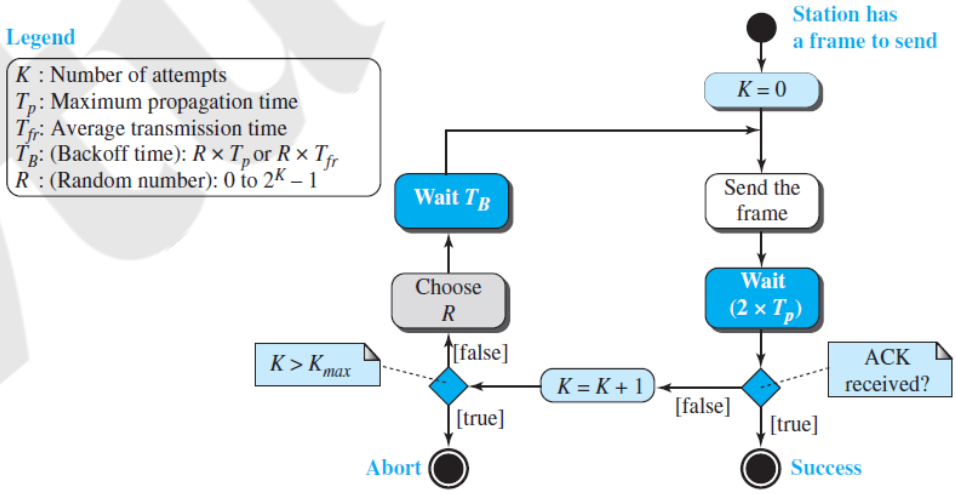
ALOHA, the earliest random access method was developed at the University of Hawaii in early 1970. The medium is shared between the stations and hence has potential arrangements.

Pure ALOHA

- ➔ The original ALOHA protocol is called pure ALOHA. The idea is that each station sends a frame whenever it has a frame to send.
- ➔ Since there is only one channel to share, there is the possibility of collision between frames from different stations.
- ➔ There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. Only two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3.

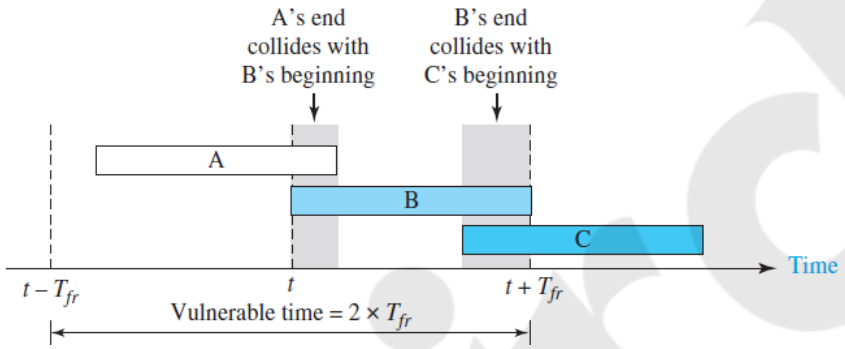


- ➔ The pure ALOHA protocol relies on acknowledgments from the receiver. If the acknowledgment does not arrive after a time-out period, the station resends the frame.
- ➔ If all these stations try to resend their frames after the time-out, the frames will collide again. Each station waits for a random time before resending, called the back-off time T_B.
- ➔ In the second method after a maximum number of retransmission attempts K_{max}, a station must give up and try later.



Vulnerable time- Station A sends a frame at time *t*. Now imagine station B has already sent a frame between *t* - *T*frand *t*. This leads to a collision between the frames from station A and station B. The end of B's frame collides with the beginning of A's frame. On the other hand, suppose that station C sends a frame between *t* and *t* + *Tfr* .Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.

Pure ALOHA vulnerable time = 2 x Tfr

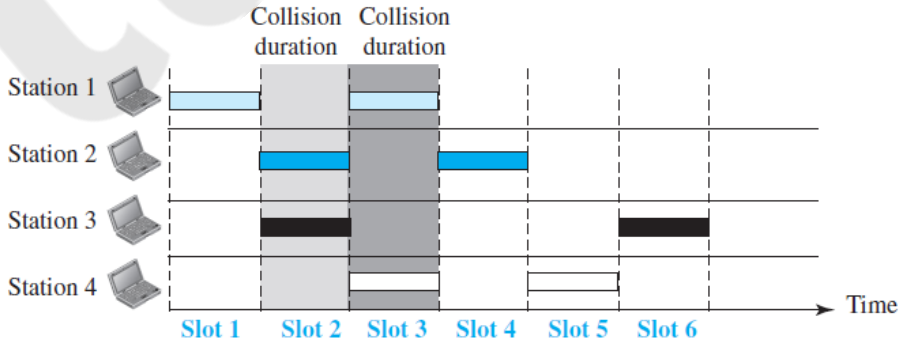


Throughput Let us call G the average number of frames generated by the system during one frame transmission time. Then it can be proved that the average number of successful transmissions for pure ALOHA is S=G x *e*-*2G*. The maximum throughput

S_{max} is 0.184, for G =1.

2. Slotted ALOHA

In slotted ALOHA we divide the time into slots of Tfrs and force the station to send only at the beginning of the time slot.

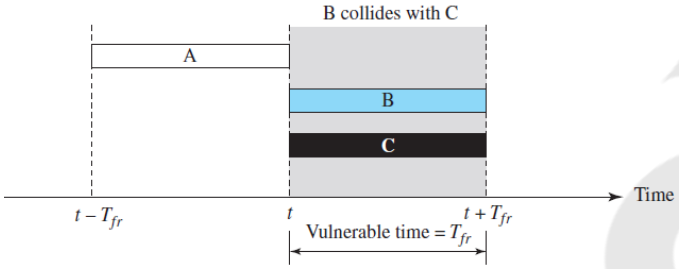


The vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

Slotted ALOHA vulnerable time = T_{fr}

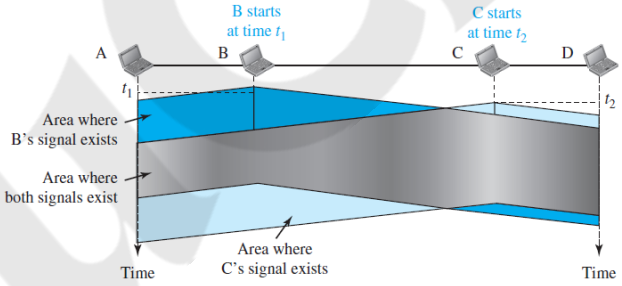
Throughput It can be proved that the average number of successful transmissions for ALOHA is

$S = G \times e^{-G}$. The maximum throughput S_{max} is 0.368, when G = 1.



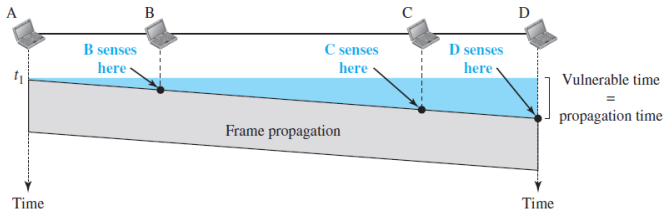
Carrier Sense Multiple Access (CSMA)

- ➔ Carrier sense multiple access (CSMA) requires that each station first listen to the medium before sending. The principle "sense before transmit" or "listen before talk."
- ➔ CSMA can reduce the possibility of collision, but it cannot eliminate it. Stations are connected to a shared channel.
- ➔ The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time for the first bit to reach every station and for every station to sense it.



Vulnerable Time

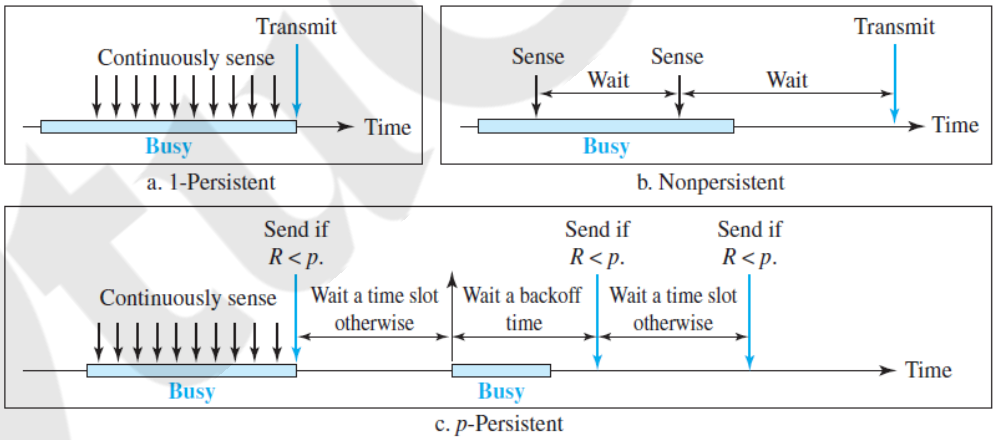
The vulnerable time for CSMA is the propagation time T_p. This is the time needed for a signal to propagate from one end of the medium to the other.

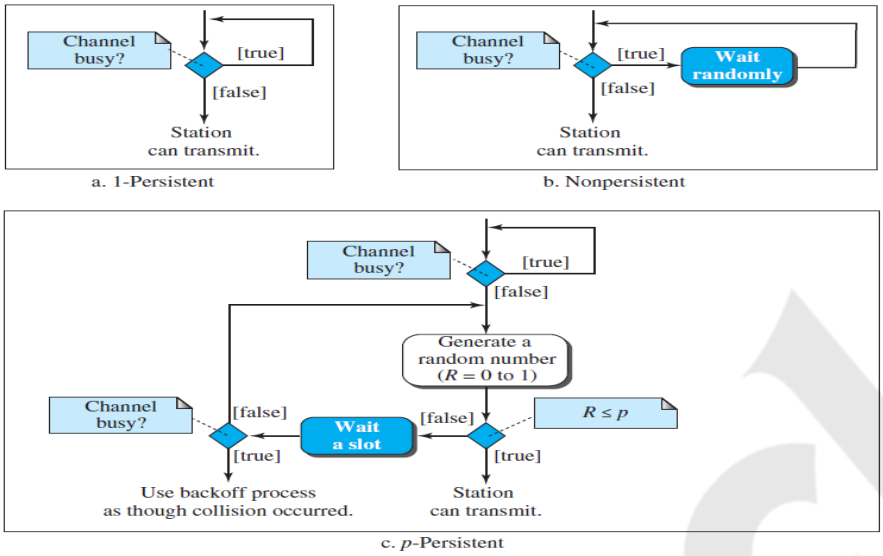


Persistence Methods

This method tells,What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised

1. **I-persistent** - In this method,after the station finds the line idle, it sends its frame immediately.
2. **Nonpersistent-** In this method**,** a station that has a frame to sendsenses the line. If the line is idle, it sends immediately. If the line is not idle, it waits arandom amount of time and then senses the line again.
3. **p-Persistent** -This method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. In this method, 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 beginning of 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 backoff procedure

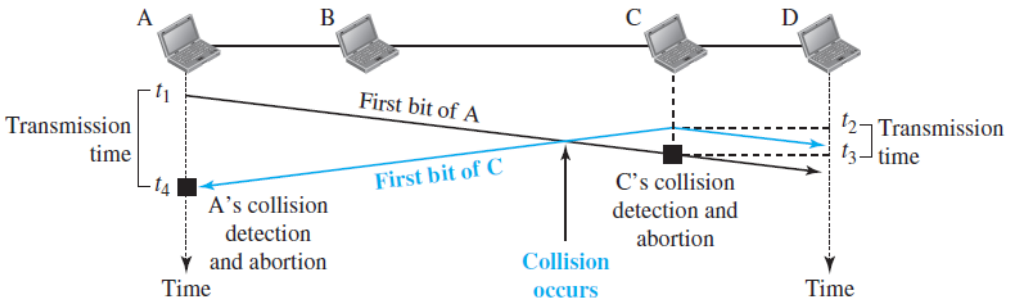




4. Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

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.

1. At time *t* 1, station A has executed its persistence procedure and starts sending the bits of its frame.
2. At time *t2,* 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.
3. The collision occurs sometime after time *t2'* Station Cdetects a collision at time *t3* when it receives the first bit of A's frame.
4. Station C immediatelyaborts transmission.
5. Station A detects collision at time *t4* when it receives the first bit of C's frame; it also immediately aborts transmission.



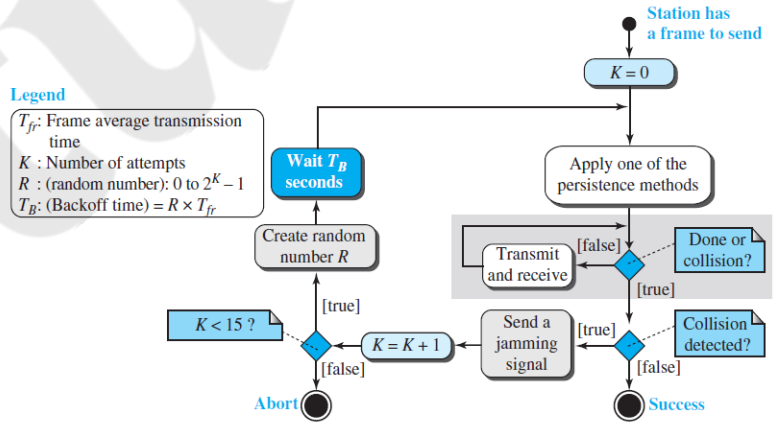
Minimum Frame Size

- ➔ For *CSMAlCD*to work, a restriction on the frame size is requored.
- ➔ Before sending the last bit of the frame, the sending station must detect a collision, if any, because it does not keep a copy of the frame.
- ➔ Therefore, the frame transmissiontime *T*frmust be at least two times the maximum propagation time *Tp.*

Procedure

It is similar to theone for the ALOHA protocol, but there are differences.

1. Addition of the persistence process.
2. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In *CSMA/CD,* transmission andcollision detection is a continuous process.
3. A short jamming signal that enforces the collisionin case other stations have not yet sensed the collision.

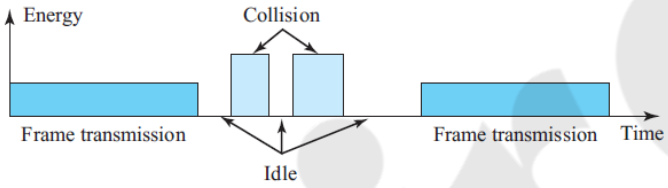


Energy Level

The level of energy in a channel can have three values: zero, normal, and abnormal.

- ➔ At the zero level, the channel is idle.
- ➔ At the normal level, a station has successfully captured the channel and is sending its frame.
- ➔ At the abnormal level, thereis a collision and the level of the energy is twice the normal level.

A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode.



Throughput

The throughput of *CSMAlCD*is greater than that of pure or slotted ALOHA. The maximum throughput occurs at a different value of G and is based on the persistence method and the value of *p* in the p-persistent approach.

1. For I-persistent method the maximumthroughput is around 50 percent when G =1.
2. For nonpersistent method, the maximumthroughput can go up to 90 percent when G is between 3 and 8

5. Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

The basic idea behind *CSMA/CD* is that a station needs to be able to receive while transmitting to detect a collision.

- ➔ When there is no collision, the station receives onesignal: its own signal. When there is a collision, the station receives two signals: its ownsignal and the signal transmitted by a second station.
- ➔ To distinguish between these twocases, the received signals in these two cases must be significantly different.

➔ In a wired network, the received signal has almost the same energy. In case of collision, the detectedenergy almost doubles.

➔ In a wireless network, much of the sent energy is lost in transmission. Thereceived signal has very little energy. Collision may add very little additional energy. This is not useful for effective collision detection.

➔ Collision must be avoided on wireless network, using the following strategies

1. ***Interframe Space (IFS)***

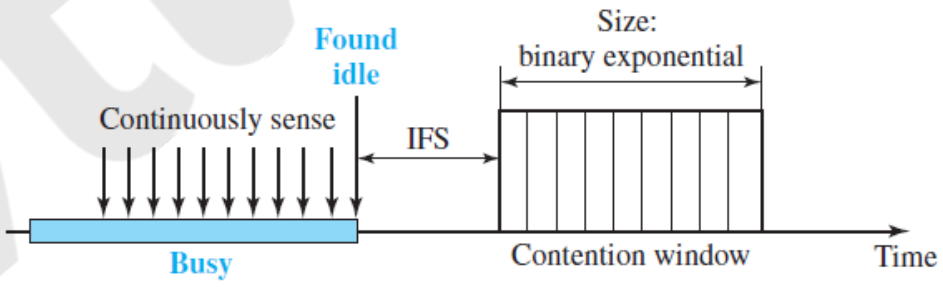
   Collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the interframe space or IFS.

2. ***Contention Window***

   The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential back-off strategy(set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time).

3. ***Acknowledgment***

   With all these precautions, there still may be a collision resulting in destroyed data.In addition, the data may be corrupted during the transmission. The positive acknowledgmentand the time-out timer can help guarantee that the receiver has received the frame.

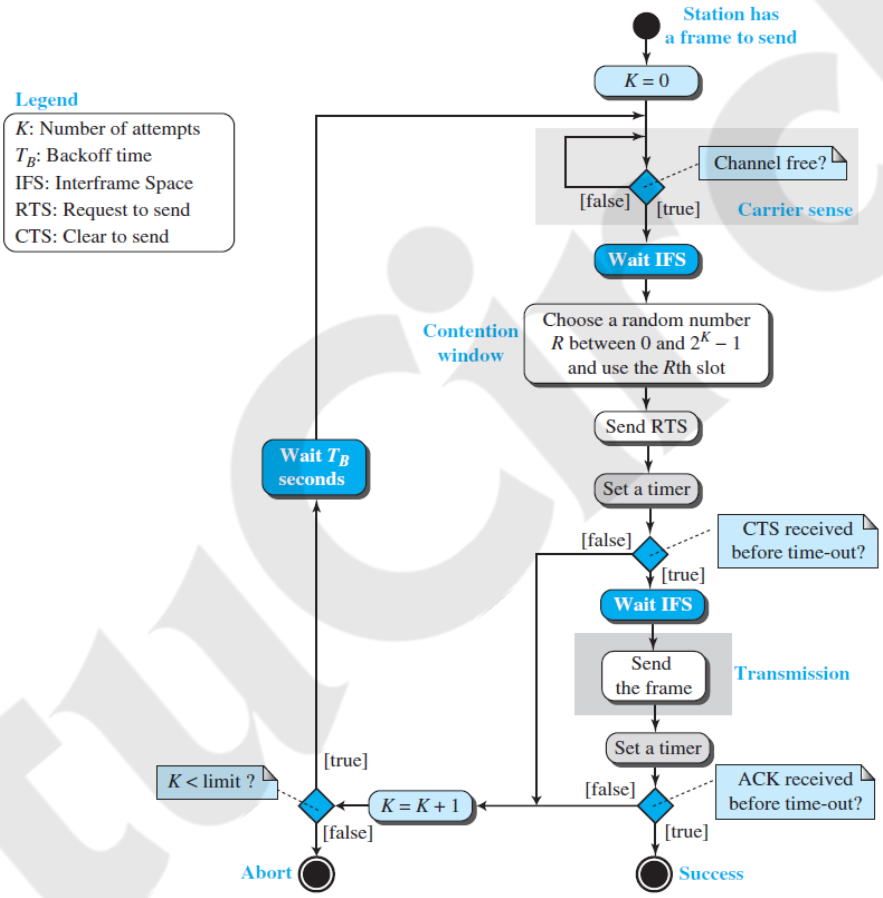


Procedure

➔ The channel needs to be sensed before and after the IFS. The channel also needs to be sensed during the contention time.

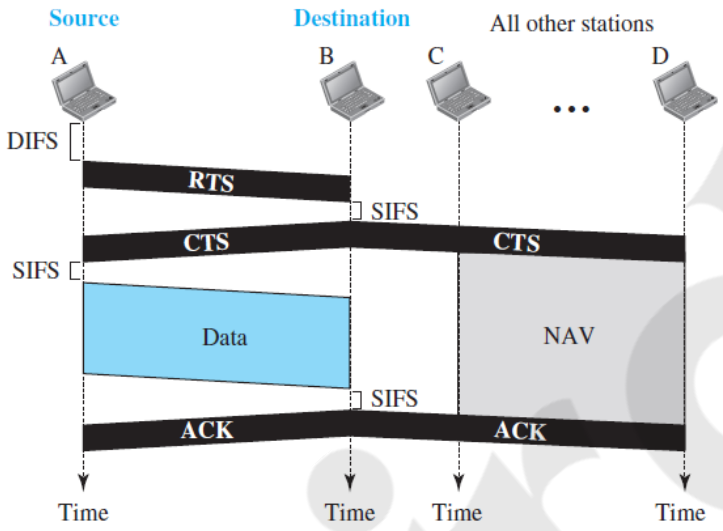
➔ For each time slot of the contention window, the channel is sensed.

➔ If it is found idle, the timer continues; if the channel is found busy, the timer is stopped and continues after the timer becomes idle again.



Frame Exchange Time Line

Figure shows the exchange of data and control frames in time.



1. Before sending a frame, the source station senses the medium by checking theenergy level at the carrier frequency.

   a. The channel uses a persistence strategy with backoff until the channel is idle.

   b. After the station is found to be idle, the station waits for a period of time calledthe *DCF interframe space (DIFS);* then the station sends a control frame calledthe *request to send (RTS).*

2. After receiving the RTS and waiting a period of time called the *short interframespace (SIFS),* the destination station sends a control frame, called the *clear tosend (CTS),* to the source station. This control frame indicates that the destinationstation is ready to receive data**.**

3. The source station sends data after waiting an amount of time equal to SIFS.

4. **.** The destination station, after waiting an amount of time equal to SIFS, sends anacknowledgment to show that the frame has been received.

Network Allocation Vector

Collision avoidance aspect of this protocol is accomplishedby a feature called *NAV.*

- ➔ When a station sends an RTS frame, it includes the duration of time that it needs to occupy the channel.
- ➔ The stations that are affected by this transmission create a timer called a **network allocation vector (NAV)** that shows how much time must pass before these stations are allowed to check the channel for idleness.
- ➔ Each station, before sensing the physical medium to see if it is idle, first checks its NAV to see if it has expired.

Collision During Handshaking

Two or more stations may try to send RTS frames at the same time. These control frames may collide. However,because there is no mechanism for collision detection, the sender assumes there has been a collision if it has not received a CTS frame from the receiver. The back off strategy is employed, and the sender tries again.

CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations.

1. **Reservation**
   - ➔ In the reservation method, a station needs to make a reservation before sending data.
   - ➔ Time is divided into intervals.
   - ➔ In each interval, a reservation frame precedes the data frames sent in that interval.

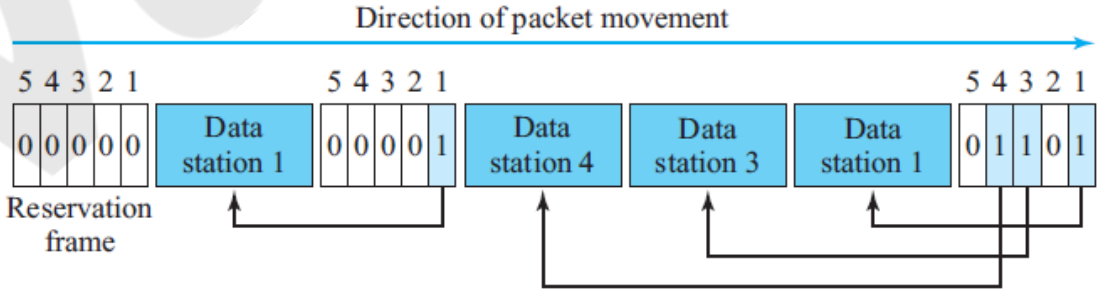


Figure shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

2. Polling

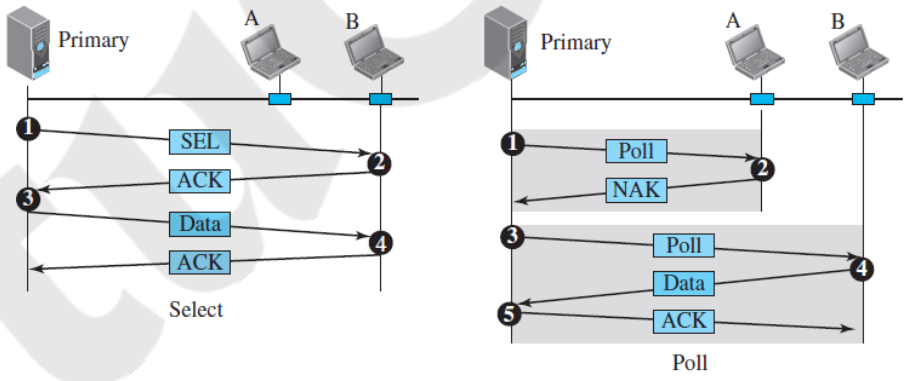
➔ Polling works with topologies in which one device is designated as a primary station and the other devices are secondary stations.

➔ The primary device controls the link; the secondary devices follow its instructions.

➔ The primary device, therefore, is always the initiator of a session

➔ If the primary wants to receive data, it asks the secondary's if they have anything tosend; this is called **poll** function. Secondary responds with DATA if any ,else negative ack (NAK) if it has nothing to send.

➔ If the primary wants to send data, it tells the secondaryto get ready to receive; this is called **select** function. It transmits SEL and waits for acknowledgement from secondary before sending data.



3. Token Passing

➔ In the token-passing method, the stations in a network are organized in a logical ring.

➔ A special packet called a token circulates through the ring.

➔ The possession of the token gives the station the right to access the channel and send its data.

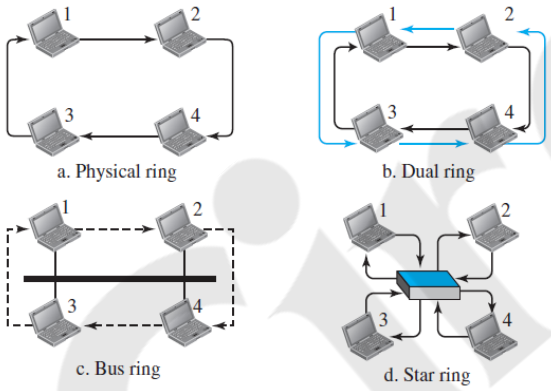
➔ When a station has some data to send, it holds the token and sends its data and releases the token once data is sent.

➔ When a station receives the token and has no data to send, it just passes the data to the next station.

➔ Stations must be limited in the time they can have possession of the token.

➔ The token must be monitored to ensure it has not been lost or destroyed.

➔ Stations do not have to be physically connected in a ring; the ring can be a logical one.



➔ The problem with this topology is that if one of the links-the medium between two adjacent stations fails, the whole system fails.

➔ The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring.