Unit 3

Data link layer: Error detection And correction: Framing, flow and error control

Types of Errors

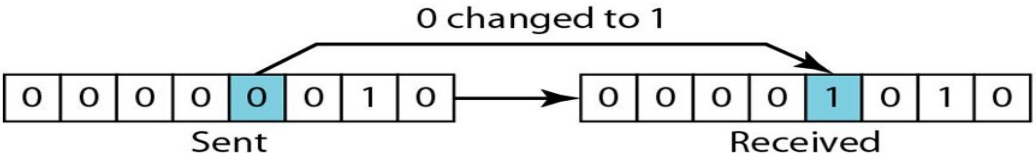
Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal. In a single-bit error, a 0 is changed to a 1 or a 1 to a O. In a burst error, multiple bits are changed. For eg,, a 11100 s burst of impulse noise on a transmission with a data rate of 1200 bps might change all or some of the12 bits of information.

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.

Figure 10.1 shows the effect of a single-bit error on a data unit. To understand the impact of the change, imagine that each group of 8 bits is an ASCII character with a 0 bit added to the left. In Figure 10.1,00000010 (ASCII *STX)* was sent, meaning *start of text,* but 00001010 (ASCII *LF)* was received, meaning *line feed.*

Figure 10.1 *Single-bit error*

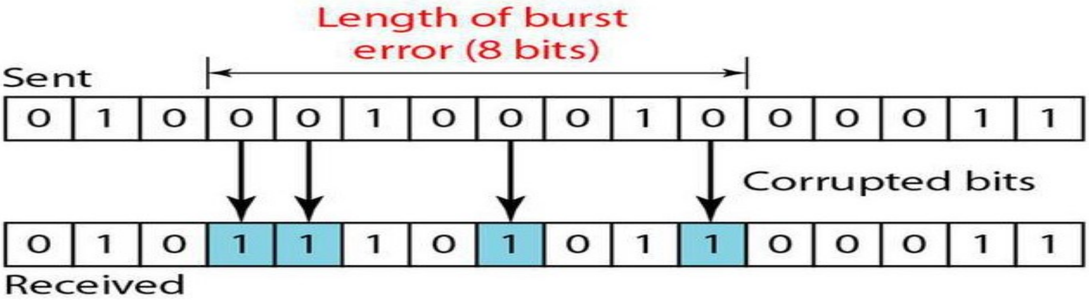


Burst Error

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.

Figure 10.2 shows the effect of a burst error on a data unit. In this case, 0100010001000011 was sent, but 0101110101100011 was received. Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

Figure 10.2 *Burst error of length 8*



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.

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. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.

In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors. If we need to correct one single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 possibilities.

Forward Error Correction versus Retransmission

There are two main methods of error correction.

Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. This is possible if the number of errors is small.

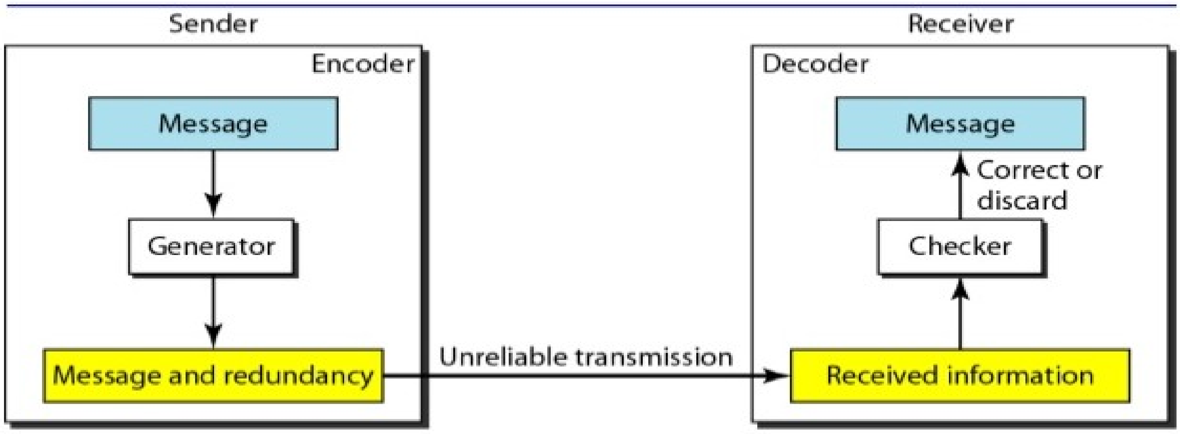
Correction by retransmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free (usually, not all errors can be detected).

Coding

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect or correct the errors. The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme. Figure 10.3 shows the general idea of coding.

We can divide coding schemes into two broad categories:
block coding and convolution coding.

Figure 10.3 *The structure of encoder and decoder*



To detect or correct errors, we need to send extra (redundant) bits with data.

Modular Arithmetic

In modular arithmetic, we use only a limited range of integers. We define an upper limit, called a modulus *N.* We then use only the integers 0 to *N* - I, inclusive. This is *modulo-N* arithmetic. For example, if the modulus is 12, we use only the integers 0 to 11, inclusive.

An example of modulo arithmetic is our clock system. It is based on modulo-12 arithmetic, substituting the number 12 for O. In a *modulo-N* system, if a number is greater than *N,* it is divided by *N* and the remainder is the result. If it is negative, as many *Ns* as needed are added to make it positive. Consider our clock system again. If we start a job at 11 A.M. and the job takes 5 h, we can say that the job is to be finished at 16:00 if we are in the military, or we can say that it will be finished at 4 P.M.

In modulo-N arithmetic, we use only the integers in the range 0 to *N* - 1, inclusive.

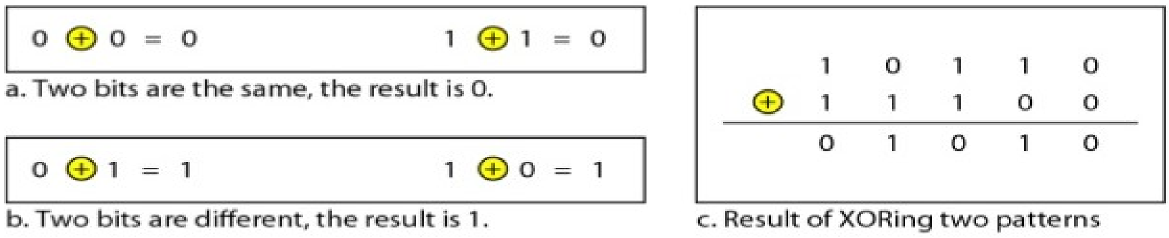
Modulo-2 Arithmetic

Of particular interest is modulo-2 arithmetic. In this arithmetic, the modulus *N* is 2. We can use only 0 and 1. Operations in this arithmetic are very simple. The following shows how we can add or subtract 2 bits.

Adding:	0+0=0	0+1=1	1+0=1	1+1=0
Subtracting:	0-0=0	0-1=1	1-0=1	1-1=0

In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is I if two bits are different. Figure 10.4 shows this operation.

Figure 10.4 ***XORing of two single bits or two words***



a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.

c. Result of XORing two patterns

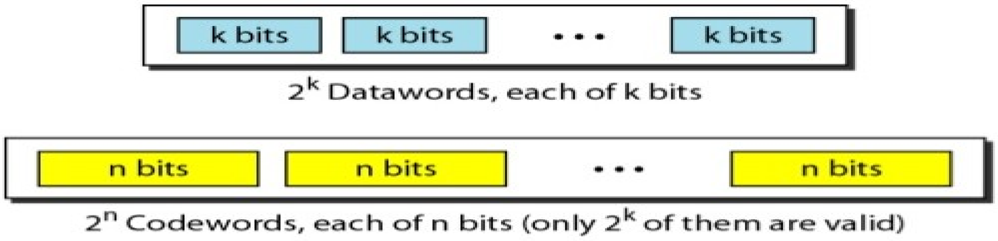
BLOCK CODING

In block coding, we divide our message into blocks, each of *k* bits, called datawords. We add *r* redundant bits to each block to make the length *n* = *k* + *r.* The resulting *n-bit* blocks are called **codewords**.We have a set of datawords, each of size *k,* and a set of codewords, each of size of *n.* With *k* bits, we can create a combination of 2_k datawords; with *n* bits, we can create a combination of 2_n codewords.

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

The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have $2n - 2k$ codewords that are not used. We call these codewords invalid or illegal. Figure 10.5 shows the situation

Figure 10.5 ***Datawords and codewords in block coding***


k bits
k bits
• • •
k bits
2^k Datawords, each of k bits
n bits
n bits
• • •
n bits
2^n Codewords, each of n bits (only 2^k of them are valid)

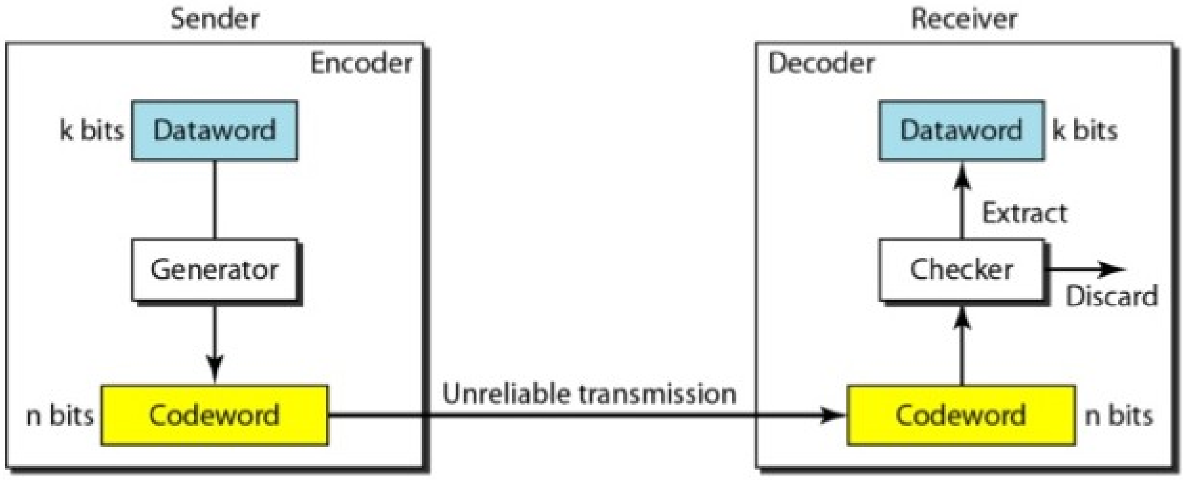
Error Detection

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

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

Figure 10.6 shows the role of block coding in error detection

Figure 10.6 ***Process of error detection in block coding***



The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding .Each codeword sent to the receiver may change during transmission.

If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use.

If the received codeword is not valid, it is discarded.

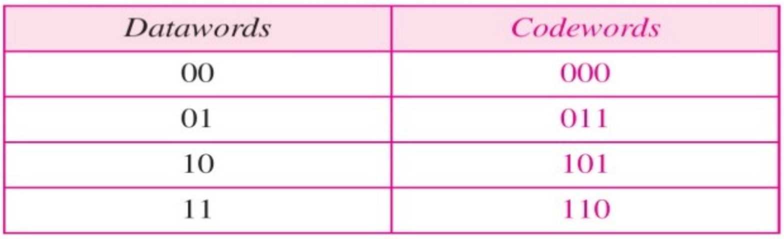
If the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected.

This type of coding can detect only single errors. Two or more errors may remain undetected.

Example

Let us assume that *k* =2 and *n* =3. Table 10.1 shows the list of datawords and codewords.

Table 10.1 ***A code for error detection (Example 10.2)***



Datawords	*Codewords*
00	000
01	011
10	101
11	110

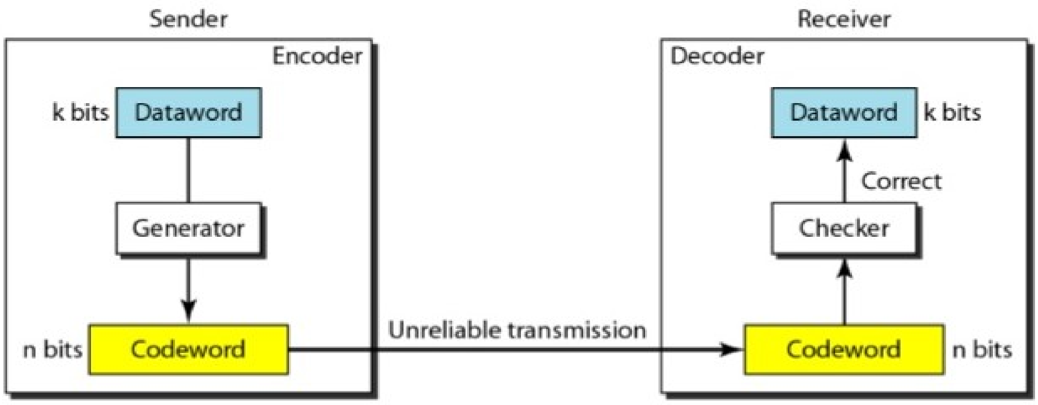
Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

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

Error Correction

In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent. We need more redundant bits for error correction than for error detection. Figure 10.7 shows the role of block coding in error correction. We can see that the idea is the same as error detection but the checker functions are much more complex.

Figure 10.7 ***Structure of encoder and decoder in error correction***



Hamming Distance

One of the central concepts in coding for error control is the idea of the Hamming distance. The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words *x* and *y* as *d(x, y)*.

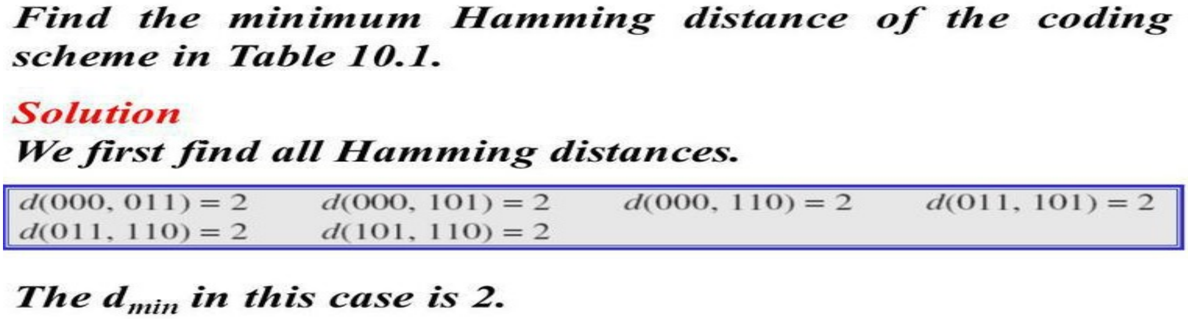
The Hamming distance can easily be found if we apply the XOR operation on the two words and count the number of 1s in the result. Note that the Hamming distance is a value greater than zero.

Let us find the Hamming distance between two pairs of words.
1. The Hamming distance *d(OOO,* 011) is 2 because 000 XOR 011 is 011 (two Is).
2. The Hamming distance *d(10101,* 11110) is 3 because 10101 XOR 11110 is 01011 (three Is).

Minimum Hamming Distance

Although the concept of the Hamming distance is the central point in dealing with error detection and correction codes, the measurement that is used for designing a code is the minimum Hamming distance.

The minimum Hamming distance is the smallest Hamming distance between all possible pairs. We use d_{min} to define the minimum Hamming distance in a coding scheme. To find this value, we find the Hamming distances between all words and select the smallest one.



Find the minimum Hamming distance of the coding scheme in Table 10.1.

Solution

We first find all Hamming distances.

$d(000, 011) = 2$ $d(000, 101) = 2$ $d(000, 110) = 2$ $d(011, 101) = 2$
$d(011, 110) = 2$ $d(101, 110) = 2$

The d_{min} in this case is 2.

LINEAR BLOCK CODES

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Minimum Distance for Linear Block Codes

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

Example 10.11

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is *dmin* =2. In our second code (Table 10.2), the numbers of Is in the nonzero codewords are 3, 3, and 4. So in this code we have *dmin* =3.

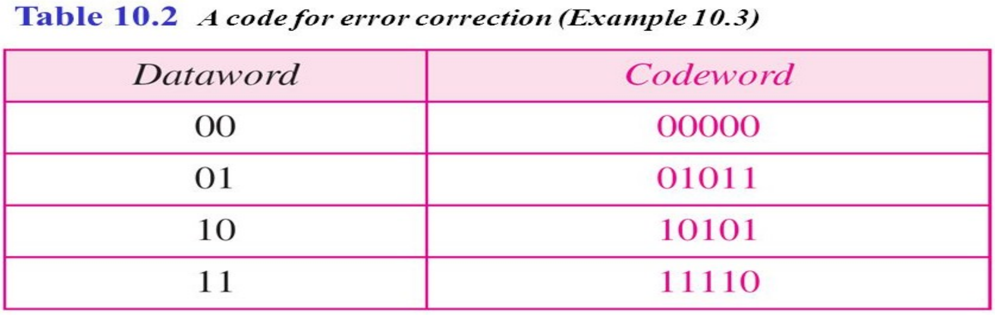


Table 10.2 *A code for error correction (Example 10.3)*

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

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 *ao* to *a6* and the bits in the second word *b0* to *b6,* we can shift the bits by using the following:

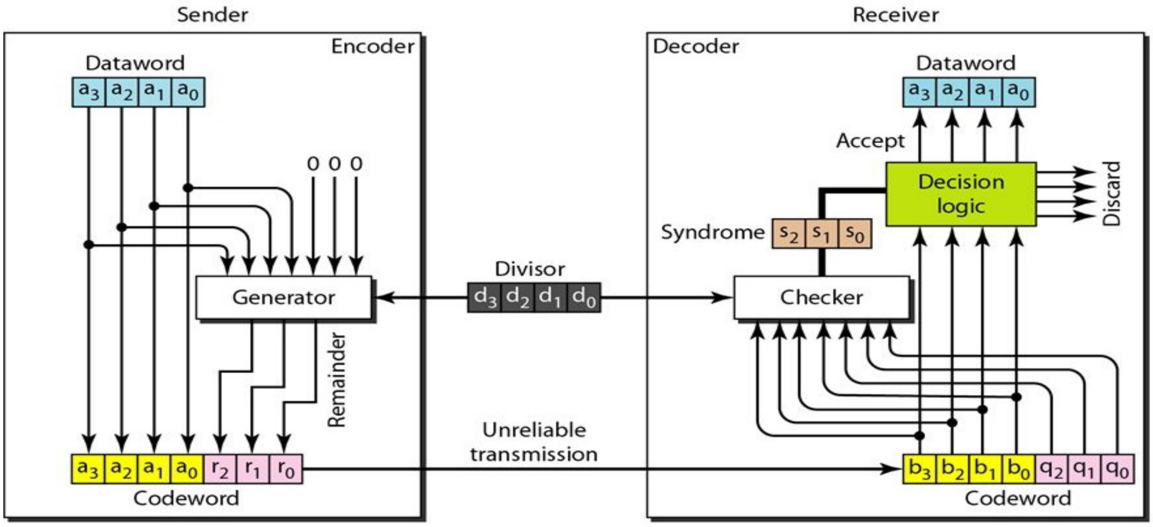
$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check

We can create cyclic codes to correct errors. We discuss a category of cyclic codes called the **cyclic redundancy check** (CRC) that is used in networks such as LANs and WANs.

Figure 10.14 *CRC encoder and decoder*

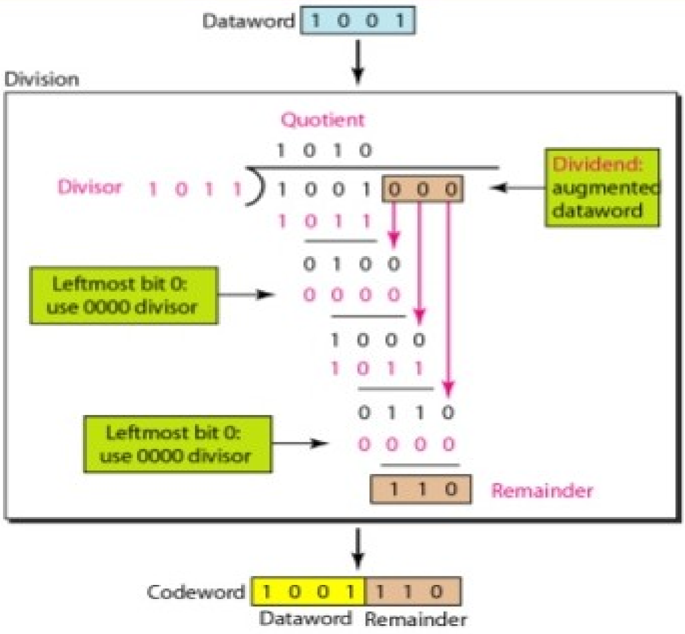


In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here).
The size of the dataword is augmented by adding *n* - *k* (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size *n* - *k* + I (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder *(r2 rl ro)* is appended to the dataword to create the codeword.
The decoder receives the possibly corrupted codeword. 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, which is fed to the decision logic analyzer. 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).

Encoder

The encoder takes the dataword and augments it with *n* - *k* number of a. It then divides the augmented dataword by the divisor, as shown in Figure 10.15.

Figure 10.15 *Division in CRC encoder*



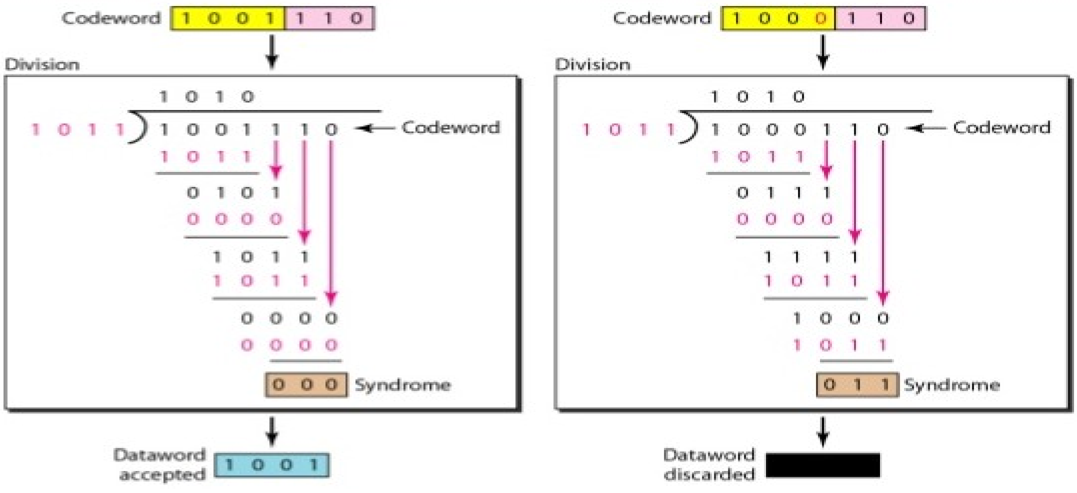
The process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long.If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-0s divisor.

When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits *(r2' rl'* and *ro).* They are appended to the dataword to create the codeword.

Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The **remainder of the division is the syndrome**. If the syndrome is all Os, there is no error; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 10.16 shows two cases: The left hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all Os (it is O11).

Figure 10.16 *Division in the CRC decoder for two cases*



MODULE 3

Framing, flow and error control

FRAMING

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination.

The **physical layer** provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing.

The **data link layer** needs to pack bits into frames, so that each frame is distinguishable from another.

Our postal system practices a type of framing. The simple act o finserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. In addition, each envelope defines the sender and receiver addresses since the postal system is a many-to-many carrier facility.

Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding 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. Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Fixed-Size 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. An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells.

Variable-Size Framing

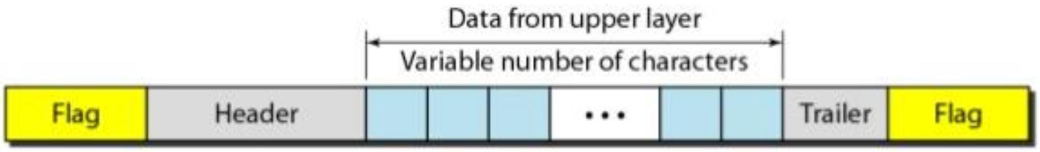
In variable-size framing, we need a way to define the end of the frame and the beginning of the next. Historically, two approaches were used for this purpose:

- a character-oriented approach
- a bit-oriented approach

Character-Oriented Protocols

In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII. The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction 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 beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Figure 11.1 shows the format ofa frame in a character-oriented protocol.

Figure 11.1 A frame in a character-oriented protocol

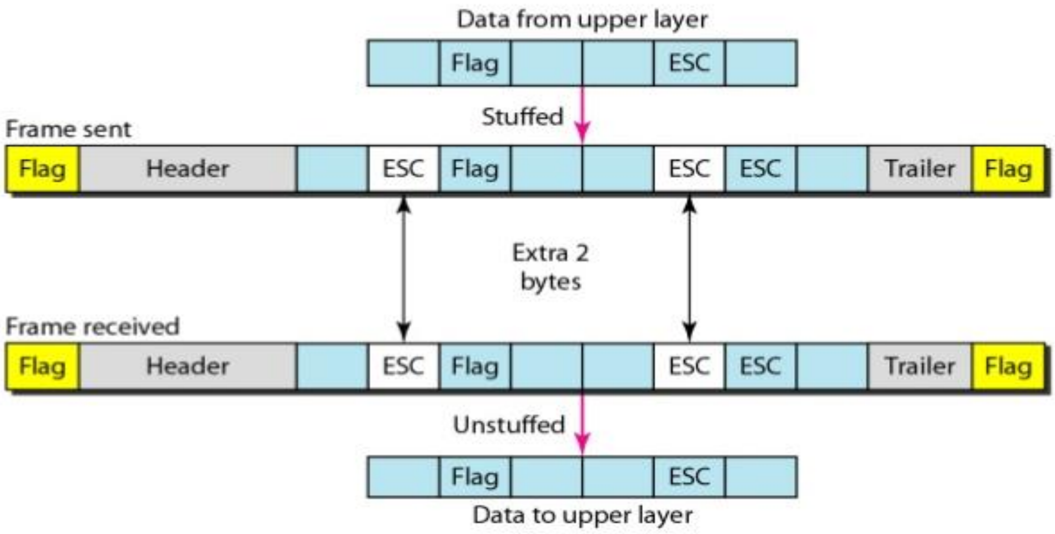


Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication.We send other types of information such as graphs, audio, and video. Any pattern used for the flag could also be part of the information. If this happens, the receiver, when it encounters 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. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which 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 a delimiting flag.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. If the text contains one or more escape characters followed by a flag the receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, 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. Figure 11.2 shows the situation.

Figure 11.2 ***Byte stuffing and unstuffing***



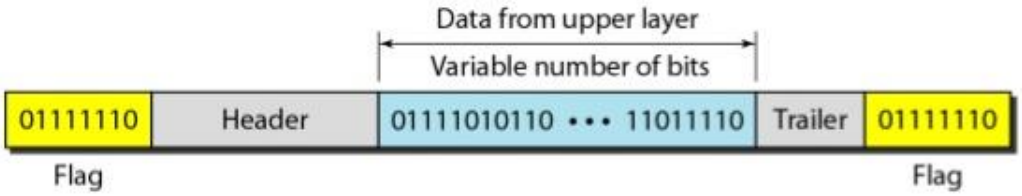
Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters.

Bit-Oriented Protocols

In a bit-oriented protocol, the data section ofa frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers we 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 beginning and the end of the frame, as shown in Figure 11.3.

Figure 11.3 *A frame in a bit-oriented protocol*



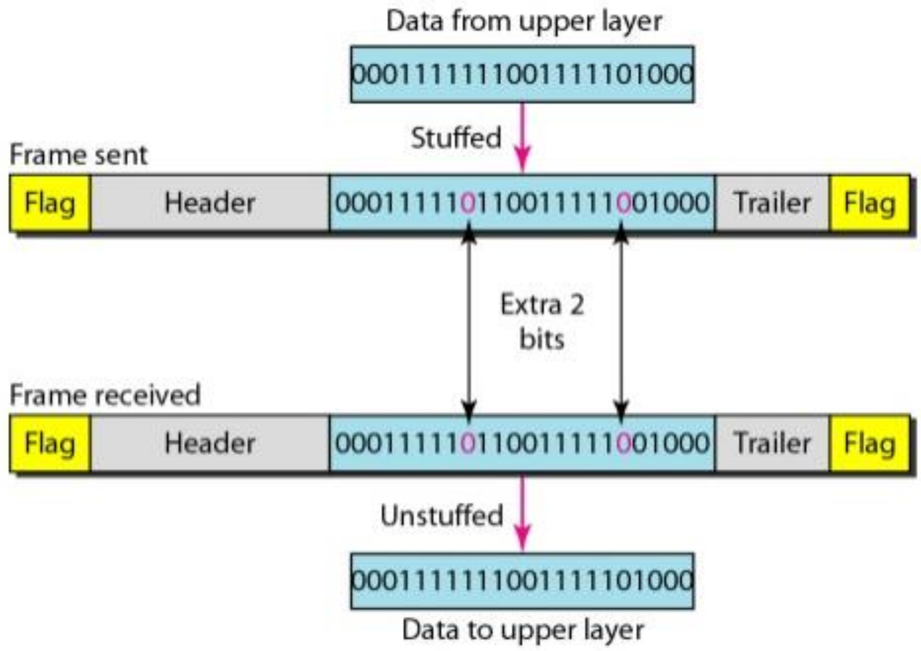
If the flag pattern appears in the data, we need to inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of I byte) to prevent the pattern from looking like a flag. The strategy is called **bit stuffing.**

In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.

Bit stuffing is the process o fadding one extra 0 whenever five consecutive 18 follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver.

Figure 11.4 ***Bit stuffing and unstuffing***



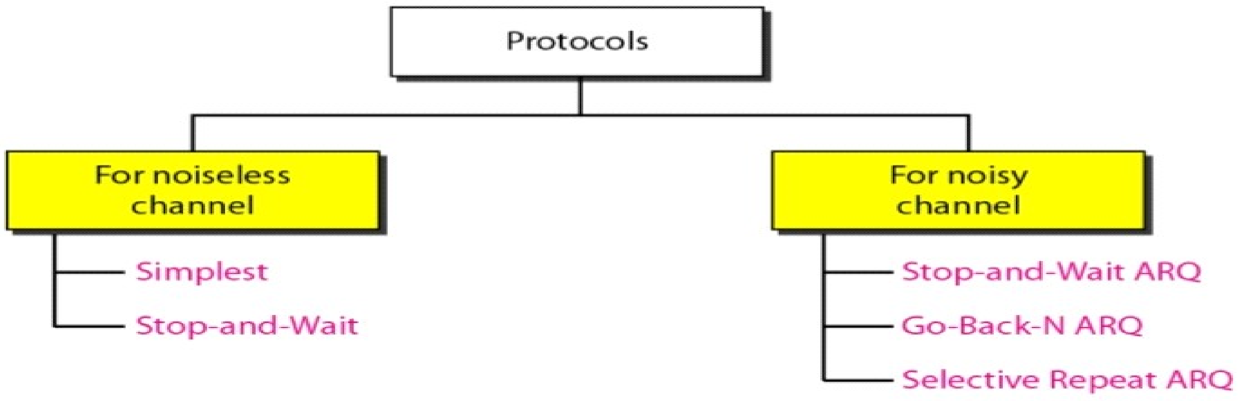
This means that if the flag like pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken as a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

UNIT 3

PROTOCOLS

The protocols can be used for noiseless (error-free) channels and for noisy (error-creating) channels. The protocols in the first category cannot be used in real life, but they serve as a basis for understanding the protocols of noisy channels.

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



All the protocols we discuss are unidirectional in the sense that the data frames travel from one node, called the sender, to another node, called the receiver. Although special frames, called **acknowledgment (ACK) and negative acknowledgment (NAK)** can flow in the opposite direction for flow and error control purposes, data flow in only one direction.

In a real-life network, the data link protocols are implemented as bidirectional; data flow in both directions. In these protocols the flow and error control information such as ACKs and NAKs is included in the data frames in a technique called **piggybacking.**

NOISELESS CHANNELS

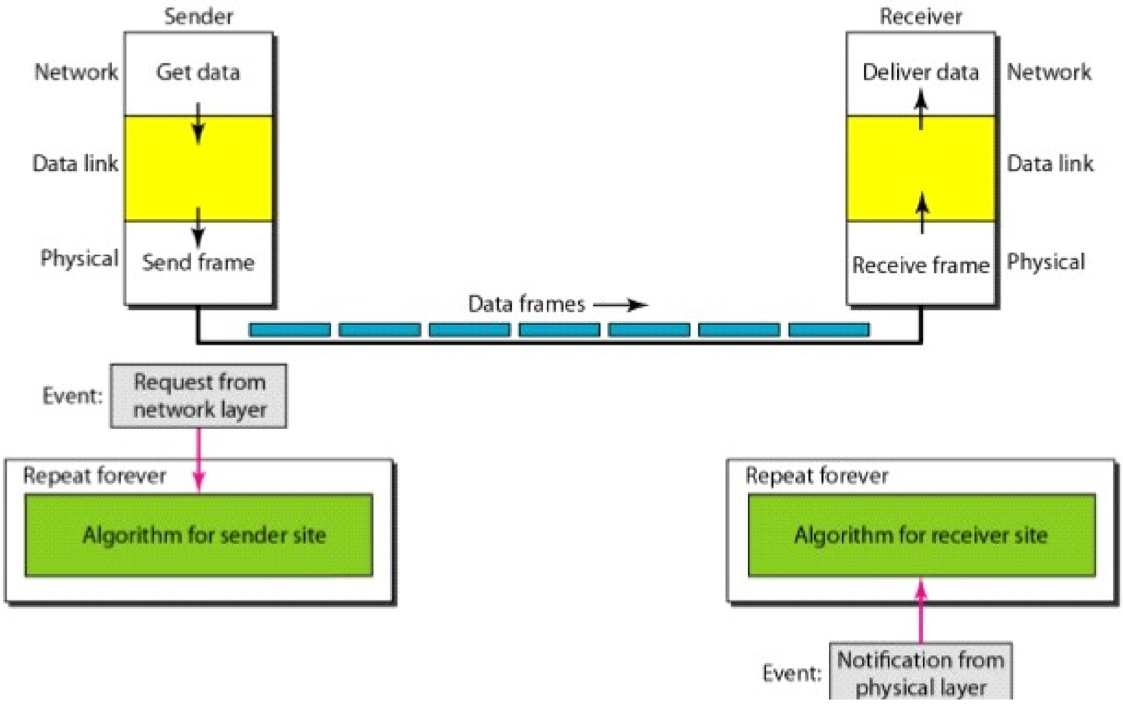
Simplest Protocol

It is a unidirectional protocol in which data frames are travelling in only one direction-from the sender to receiver. The receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately. In other words, the receiver can never be overwhelmed with incoming frames.

Design

There is **no need for flow control** in this scheme. The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it. The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer. The data link layers of the sender and receiver provide transmission services for their network layers. The data link layers use the services provided by their physical layers (such as signaling, multiplexing, and so on) for the physical transmission of bits. Figure 11.6 shows a design.

Figure 11.6 ***The design of the simplest protocol with no flow or error control***

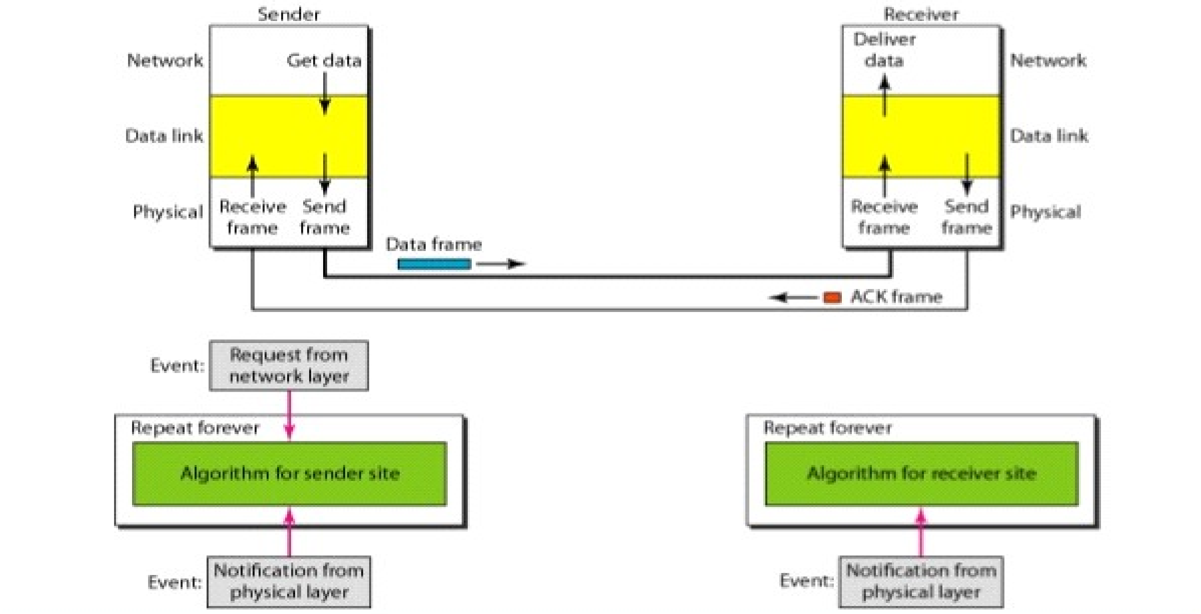


Stop-and-Wait Protocol

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames, we need to tell the sender to slow down. There must be feedback from the receiver to the sender.

The protocol is called the **Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver** (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction.

Figure 11.8 *Design of Stop-and-Wait Protocol*



NOISY CHANNELS

Stop-and-Wait Automatic Repeat Request

Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-and Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver. Lost frames are more difficult to handle than corrupted ones.

In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. **The solution is to number the frames**. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated. The completed and lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, the sender keeps a copy of the sent frame. At the same time, **it starts a timer.** If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number. The ACK frame for this protocol has a sequence number field. In this protocol, the sender simply discards a corrupted ACK frame or ignores an out-of-order one.

Sequence Numbers

A field is added to the data frame to hold the sequence number of that frame. In Stop-and-Wait ARQ we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

Acknowledgment Numbers

The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next). If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

In Stop-and-Wait ARQ the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.

Design

Figure 11.10 shows the design of the Stop-and-Wait ARQ Protocol. The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. A data frames uses a seqNo (sequence number); an ACK frame uses an ackNo (acknowledgment number). The sender has a control variable, which we call Sn (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).

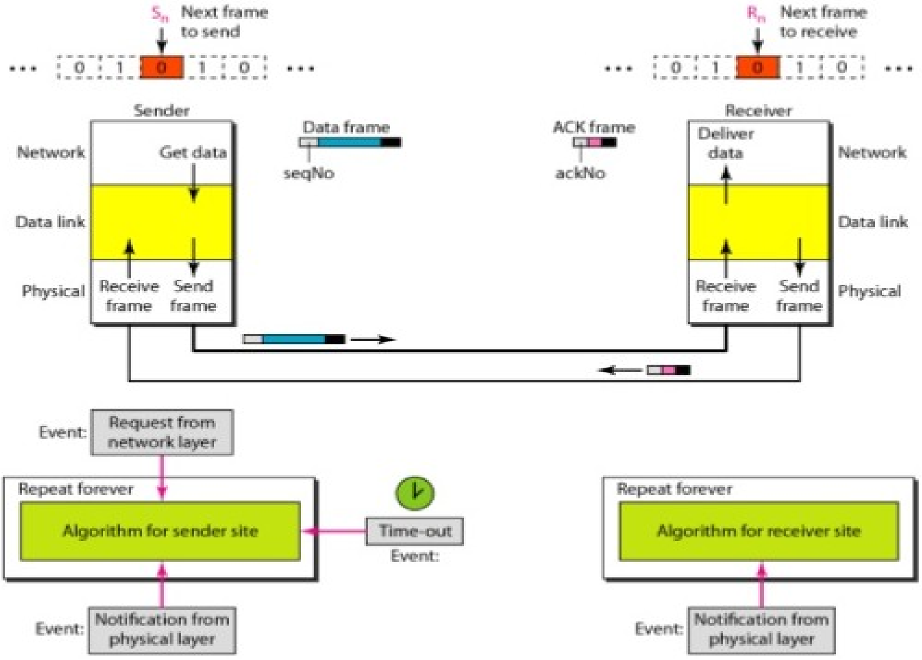


Figure 11.10 Design of the Stop-and-Wait ARQ Protocol

The receiver has a control variable, which we call Rn (receiver, next frame expected), that holds the number of the next frame expected. When a frame is sent, the value of Sn is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. When a frame is received, the value of Rn is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. Three events can happen at the sender site; one event can happen at the receiver site. Variable Sn points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged; Rn points to the slot that matches the sequence number of the expected frame.

Pipelining

In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining. There is no pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent. However, pipelining does apply to our next two protocols because several frames can be sent before we receive news about the previous frames. Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Go-Back-N Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment. The first is called Go-Back-N Automatic Repeat Request. In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sequence Numbers

Frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows m bits for the sequence number, the sequence numbers range from 0 to 2m - 1. For example, if m is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

0, 1,2,3,4,5,6, 7,8,9, 10, 11, 12, 13, 14, 15,0, 1,2,3,4,5,6,7,8,9,10, 11, ...

In other words, the sequence numbers are modulo-2^m.

In the Go-Back-N Protocol, the sequence numbers are modulo 2^m where m is the size of the sequence number field in bits.

Sliding Window

In this protocol (and the next), the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the **send sliding window;** the range that is the concern of the receiver is called **the receive sliding window**.

The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit. In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is $2^m - 1$.

Figure 11.12 shows a sliding window of size 15 (m =4). The window at any time divides the possible sequence numbers into four regions.

The **first region,** from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.

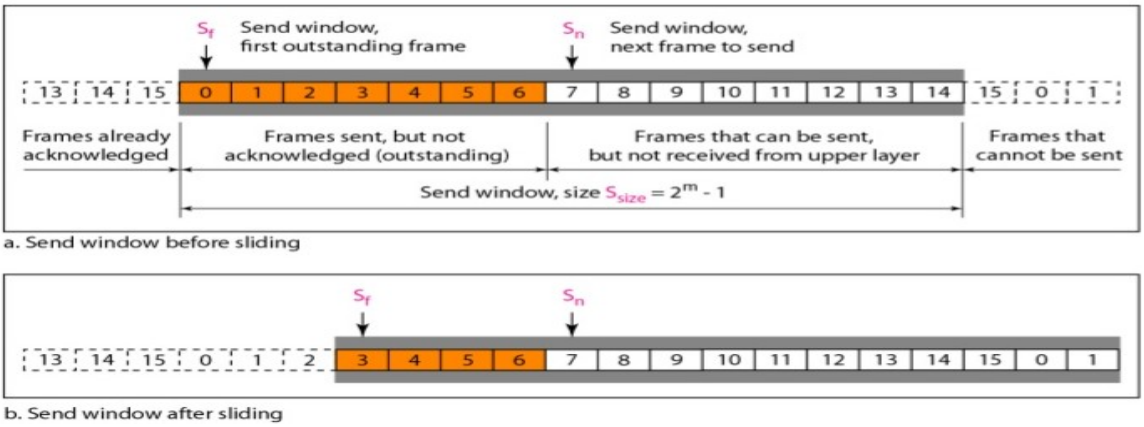
The **second region,** colored in Figure 11.12a, defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. These are the outstanding frames.

The **third range,** white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer.

Finally, the **fourth region** defines sequence numbers that cannot be used until the window slides, as we see next. The window itself is an abstraction; three variables define its size and location at any time.

We call these variables S_f(send window, the first outstanding frame), S_n (send window, the next frame to be sent), and S_{size} (send window, size). The variable S_f defines the sequence number of the first (oldest) outstanding frame. The variable S_n holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable S_{size} defines the size of the window, which is fixed in our protocol.

Figure 11.12 *Send window for Go-Back-N ARQ*



S_f Send window, first outstanding frame
S_n Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged (outstanding)
Frames that can be sent, but not received from upper layer
Frames that cannot be sent
Send window, size $S_{size} = 2^m - 1$
a. Send window before sliding
S_f
S_n
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
b. Send window after sliding

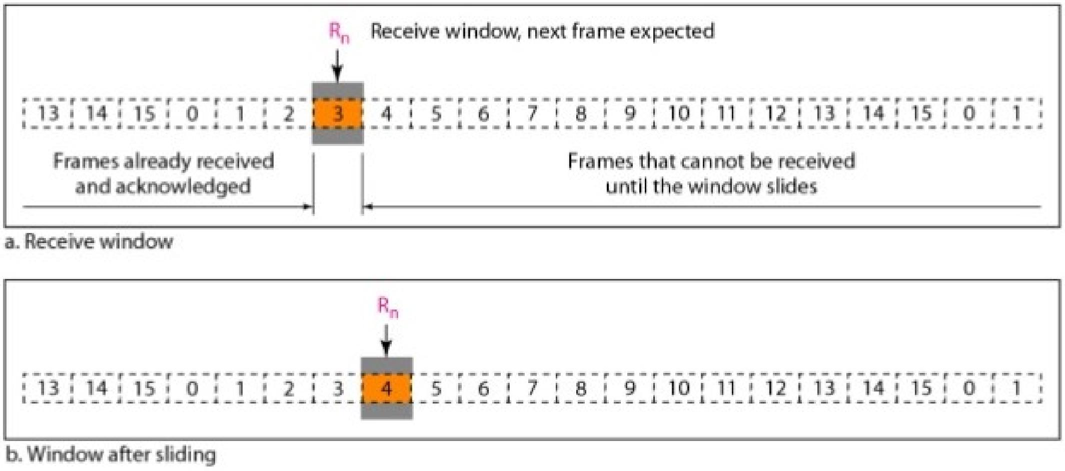
The send window is an abstract concept defining an imaginary box of size 2^m - 1 with three variables: Sp Sm and Ssize.

Figure 11.12b shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. As we will see shortly, the acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure 11.12b, frames 0, I, and 2 are acknowledged, so the window has slide to the right three slots. Note that the value of Sf is 3 because frame 3 is now the first outstanding frame.

The send window can slide one or more slots when a valid acknowledgment arrives.

The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent. The size of the receive window is always 1. The receiver is always looking for the arrival of a specific frame. Any frame arriving out of order is discarded and needs to be resent. Figure 11.13 shows the receive window.

Figure 11.13 *Receive window for Go-Back-N ARQ*



The receive window is an abstract concept defining an imaginary box ofsize 1 with one single variable Rn.The window slides when a correct frame has arrived; sliding occurs one slot at a time.

we need only one variable Rn (receive window, next frame expected) to define this abstraction. The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of Rn is accepted and acknowledged. The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.

Acknowledgment

The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire. This causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

Resending a Frame

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3, 4,5, and 6 again. That is why the protocol is called Go-Back-NARQ.

Design

Figure 11.14 shows the design for this protocol. As we can see, multiple frames can be in transit in the forward direction, and multiple acknowledgments in the reverse direction. The idea is similar to Stop-and-Wait ARQ; the difference is that the send window allows us to have as many frames in transition as there are slots in the send window.

Send Window Size

We can now show why the size of the send window must be less than 2^m. As an example, we choose m =2, which means the size of the window can be 2^m - 1, or 3. Figure 11.15 compares a window size of 3 against a window size of 4.

If the size of the window is 3 (less than 22) and all three acknowledgments are lost, the frame timer expires and all three frames are resent. The receiver is now expecting frame 3, not frame 0, so the duplicate frame is correctly discarded. On the other hand, if the size of the window is 4 (equal to 22) and all acknowledgments are lost, the sender will send a duplicate of frame 0. However, this time the window of the receiver expects to receive frame 0, so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is an error.

Figure 11.14 *Design of Go-Back-N ARQ*

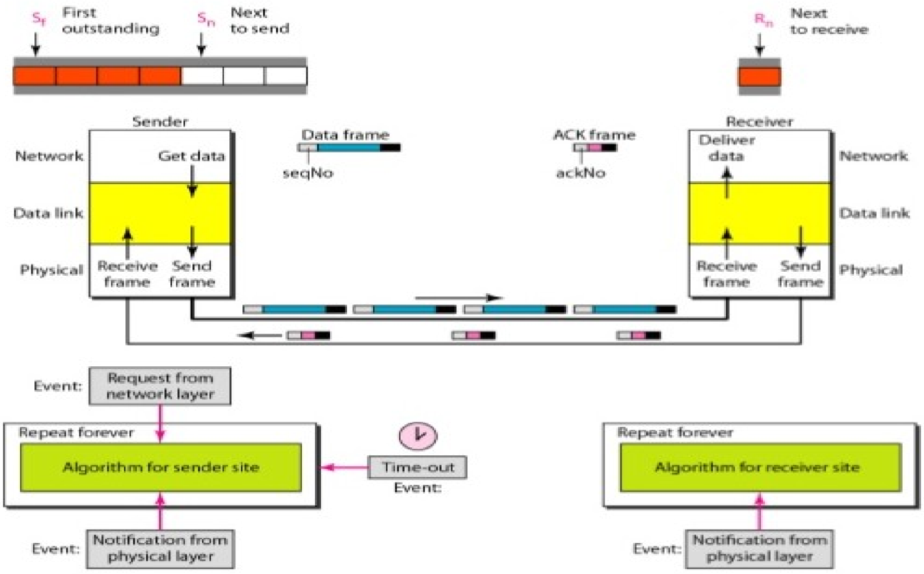
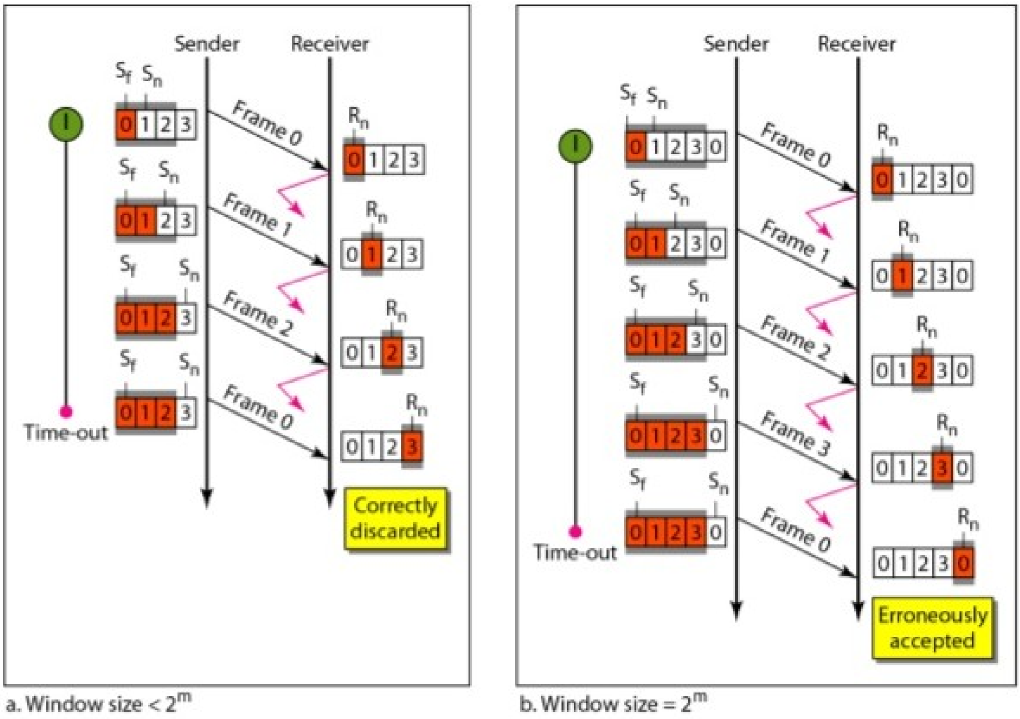


Figure 11.15 *Window size for Go-Back-N ARQ*



In Go-Back-NARQ, the size of the send window must be less than r; the size of the receiver window is always 1.

Selective Repeat Automatic Repeat Request

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ. It is more efficient for noisy links, but the processing at the receiver is more complex.

Windows

The Selective Repeat Protocol also uses two windows:

- a send window
- a receive window.

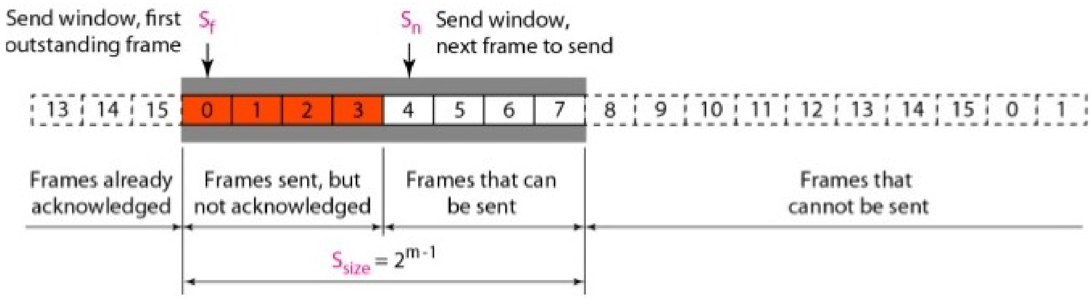
There are differences between the windows in this protocol and the ones in Go-Back-N.

First, the size of the send window is much smaller; it is 2^m- 1.

Second, the receive window is the same size as the send window. The send window maximum size can be 2^m- 1.

For eg: if m = 4, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the Go-Back-N Protocol). The smaller window size means less efficiency in filling the pipe, but the fact that there are fewer duplicate frames can compensate for this. The protocol uses the same variables as for Go-Back-N. We show the Selective Repeat send window in Figure 11.18 to emphasize the size. Compare it with Figure 11.12.

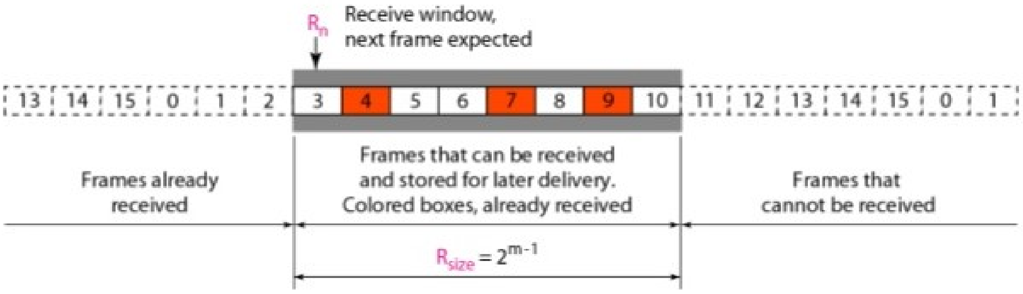
Figure 11.18 ***Send window for Selective Repeat ARQ***


Send window, first outstanding frame
S_f
S_n
Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent
$S_{size} = 2^{m-1}$

The receive window in Selective Repeat is totally different from the one in Go Back-N.

First, the size of the receive window is the same as the size of the send window (2^m- 1). The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out oforder and be stored until they can be delivered. We need, however, to mention that the receiver never delivers packets out of order to the network layer. Figure 11.19 shows the receive window in this protocol. Those slots inside the window that are colored define frames that have arrived out oforder and are waiting for their neighbors to arrive before delivery to the network layer.

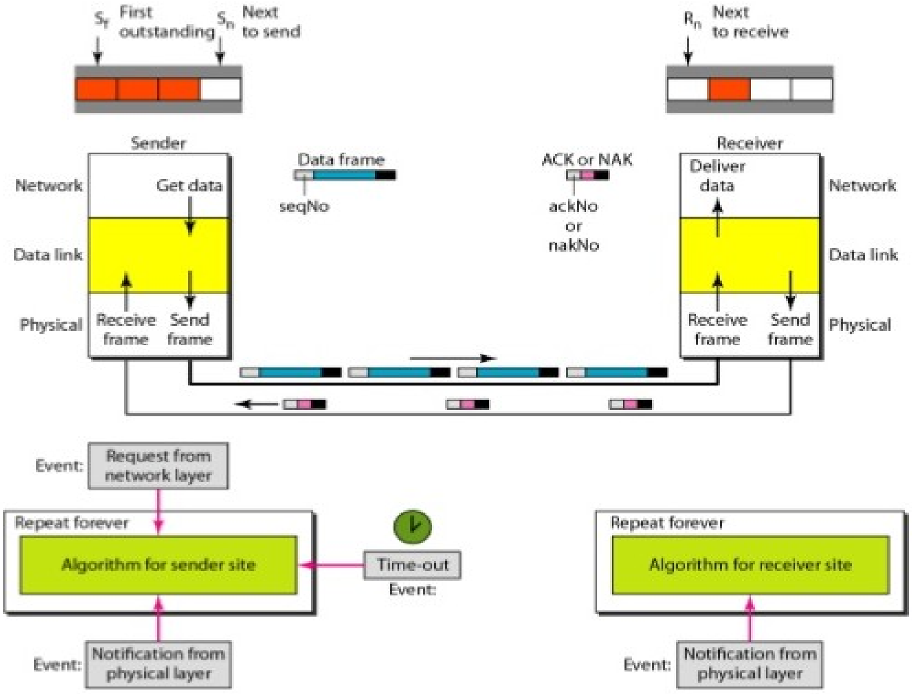
Figure 11.19 *Receive window for Selective Repeat ARQ*


R_n
Receive window,
next frame expected
13
14
15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
R_size = 2^m-1

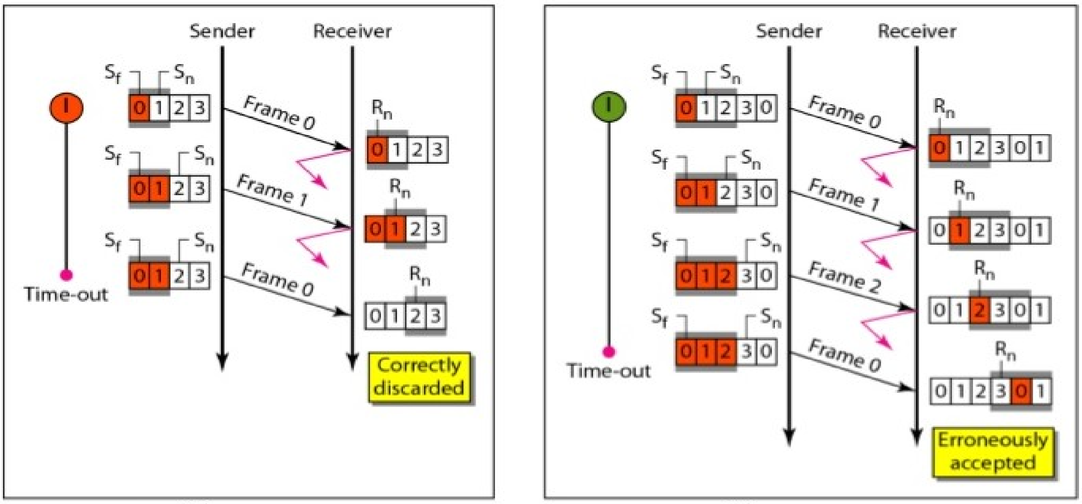
Design

The design in this case is to some extent similar to the one we described for the G0 Back-N, but more complicated, as shown in Figure 11.20.

Figure 11.20 *Design of Selective Repeat ARQ*



Window Sizes



a. Window size = 2^{m-1}

b. Window size > 2^{m-1}

In Selective Repeat ARQ, the size of the sender and receiver window must be atmost one-half of 2^m

Piggybacking

The three protocols in this section are all unidirectional: data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction.

In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. 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.

We show the design for a Go-Back-N ARQ using piggybacking in Figure 11.24.Each node now has two windows:

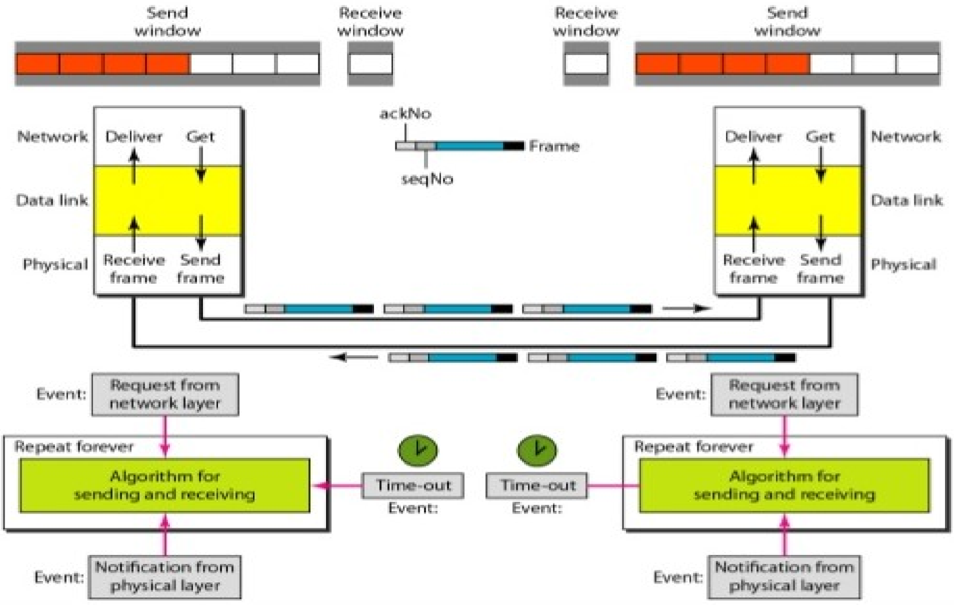
- one send window
- One receive window.

Both also need to use a timer. Both are involved in three types of events:

- request,
- arrival,
- Time-out.

However, the arrival event here is complicated; when a frame arrives, the site needs to handle control information as well as the frame itself. Both of these concerns must be taken care of in one event, the arrival event. The request event uses only the send window at each site; the arrival event needs to use both windows. An important point about piggybacking is that both sites must use the same algorithm. This algorithm is complicated because it needs to combine two arrival events into one.

Figure 11.24 *Design of piggybacking in Go-Back-N ARQ*

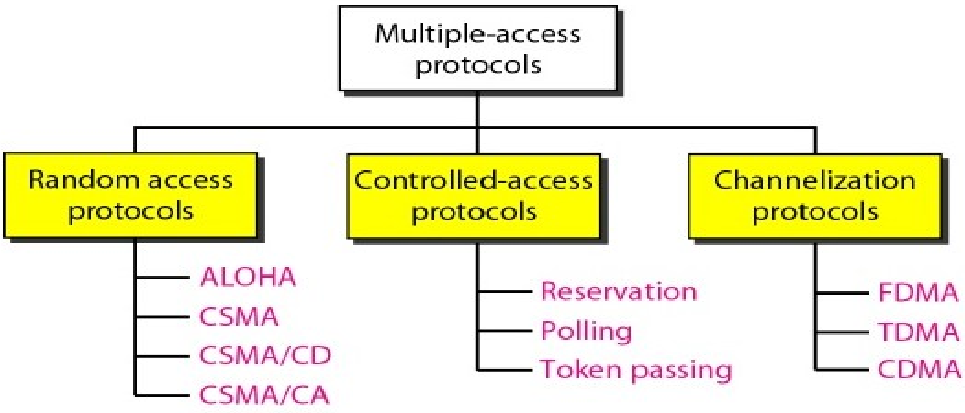


Unit 3

Multiple Access Protocols . Random Access-ALOHA, CSMA.

Multiple Access Protocols

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. Multiple access protocol is used to coordinate access to the link. Nodes can regulate their transmission onto the shared broadcast channel by using Multiple access protocol.



RANDOM ACCESS

In random access or contention methods, no station is superior to another station and none is assigned the control over another. No station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including the testing of the state of the medium

.

FEATURES.

1) There is no scheduled time for a station to transmit. Transmission is random among the stations. Transmission is random among the stations. That is why these methods are called *random access.*

2) No rules specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

In a random access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict-collision-and the frames will be either destroyed or modified.

The random access methods have evolved from a protocol known as ALOHA, which used a very simple procedure called multiple access (MA). The method was improved with the

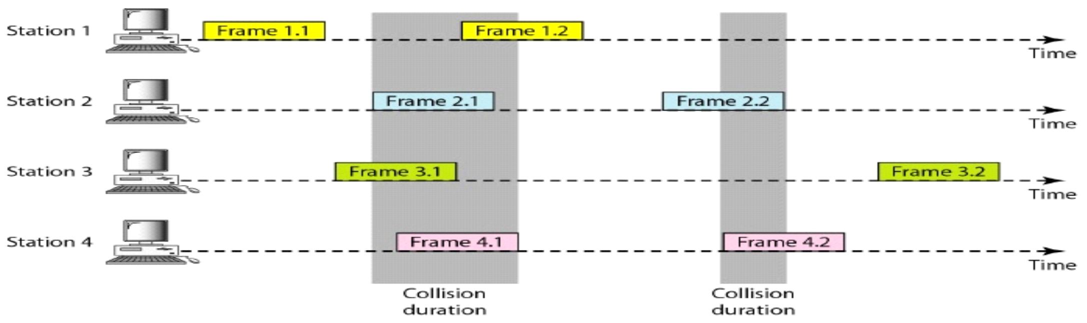
addition of a procedure that forces the station to sense the medium before transmitting. This was called carrier sense multiple access. This method later evolved into two parallel methods:
1) carrier sense multiple access with collision detection (CSMA/CD)
2) Carrier sense multiple access with collision avoidance *(CSMA/CA).*
CSMA/CD tells the station what to do when a collision is detected.
CSMA/CA tries to avoid the collision

ALOHA

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. However, since there is only one channel to share, there is the possibility of collision between frames from different stations.

Figure below shows an example of frame collisions in pure ALOHA.

Figure 12.3 *Frames in a pure ALOHA network*



12.5

There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel.

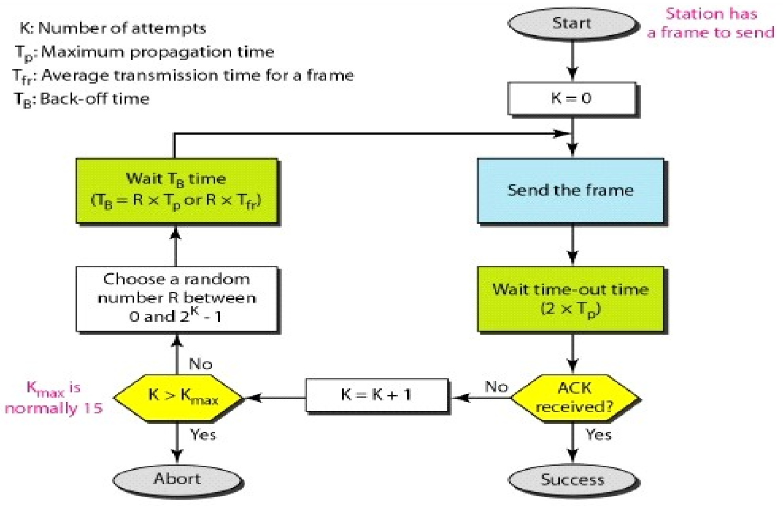
Fig shows that **two frames survive**: frame 1.1 from station 1 and frame 3.2 from station 3. It is obvious that we need to resend the frames that have been destroyed during transmission. The pure ALOHA protocol relies on acknowledgments from the receiver.

When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the 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. This time is called **Back off time T_B.**

Pure ALOHA has a **second method** to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts K_{max}' a station must give up and try later. Figure below shows the procedure for pure ALOHA based on the above strategy.

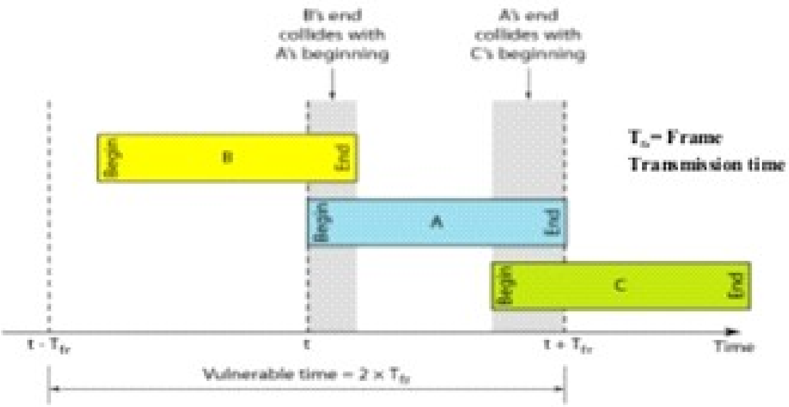
Figure 12.4 *Procedure for pure ALOHA protocol*


K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Start
Station has a frame to send
K = 0
Send the frame
Wait time-out time ($2 \times T_p$)
ACK received?
No
Yes
Success
K = K + 1
$K > K_{max}$
K_{max} is normally 15
Yes
Abort
No
Choose a random number R between 0 and $2^K - 1$
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)

12.6

Vulnerable time

Vulnerable time is the length of time, in which there is a possibility of collision. Assume that the stations send fixed-length frames with each frame taking *Tfr* s to send. Figure below shows the vulnerable time for station A.


B's end collides with A's beginning
A's end collides with C's beginning
T_{fr} – Frame Transmission time
Begin
B
End
Begin
A
End
Begin
C
End
$t - T_{fr}$
t
$t + T_{fr}$
Time
Vulnerable time = $2 \times T_{fr}$

Vulnerable time for pure ALOHA protocol

Slotted ALOHA

Pure ALOHA has a vulnerable time of 2 x T_{fr} . This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or soon before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure 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. Figure shows an example of frame collisions in slotted ALOHA.

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 which 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. However, the vulnerable time is now reduced to one-half, equal to *Tfr*

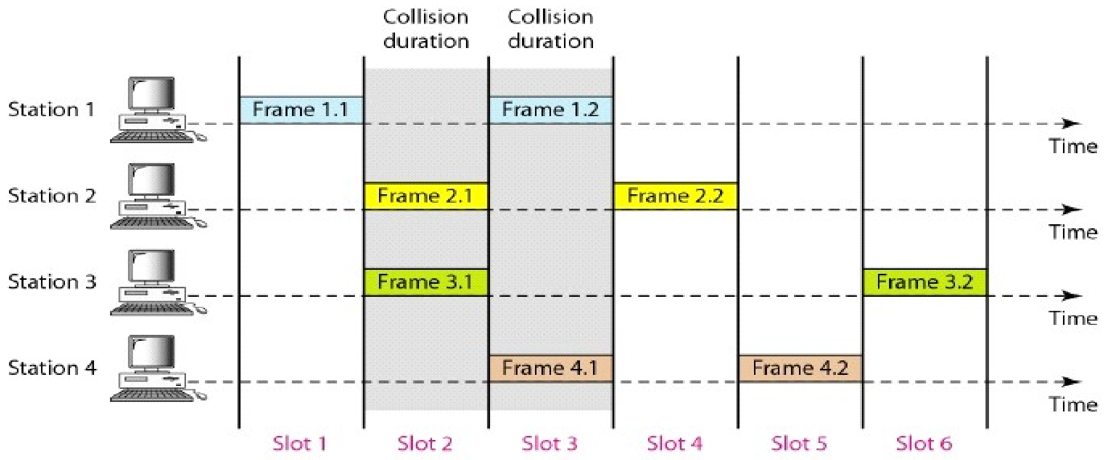


Figure: *Frames in a slotted ALOHA network*

Slotted ALOHA vulnerable time = Tfr

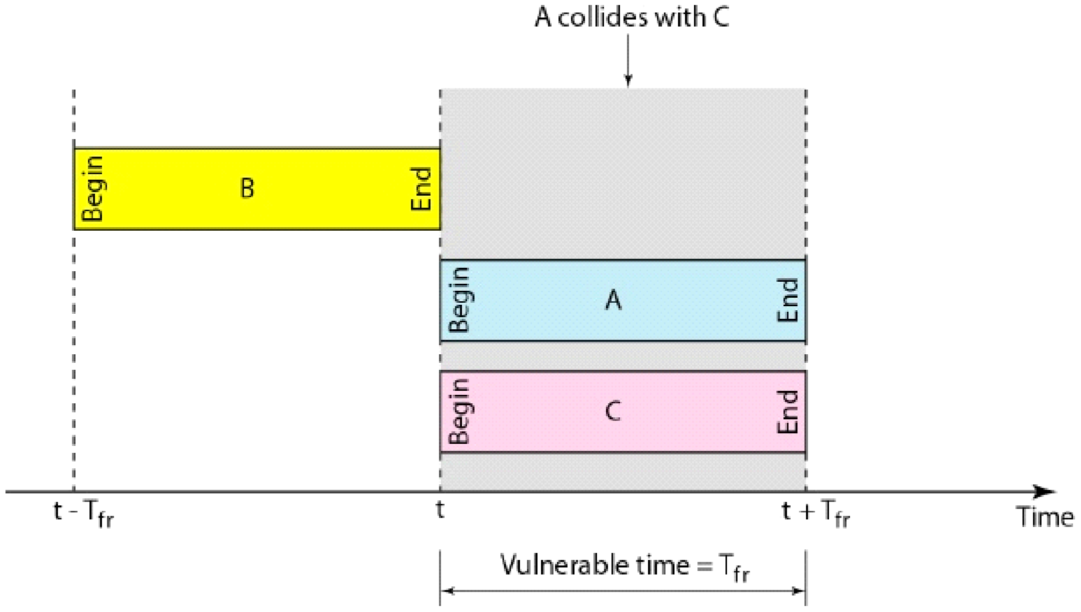


Figure: Vulnerable time for slotted ALOHA protocol

Carrier Sense Multiple Access (CSMA)

To minimize the chance of collision and to increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. 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."

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium).

B starts at

C starts at
time t2

time t1

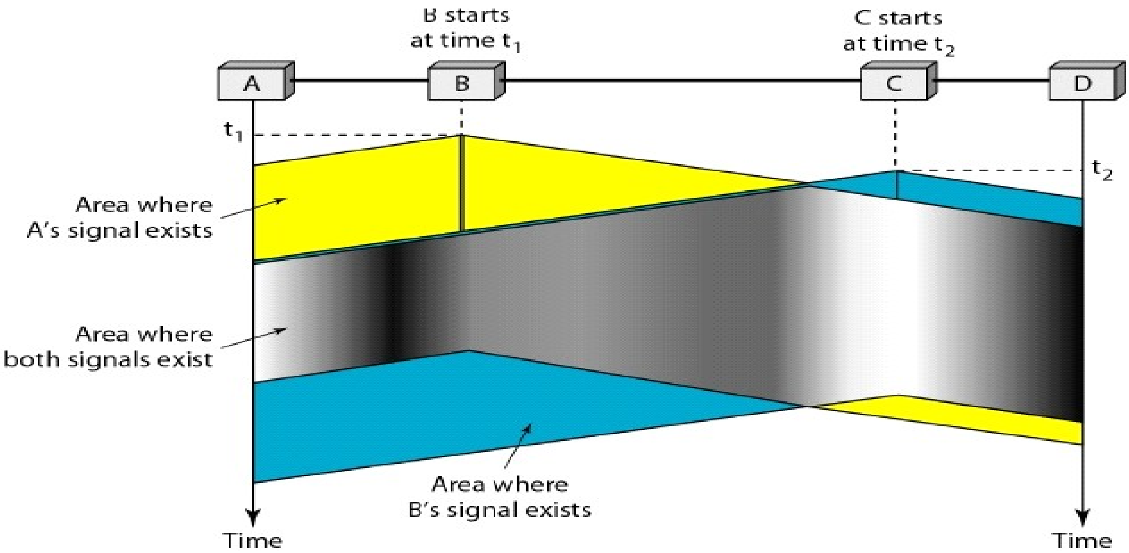


Figure:Space/time model of collision in CSMA

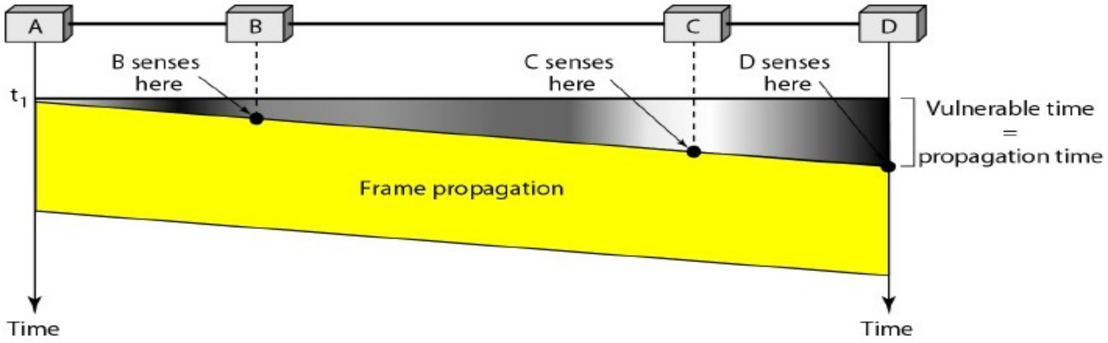
The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

At time *t1* station B senses the medium and finds it idle, so it sends a frame. At time *t2 (t2> tI)* 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 . This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending.

Figure 12.9 *Vulnerable time in CSMA*



The leftmost station A sends a frame at time tl' which reaches the rightmost station D at time tl + Tp. The gray area shows the vulnerable area in time and space.

Persistence Methods

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 to answer these questions:

- the I-persistent method
- the non persistent method
- the p-persistent method.

Figure 12.10 shows the behavior of three persistence methods when a station finds a channel busy.

I-Persistent

The I-persistent method is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately. This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

Non persistent

In the non persistent method, a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The non persistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.

p-Persistent

The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

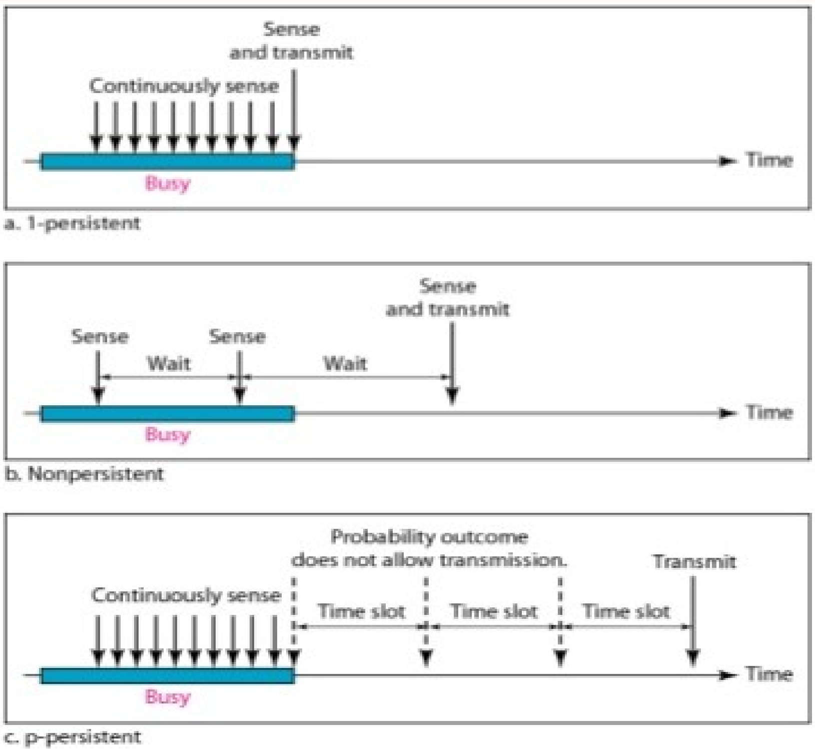
1. With probability p, the station sends its frame.

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 back off procedure.

Figure 12.10 *Behavior of three persistence methods*



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

The CSMA method does not specify the procedure following a collision.Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision.

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. To understand CSMA/CD, consider the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure 12.12, stations A and C are involved in the collision.

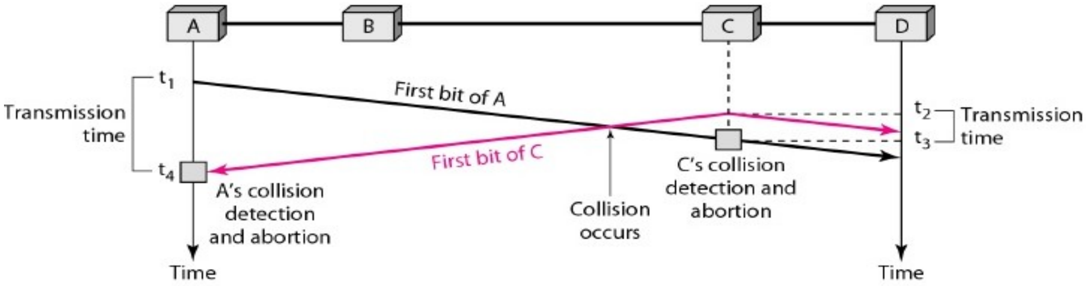
At time t1, station A has executed its persistence procedure and starts sending the bits of its frame.

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. The collision occurs sometime after time t2'.

Station C detects a collision at time t3 when it receives the first bit of A's frame. Station C immediately (or after a short time) aborts transmission.

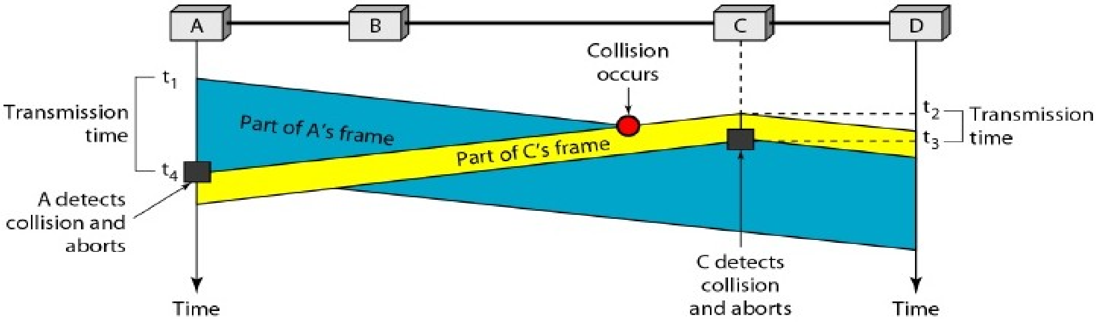
Station A detects collision at time t4 when it receives the first bit of C's frame; it also immediately aborts transmission.

Figure 12.12 *Collision of the first bit in CSMA/CD*



Looking at the figure, we see that A transmits for the duration t4 - tl; C transmits for the duration t3 - t2'. For the protocol to work, the length of any frame divided by the bit rate in this protocol must be more than either of these durations. At time t4, the transmission of A's frame, though incomplete, is aborted; at time t3, the transmission of B's frame, though incomplete, is aborted.

Figure 12.13 *Collision and abortion in CSMA/CD*



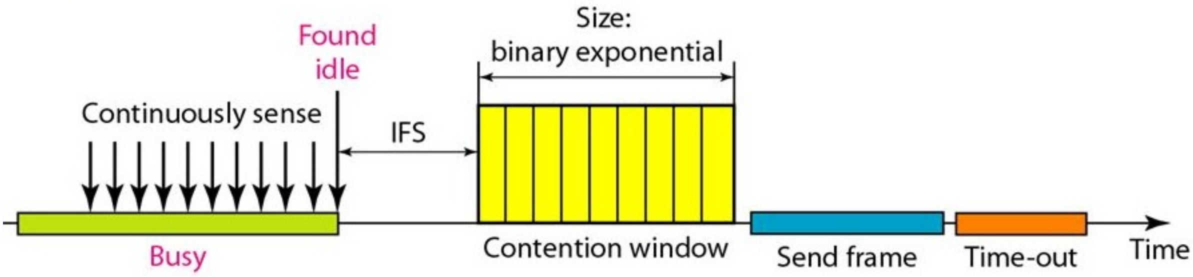
Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

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 one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station. To distinguish between these two cases, the received signals in these two cases must be significantly different. In other words, the signal from the second station needs to add a significant amount of energy to the one created by the first station.

In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles. However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection. We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for this network. Collisions are avoided through the use of CSMA/CA's 3 strategies:

- the interframe space
- the contention window
- acknowledgments as shown in Figure 12.16.

Figure 12.16 *Timing in CSMA/CA*



Interframe Space (IFS)

First, 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. Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the **contention time**. The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned a **shorter IFS has a higher priority.**

In CSMA/CA, the IFS can also be used to define the priority of a station or a frame.

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.This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the p-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.

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 acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

UNIT 3

Wired LANs-IEEE standards, wireless LANs-Bluetooth, cellular telephony

A local area network (LAN) is a computer network that is designed for a limited geographic area such as a building or a campus. Most LANs today are also linked to a wide area network (WAN) or the Internet.
The LAN market has seen several technologies such as Ethernet, Token Ring, Token Bus, FDDI, and ATM LAN. Some of these technologies survived for a while, but Ethernet is by far the dominant technology.
The **IEEE Standard Project 802** , designed to regulate the manufacturing and interconnectivity between different LANs.

IEEE STANDARDS

In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers. It is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.
The standard was adopted by the American National Standards Institute (ANSI). In 1987, the International Organization for Standardization (ISO) also approved it as an international standard under the designation ISO 8802.

The IEEE has subdivided the **data link** layer into two sub layers:

- logical link control (LLC)
- Media access control (MAC).

IEEE has also created several physical layer standards for different LAN protocols.

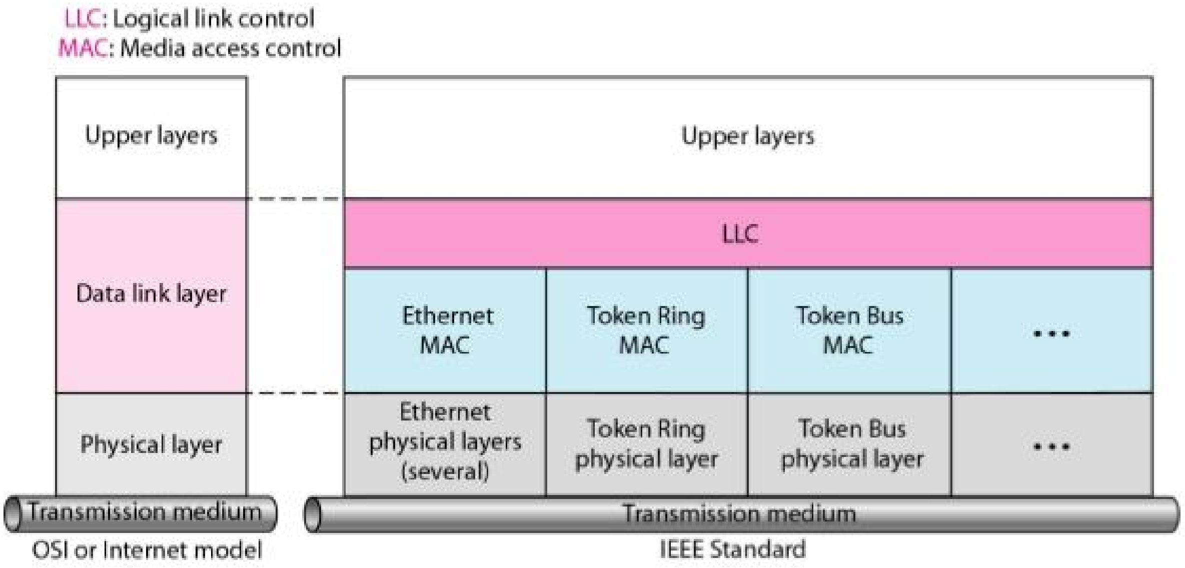


Figure: IEEE standard for LANs

Logical link control (LLC)

Data link layer controls handles framing, flow control, and error control. In IEEE Project 802, **flow control, error control, and part of the framing duties** are collected into one sub layer called the **logical link control**.

Framing is handled in **both** the LLC sublayer and the MAC sublayer. The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer, which provides different protocols for different LANs. A single LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent.

Framing

LLC defines a protocol data unit (PDU) that is somewhat similar to that of HDLC. The header contains a control field like the one in HDLC; this field is used for flow and error control. The two other header fields define the upper-layer protocol at the source and destination that uses LLC. These fields are called the destination service access point (DSAP) and the source service access point (SSAP). The other fields defined in a typical data link control protocol such as HDLC are moved to the MAC sublayer. In other words, a frame defined in HDLC is divided into a PDU at the LLC sublayer and a frame at the MAC sublayer.

Need for LLC

The purpose of the LLC is to provide flow and error control for the upper-layer protocols. If a LAN or several LANs are used in an isolated system, LLC may be needed to provide flow and error control for the application layer protocols. However, most upper-layer protocols such as IP do not use the services of LLC.

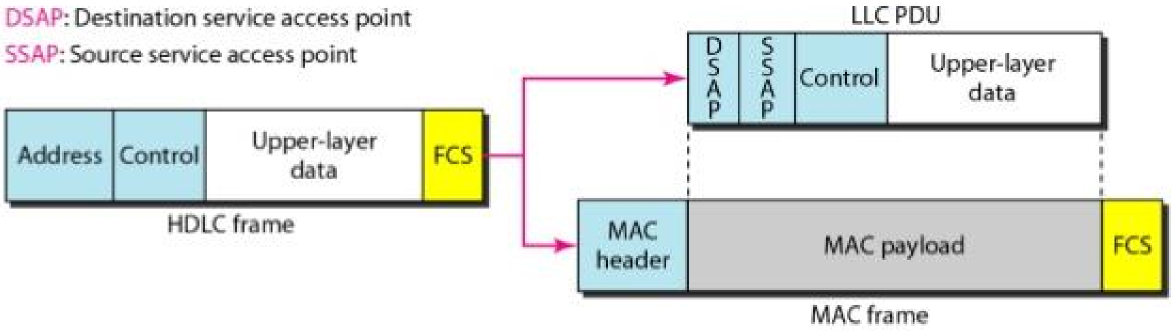


Figure: *HDLC frame compared with LLC and MAC frames*

Media Access Control (MAC)

IEEE Project 802 has created a sublayer called media access control that defines the specific access method for each LAN. For example, it defines *CSMA/CD* as the media access method for Ethernet LANs and the token passing method for Token Ring and Token Bus LANs.
In contrast to the LLC sublayer, the MAC sublayer contains a number of distinct modules; each defines the access method and the framing format specific to the corresponding LAN protocol.

Physical Layer

The physical layer is dependent on the implementation and type of physical media used. IEEE defines detailed specifications for each LAN implementation. For example, although there is only one MAC sublayer for Standard Ethernet, there is a different physical layer specifications for each Ethernet implementations.

Bluetooth

Bluetooth is a wireless LAN technology designed to connect devices of different functions such as telephones, notebooks, computers (desktop and laptop), cameras, printers, coffee makers, and so on.

A Bluetooth LAN is an ad hoc network, which means that the network is formed spontaneously; the devices, sometimes **called gadgets**, find each other and make a network called a **piconet**.

A Bluetooth LAN can even be connected to the Internet if one of the gadgets has this capability. A Bluetooth LAN, by nature, cannot be large. If there are many gadgets that try to connect, there is chaos. Bluetooth technology has several applications. Peripheral devices such as a wireless mouse or keyboard can communicate with the computer through this technology. Monitoring devices can communicate with sensor devices in a small health care center. Home security devices can use this technology to connect different sensors to the main security controller. Conference attendees can synchronize their laptop computers at a conference.

Bluetooth technology is the implementation of a protocol defined by the IEEE 802.15 standard. The standard defines a wireless personal-area network (PAN) operable in an area the size of a room or a hall.

Architecture

Bluetooth defines two types of networks:

* **piconet**

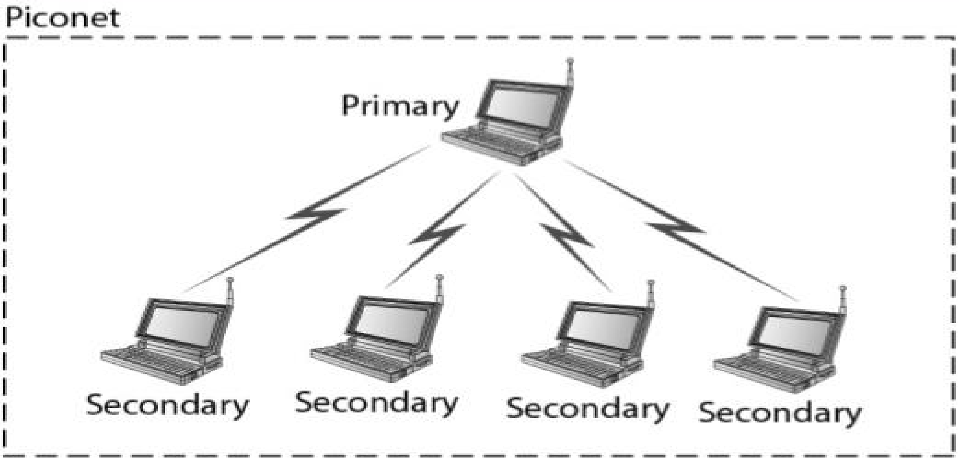
* **scatternet.**

Piconets

A Bluetooth network is called a **piconet,** or a small net. A piconet can have up to **eigh**t stations, one of which is called the primary;t the rest are called secondaries. All the secondary stations synchronize their clocks and hopping sequence with the primary. A piconet can have only one primary station. The communication between the primary and the secondary can be one-to-one or one-to-many. Figure 14.19 shows a piconet.

Although a piconet can have a maximum of seven secondaries, an additional eight secondaries can be in the ***parked state***. A secondary in a parked state is synchronized with the primary, but cannot take part in communication until it is moved from the parked state. Because only eight stations can be active in a piconet, activating a station from the parked state means that an active station must go to the parked state.

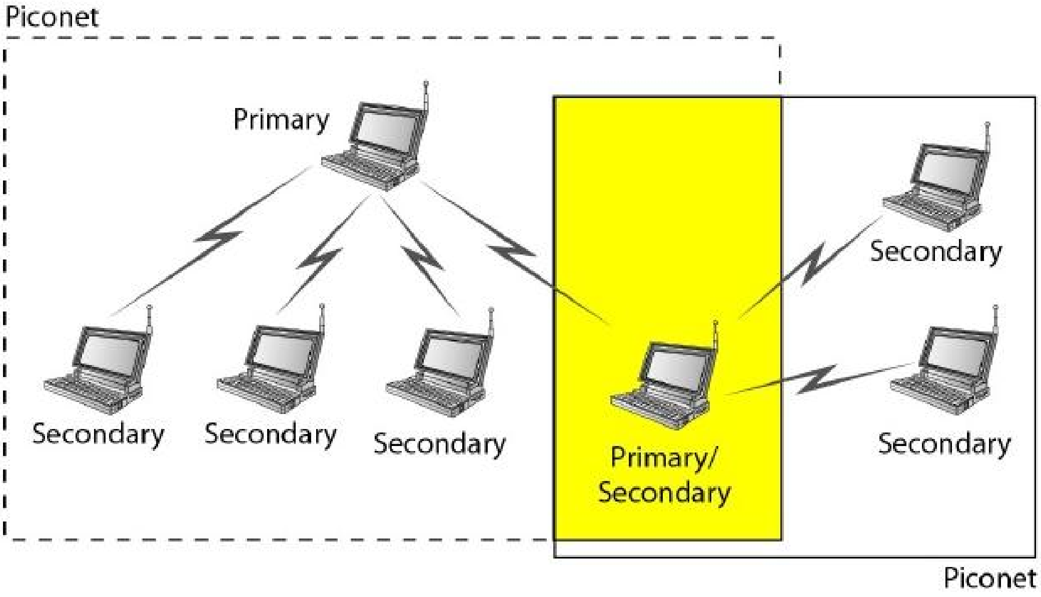
Figure 14.19 ***Piconet***



Scatternet

Piconets can be combined to form what is called a **scatternet**. A secondary station in one piconet can be the primary in another piconet. This station can receive messages from the primary in the first piconet (as a secondary) and, acting as a primary, deliver them to secondaries in the second piconet. A station can be a member of two piconets. Figure 14.20 illustrates a scatternet.

Figure 14.20 ***Scatternet***



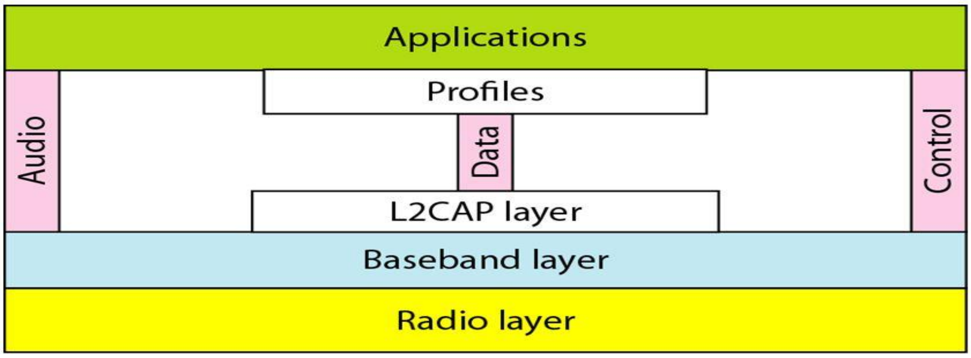
Bluetooth Devices

A Bluetooth device has a built-in short-range radio transmitter. The current data rate is 1Mbps with a 2.4-GHz bandwidth. This means that there is a possibility of interference between the IEEE 802.11b wireless LANs and Bluetooth LANs.

Bluetooth Layers

Bluetooth uses several layers that do not exactly match those of the Internet. Figure 14.21 shows these layers.

Figure 14.21 *Bluetooth layers*



Radio Layer

The radio layer is roughly equivalent to the physical layer of the Internet model. Bluetooth devices are low-power and have a range of 10 m.

Band

Bluetooth uses a 2.4-GHz ISM band divided into 79 channels of 1 MHz each.

FHSS

Bluetooth uses the frequency-hopping spread spectrum (FHSS) method in the physical layer to avoid interference from other devices or other networks. Bluetooth hops 1600 times per second, which means that each device changes its modulation frequency 1600 times per second.

Baseband Layer

The baseband layer is roughly equivalent to the MAC sublayer in LANs. The access method is TDMA. The primary and secondary communicate with each other using time slots. The length of a time slot is exactly the same as the dwell time, 625 s. This means that during the time that one frequency is used, a sender sends a frame to a secondary, or a secondary sends a frame to the primary. Note that the communication is only between the primary and a secondary; secondaries cannot communicate directly with one another.

TDMA

Bluetooth uses a form of TDMA that is called TDD-TDMA (time division duplex TDMA). TDD-TDMA is a kind of half-duplex communication in which the secondary and receiver send and receive data, but not at the same time (half duplex); however, the communication for each direction uses different hops. This is similar to walkie-talkies using different carrier frequencies.

Single-Secondary Communication: If the piconet has only one secondary, the TDMA operation is very simple. The time is divided into slots of 625 Ils. The primary uses even numbered slots (0, 2, 4, ...); the secondary uses odd-numbered slots (1, 3, 5, ...). TDD-TDMA allows the primary and the secondary to communicate in half-duplex mode.

Multiple-Secondary Communication : The process is a little more involved if there is more than one secondary in the piconet. Again, the primary uses the even-numbered slots, but a secondary sends in the next odd-numbered slot if the packet in the previous slot was addressed to it. All secondaries listen on even-numbered slots, but only one secondary sends in any odd-numbered slot.

Physical Links

Two types of links can be created between a primary and a secondary: SCQ links and ACL links.

SCO

A synchronous connection-oriented (SCO) link is used when avoiding latency (delay in data delivery) is more important than integrity (error-free delivery). In an SCQ link, a physical link is created between the primary and a secondary by reserving specific slots at regular intervals. The basic unit of connection is two slots, one for each direction. If a packet is damaged, it is never retransmitted. SCO is used for real-time audio where avoiding delay is all-important. A secondary can create up to three SCO links with the primary, sending digitized audio (PCM) at 64 kbps in each link.

ACL

An asynchronous connectionless link (ACL) is used when data integrity is more important than avoiding latency. In this type of link, if a payload encapsulated in the frame is corrupted, it is retransmitted. A secondary returns an ACL frame in the available odd-numbered slot if and only if the previous slot has been addressed to it. ACL can use one, three, or more slots and can achieve a maximum data rate of 721 kbps.

Frame Format

A frame in the baseband layer can be one of three types:

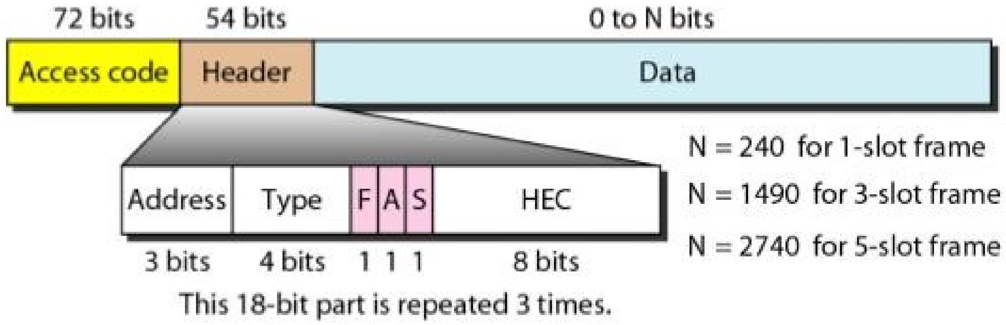
- one-slot,
- three-slot,
- five-slot.

In a **one-slot** frame exchange, 259 s is needed for hopping and control mechanisms. This means that a one-slot frame can last only 625 - 259, or 366 s. With a 1-MHz bandwidth and 1 bit/Hz, the size of a one-slot frame is 366 bits.

A **three-slot** frame occupies three slots. However, since 259 s is used for hopping, the length of the frame is 3 x 625 - 259 = 1616 s or 1616 bits. A device that uses a three-slot frame remains at the same hop (at the same carrier frequency) for three slots. Even though only one hop number is used, three hop numbers are consumed. That means the hop number for each frame is equal to the first slot of the frame.

A **five-slot** frame also uses 259 bits for hopping, which means that the length of the frame is 5 x 625 - 259 =2866 bits. Figure 14.24 shows the format of the three frame types. The following describes each field:

Figure 14.24 ***Frame format types***



- **Access code.** This 72-bit field normally contains synchronization bits and the identifier of the primary to distinguish the frame of one piconet from another.
- Header. This 54-bit field is a repeated I8-bit pattern. Each pattern has the following subfields:

1. Address. The 3-bit address subfield can define up to seven secondaries (l to 7). If the address is zero, it is used for broadcast communication from the primary to all secondaries.

2. Type. The 4-bit type subfield defines the type of data coming from the upper layers.

3. F. This I-bit subfield is for flow control. When set (I), it indicates that the device is unable to receive more frames (buffer is full).

4. A. This 1-bit subfield is for acknowledgment. Bluetooth uses Stop-and-Wait ARQ; 1 bit is sufficient for acknowledgment.

5. S. This 1-bit subfield holds a sequence number. Bluetooth uses Stop-and-Wait ARQ; I bit is sufficient for sequence numbering.

6. HEC. The 8-bit header error correction subfield is a checksum to detect errors in each 18-bit header section. The header has three identical 18-bit sections. The receiver compares these three sections, bit by bit. If each of the corresponding bits is the same, the bit is accepted; if not, the majority opinion rules. This is a form of forward error correction (for the header only). This double error control is needed because the nature of the communication, via air, is very noisy. There is no retransmission in this sublayer.

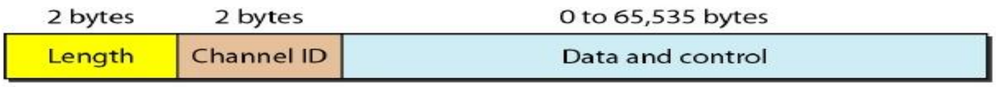
- **Payload**. This subfield can be 0 to 2740 bits long. It contains data or control information corning from the upper layers.

L2CAP

The Logical Link Control and Adaptation Protocol, or L2CAP (L2 here means LL), is roughly equivalent to the LLC sublayer in LANs. It is used for data exchange on an ACL link; SCQ channels do not use L2CAP. Figure 14.25 shows the format of the data packet at this level. The 16-bit length field defines the size of the data, in bytes, coming from the upper layers. Data can be up to 65,535 bytes. The channel ID (CID) defines a unique identifier for the virtual channel created at this level.

The L2CAP has specific duties: **multiplexing, segmentation and reassembly, quality of service (QoS), and group management.**

Figure 14.25 *L2CAP data packet format*



Multiplexing

The L2CAP can do multiplexing. At the sender site, it accepts data from one of the upper-layer protocols, frames them, and delivers them to the baseband layer. At the receiver site, it accepts a frame from the baseband layer, extracts the data, and delivers them to the appropriate protocol layer.

Segmentation and Reassembly

The maximum size of the payload field in the baseband layer is 2774 bits, or 343 bytes. This includes 4 bytes to define the packet and packet length. Therefore, the size of the packet that can arrive from an upper layer can only be 339 bytes. However, application layers sometimes need to send a data packet that can be up to 65,535 bytes (an Internet packet, for example). The L2CAP divides these large packets into segments and adds extra information to define the location of the segments in the original packet. The L2CAP segments the packet at the source and reassembles them at the destination.

QoS

Bluetooth allows the stations to define a quality-of-service level. Bluetooth defaults to what is called best-effort service; it will do its best under the circumstances.

Group Management

Another functionality of L2CAP is to allow devices to create a type of logical addressing between themselves. This is similar to multicasting. For example, two or three secondary devices can be part of a multicast group to receive data from the primary.

Other Upper Layers

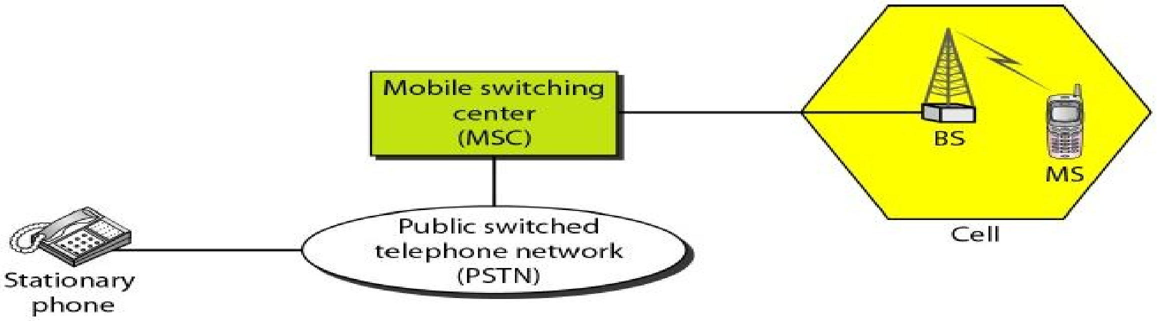
Bluetooth defines several protocols for the upper layers that use the services ofL2CAP; these protocols are specific for each purpose.

Cellular Telephony

Cellular telephony is designed to provide communications between two moving units, called **mobile stations** (MSs), or between one mobile unit and one stationary unit, often called a land unit. A service provider must be able to locate and track a caller, assign a channel to the call, and transfer the channel from base station to base station as the caller moves out of range.

To make this tracking possible, each cellular service area is divided into small regions called **cells**. Each cell contains an antenna and is controlled by a solar or AC powered network station, called the **base station** (BS). Each base station, in turn, is controlled by a switching office, called **a mobile switching center** (MSC). The MSC coordinates communication between all the base stations and the telephone central office. It is a computerized center that is responsible for connecting calls, recording call information, and billing (see Figure 16.1).

Figure 16.1 *Cellular system*

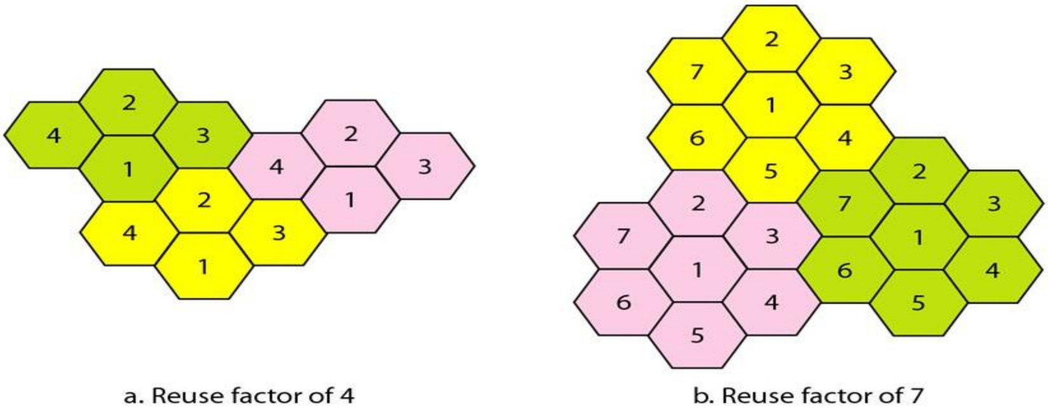

Mobile switching center (MSC)
BS
MS
Cell
Stationary phone
Public switched telephone network (PSTN)

Cell size is not fixed and can be increased or decreased depending on the population of the area. The typical radius of a cell is 1 to 12 mi. High-density areas require more, geographically smaller cells to meet traffic demands than do low-density areas. Once determined, cell size is optimized to prevent the interference of adjacent cell signals. The transmission power of each cell is kept low to prevent its signal from interfering with those of other cells.

Frequency-Reuse Principle

Neighbouring cells cannot use the same set of frequencies for communication because it may create interference for the users located near the cell boundaries. However, the set of frequencies available is limited, and frequencies need to be reused. A frequency reuse pattern is a configuration of N cells, N being the reuse factor, in which each cell uses a unique set of frequencies. When the pattern is repeated, the frequencies can be reused. There are several different patterns. Figure 16.2 shows two of them.

Figure 16.2 *Frequency reuse patterns*



The numbers in the cells define the pattern. The cells with the same number in a pattern can use the same set of frequencies. We call these cells the **reusing cells**. As Figure 16.2 shows, in a pattern with reuse factor 4, only one cell separates the cells using the same set of frequencies. In the pattern with reuse factor 7, two cells separate the reusing cells.

Transmitting

To place a call from a mobile station, the caller enters a code of 7 or 10 digits (a phone number) and presses the send button. The mobile station then scans the band, seeking a setup channel with a strong signal, and sends the data (phone number) to the closest base station using that channel. The base station relays the data to the MSC. The MSC sends the data on to the telephone central office. If the called party is available, a connection is made and the result is relayed back to the MSC. At this point, the MSC assigns an unused voice channel to the call, and a connection is established. The mobile station automatically adjusts its tuning to the new channel, and communication can begin.

Receiving

When a mobile phone is called, the telephone central office sends the number to the MSC. The MSC searches for the location of the mobile station by sending query signals to each cell in a process called **paging**. Once the mobile station is found, the MSC transmits a ringing signal and, when the mobile station answers, assigns a voice channel to the call, allowing voice communication to begin.

Handoff

It may happen that, during a conversation, the mobile station moves from one cell to another. When it does, the signal may become weak. To solve this problem, the MSC monitors the level of the signal every few seconds. If the strength of the signal diminishes, the MSC seeks a new cell that can better accommodate the communication. The MSC then changes the channel carrying the call (hands the signal off from the old channel to a new one).

Hard Handoff: Early systems used a hard handoff. In a hard handoff, a mobile station only communicates with one base station. When the MS moves from one cell to another, communication must first be broken with the previous base station before communication can be established with the new one. This may create a rough transition.

Soft Handoff : New systems use a soft handoff. In this case, a mobile station can communicate with two base stations at the same time. This means that, during handoff, a mobile station may continue with the new base station before breaking off from the old one.

Roaming

One feature of cellular telephony is called **roaming**. Roaming means that a user can have access to communication or can be reached where there is coverage. A service provider usually has limited coverage. Neighbouring service providers can provide extended coverage through a roaming contract. The situation is similar to snail mail between countries. The charge for delivery of a letter between two countries can be divided upon agreement by the two countries.

First Generation

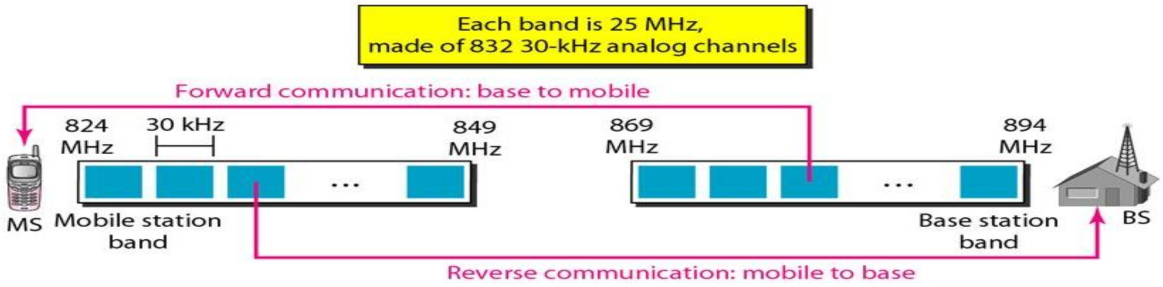
Cellular telephony is now in its second generation with the third on the horizon. The first generation was designed for voice communication using analog signals. We discuss one first-generation mobile system used in North America, AMPS

AMPS

Advanced Mobile Phone System (AMPS) is one of the leading analog cellular systems in North America. It uses FDMA to separate channels in a link.

Bands AMPS operates in the ISM 800-MHz band. The system uses two separate analog channels, one for forward (base station to mobile station) communication and one for reverse (mobile station to base station) communication. The band between 824 and 849 MHz carries reverse communication; the band between 869 and 894 MHz carries forward communication (see Figure 16.3).

Figure 16.3 *Cellular bands for AMPS*


Each band is 25 MHz, made of 832 30-kHz analog channels
Forward communication: base to mobile
824 MHz
30 kHz
849 MHz
869 MHz
894 MHz
...
...
MS
Mobile station band
Base station band
BS
Reverse communication: mobile to base

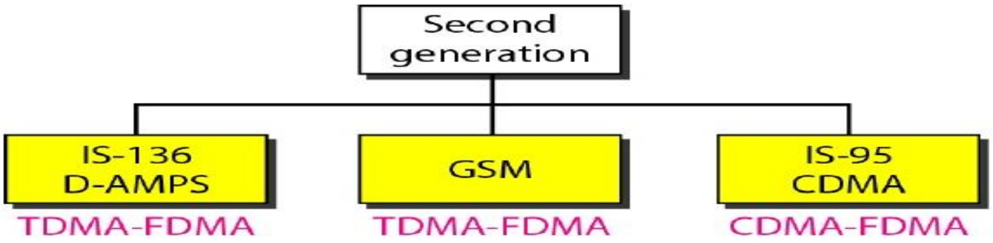
Each band is divided into 832 channels. However, two providers can share an area, which means 416 channels in each cell for each provider. Out of these 416, 21 channels are used for control, which leaves 395 channels. AMPS has a **frequency reuse factor of 7** ; this means only one-seventh of these 395 traffic channels are actually available in a cell.

Second Generation

To provide higher-quality (less noise-prone) mobile voice communications, the second generation of the cellular phone network was developed. While **the first generation** was designed for **analog voice communication**, the **second generation** was mainly designed for **digitized voice**.

Three major systems evolved in the second generation, as shown in Figure 16.5.

Figure 16.5 *Second-generation cellular phone systems*



D-AMPS

The product of the evolution of the analog AMPS into a digital system is digital AMPS (D-AMPS). D-AMPS was designed to be backward-compatible with AMPS. This means that in a cell, one telephone can use AMPS and another D-AMPS. D-AMPS was first defined by IS-54 (Interim Standard 54) and later revised by IS-136.

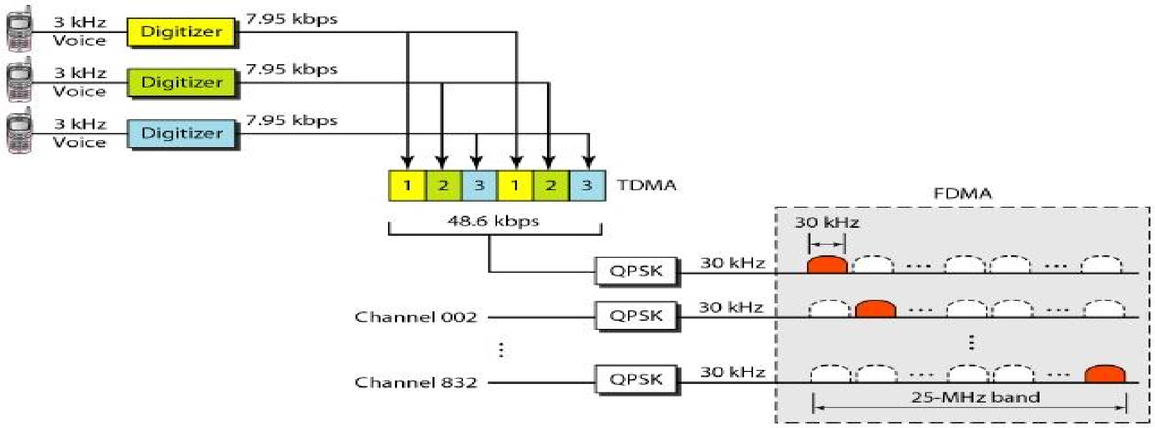
Band: D-AMPS uses the same bands and channels as AMPS.

Transmission: Each voice channel is digitized using a very complex PCM and compression technique. A voice channel is digitized to 7.95 kbps. Three 7.95-kbps digital voice channels are combined using TDMA. The result is 48.6 kbps of digital data; much of this is overhead.

As Figure 16.6 shows, the system sends 25 frames per second, with 1944 bits per frame. Each frame lasts 40 ms (1/25) and is divided into six slots shared by three digital channels; each channel is allotted two slots. Each slot holds 324 bits. However, only 159 bits comes from the digitized voice; 64 bits are for control and 101 bits are for error correction. In other words, each channel drops 159 bits of data into each of the two channels assigned to it. The system adds 64 control bits and 101 error-correcting bits.

The resulting 48.6 kbps of digital data modulates a carrier using QPSK; the result is a 3D-kHz analog signal. Finally, the 3D-kHz analog signals share a 25-MHz band (FDMA). D-AMPS has a frequency reuse factor of 7.

Figure 16.6 *D-AMPS*

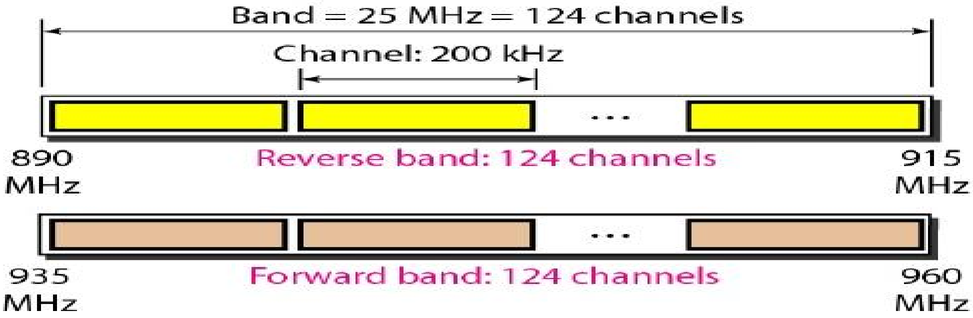


GSM

The Global System for Mobile Communication (GSM) is a European standard that was developed to provide a common second-generation technology for all Europe. The aim was to replace a number of incompatible first-generation technologies.

Bands GSM uses two bands for duplex communication. Each band is 25 MHz in width, shifted toward 900 MHz, as shown in Figure 16.7. Each band is divided into 124 channels of 200 kHz separated by guard bands.

Figure 16.7 *GSM bands*


Band = 25 MHz = 124 channels
Channel: 200 kHz
890 MHz
Reverse band: 124 channels
915 MHz
935 MHz
Forward band: 124 channels
960 MHz

Transmission: Figure 16.8 shows a GSM system. Each voice channel is digitized and compressed to a 13-kbps digital signal. Each slot carries 156.25 bits (see Figure 16.9). Eight slots share a frame (TDMA). Twenty-six frames also share a multiframe (TDMA). We can calculate the bit rate of each channel as follows:

Channel data rate = (11120 IDS) x 20 X 8 X 156.25=270.8 kbps

IS-95

One of the dominant second-generation standards in North America is Interim Standard 95 (IS-95). It is based on CDMA and DSSS.

Bands and Channels IS-95 uses two bands for duplex communication. The bands can be the traditional ISM 800-MHz band or the ISM 1900-MHz band. Each band is divided into 20 channels of 1.228 MHz separated by guard bands. Each service provider is allotted 10 channels. IS-95 can be used in parallel with AMPS. Each IS-95 channel is equivalent to 41 AMPS channels (41 x 30 kHz;:::: 1.23 MHz).
Synchronization All base channels need to be synchronized to use CDMA. To provide synchronization, bases use the services of GPS (Global Positioning System), a satellite system.
Forward Transmission IS-95 has two different transmission techniques: one for use in the forward (base to mobile) direction and another for use in the reverse (mobile to base) direction. In the forward direction, communications between the base and all mobiles are synchronized; the base sends synchronized data to all mobiles.
Reverse **Transmission** The use of CDMA in the forward direction is possible because the pilot channel sends a continuous sequence of Is to synchronize transmission. The synchronization is not used in the reverse direction because we need an entity to do that, which is not feasible. Instead of CDMA, the reverse channels use DSSS (direct sequence spread spectrum)

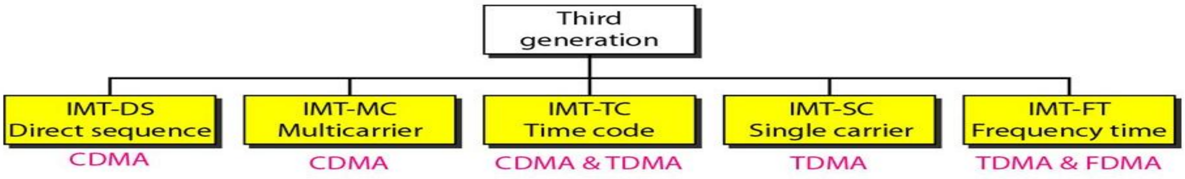
Third Generation

The third generation of cellular telephony refers to a combination of technologies that provide a variety of services. Ideally, when it matures, the third generation can provide both digital data and voice communication. Using a small portable device, a person should be able to talk to anyone else in the world with a voice quality similar to that of the existing fixed telephone network. A person can download and watch a movie, can download and listen to music, can surf the Internet or play games, can have a video conference, and can do much more. One of the interesting characteristics of a third generation system is that the portable device is always connected; you do not need to dial a number to connect to the Internet.
The third-generation concept started in 1992, when ITU issued a blueprint called the Internet Mobile Communication 2000 (IMT-2000). The blueprint defines some criteria for third-generation technology as outlined below:

- Voice quality comparable to that of the existing public telephone network.
- Data rate of 144 kbps for access in a moving vehicle (car), 384 kbps for access as the user walks (pedestrians), and 2 Mbps for the stationary user (office or home).
- Support for packet-switched and circuit-switched data services.
- A band of 2 GHz.
- Bandwidths of 2 MHz.
- Interface to the Internet

The main goal of third-generation cellular telephony is to provide universal personal communication.

Figure 16.12 *IMT-2000 radio interfaces*



IMT-DS This approach uses a version of COMA called wideband COMA or W-COMA.

W-COMA uses a 5-MHz bandwidth. It was developed in Europe, and it is compatible with the COMA used in IS-95.

IMT-MC This approach was developed in North America and is known as COMA 2000. It is an evolution of COMA technology used in IS-95 channels. It combines the new wideband (I5-MHz) spread spectrum with the narrowband (l.25-MHz) COMA of

IS-95. It is backward-compatible with IS-95. It allows communication on multiple

I.25-MHz channels (I, 3, 6, 9, 12 times), up to 15 MHz. The use of the wider channels allows it to reach the 2-Mbps data rate defined for the third generation.

IMT-TC This standard uses a combination of W-COMA and TDMA. The standard tries to reach the IMT-2000 goals by adding TOMA multiplexing to W-COMA.

IMT-SC This standard only uses TOMA.

IMT-FT This standard uses a combination of FDMA and TOMA.