problems at the destination. This is called *FEC (forward error correction)*. The receiver is responsible for fixing errors, if possible.

- **Error detection strategy :** Send only enough extra information to detect an error, and then request a retransmission from the source. This is called *ARQ (automatic repeat request)*. In this case, both the sender and receiver may take part in retransmitting lost information. It uses ACKs and retransmissions to recover from the errors.

First we develop some terminology and mathematics needed to describe errors and then we give several methods of error detection and correction.

(a) **Hamming distance**—*H*(*a*, *b*) = Number of bit positions in which 'a' differs from 'b', i.e., the number of 1's in a XOR b.

(b) **Forward error correction**—Enough redundancy is transmitted in the code that errors can be corrected by the receiver without retransmission (used when retransmission is impractical, as in distant space probes, simplex lines, broadcast media where many stations may have to listen to unnecessary retransmission).

(c) **Backward error correction**—Errors are detected and retransmission requested. (Most common, generally more efficient, but has the flaw that a single error, repeated in every retransmission, will mean that the message is never delivered).

9.3.1 Error Detection

Three possibilities in communication are:

- Frame is ok.
- Frame is bad and undetected.
- Frame is bad and detected.

Here the middle possibility should concern us.

Error Detection simply means that the receiver detects the fact that **some** errors exist in a received block.

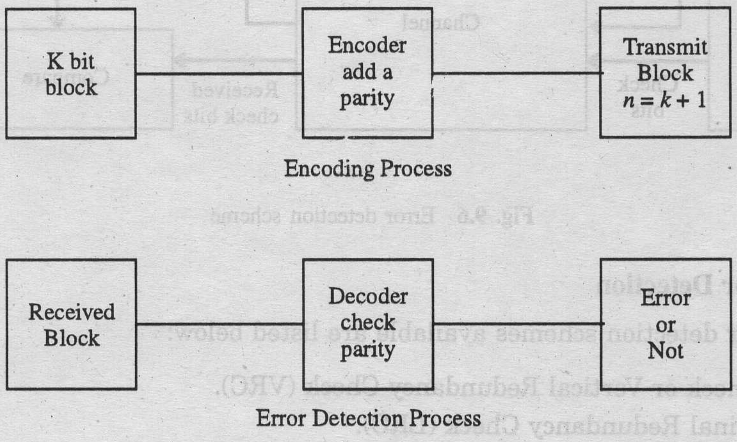


Fig. 9.4 Simple error detection scheme

This is typically achieved by adding redundant bits in each frame during its transmission. Based on all the bits in the frame (data + error check bits), the receiver may be able to detect errors in the frame. Then the sender retransmits that frame.

Principle of Error Detection

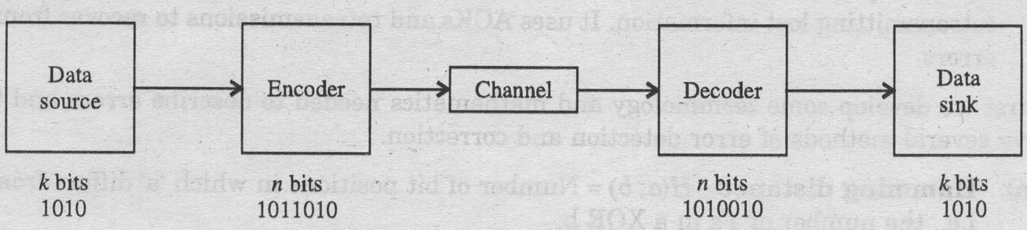


Fig. 9.5 Principle of error detection

- **Message (or Data) Source :** Source for information in bits (*k*-bits).
- **Encoding Process :** Process of converting a block of *k*-bit information to *n*-bit codeword, $n = k$.
- **Channel :** Medium in which *n*-bit codewords pass through.
- **Decoding Process :** Process of converting n bit received block to a *k*-bit message.
- **Message sink :** Destination for *k*-bit information in bits.

This principle is implemented in the following figure :

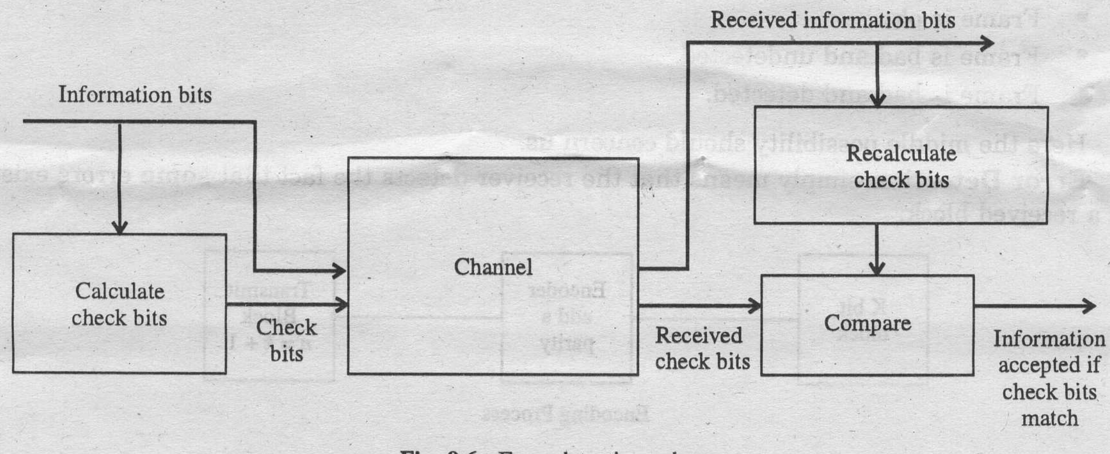


Fig. 9.6 Error detection scheme

Methods of Error Detection

The various error detection schemes available are listed below:

- Parity Check or Vertical Redundancy Check (VRC).
- Longitudinal Redundancy Check (LRC).

- Cyclical Redundancy Check (CRC).
- Checksum.

In this chapter we will discuss just the first two techniques.

1. Parity Check

This is the simplest error-detection mechanism. A parity bit (Redundant bit) is appended to a block of data, normally at the end of a 7-bit ASCII (American Standard Code for Information Interchange) character. Two techniques-even parity or odd parity-are available, and which method is used is up to the user. In **even parity**, a parity bit is selected so that the character has an even number of 1's. In odd parity, the parity bit is selected so that the character has an odd number of 1's.

For example, if even parity is selected and a computer receives a character with an odd number of 1's, it assumes an error and asks for a retransmission. Consider the following case:

Data to be sent : 1 1 0 1 1 0 1 1

Assume even parity generator : As the data contains even number of 1's so the parity bit is 0.

Block transmitted : Data bits 1 1 0 1 1 0 1 1 + Parity bit $\underline{0}$ → 1 1 0 1 1 0 1 1 $\underline{0}$
Data received by the receiver : 1 1 0 0 1 0 1 1 $\underline{0}$

The receiver counts the 1's and comes with odd number of 1s but there should be even number of 1,s so the receiver detects the corrupted data and asks for retransmission.

For small, non-critical data transmissions, this method is a reasonable tradeoff between reliability and efficiency. But it presents problems in cases where highly reliable data must be transmitted.

Limitation : The primary problem with using a single parity bit is that it cannot detect the presence of more than one transmission error. If two bits are incorrect, the parity can still be even and no error can be detected. So this method easily breaks down.

2. LRC

Longitudinal Redundancy Check scheme uses not only one parity bit per word (or row of the frame, considered now as a matrix), but also a "parity check character", comprising the entire last row of the matrix, with each bit in the row checking the parity of the corresponding column. The column parity bits form the last row and are called the LRC or the parity check character.

Consider the following example:

- There is a block of 16-bits to be transmitted:

  Data bits : 1 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0

- Organize it in a table of two rows and eight columns:

  1 0 1 1 0 1 0 0
  1 1 0 1 1 1 1 0

- Calculate the parity bit for each column and create a new row of 8-bits that are the redundant or parity bits to be transmitted along with data.

```
               10110100
               11011110
               --------
LRC  →         01101010    (Even Parity considered)
```

- Therefore, the data to be transmitted is Data bits + LRC

  1 0 1 1 0 1 0 0   1 1 0 1 1 1 1 0 + 01101010

- When the receiver receives the data it checks the LRC again and if some of the bits do not follow the even parity rule then the whole block is discarded.

Limitation : LRC will fail to detect errors that occur in an "even rectangular form", and other forms harder to describe as long as there are an even number of errors in each column and each row.

9.3.2 Error Correction

In the previous section we discussed error-detecting schemes. However, that kind of redundancy doesn't allow for the **correction** of the error. Error-correcting codes do exactly this: they add redundancy to the original message in such a way that it is possible for the receiver to detect the error and correct it, recovering the original message. This is crucial for certain applications where the re-sending of the message is not possible (for example, for interplanetary communications and storage of data). The crucial problem to be resolved then is how to add this redundancy in order to detect and correct errors in the most efficient way.

Error-correcting codes are particularly suited when the transmission channel is noisy. This is the case of wireless communication. Nowadays, all digital wireless communications use error-correcting codes.

Error correction is most useful in three contexts:

1. Simplex links (e.g., those that provide only one-way communication).
2. Long delay paths, where retransmitting data leads to long delays (e.g., satellites).
3. Links with very high error rates, where there is often one or two errors in each frame. Without forward error correction, most frames would be damaged, and retransmitting them would result in the frames becoming garbled again.

Rather than transmitting digital data in a raw bit for bit form, the data is encoded with extra bits at the source. The longer "code word" is then transmitted, and the receiver can decode it to retrieve the desired information. An error occurs when the receiver reads 1 as 0 or 0 as 1. To correct the error the receiver simply reverses the altered bit. So we are mainly to determine the position of bit where error has occurred. The hamming code discussed next, explains how to correct a single-bit error.

Hamming Code

In 1950, Richard Hamming developed an innovative way of adding bits to a number in such a way that transmission errors involving no more than a single bit could be detected and corrected.

The number of parity bits to be sent along with the data depends on the number of data bits. Some possible values are shown in table 9.1.

Data Bits :	4	8	16	32	64	128
Parity Bits :	3	4	5	6	7	8
Codeword :	7	12	21	38	71	136

Table 9.1

We can say that for N data bits, $(\log_2 N) + 1$ parity bits are required. In other words, for a data of size 2^n bits, $n + 1$ parity bits are embedded to form the codeword. It's interesting to note that doubling the number of data bits results in the addition of only 1 more data bit. Of course, the longer the codeword, and the greater the chance that more than error might occur.

Placing the Parity Bits

(From this point onward we will number the bits from left to right, beginning with 1. In other words, bit 1 is the most significant bit.)

The parity bit positions are powers of 2: {1, 2, 4, 8, 16, 32, ...}. All remaining positions hold data bits. Here is a table representing a 21-bit code word:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P	P		P				P								P					

The 16-bit data value 1000111100110101 would be stored as follows: (For 16-bit data there will 5 parity bits, so total bits in this codeword will be 21.)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P	P	1	P	0	0	0	P	1	1	1	1	0	0	1	P	1	0	1	0	1

Calculating Parity

For any data bit located in position N in the code word, the bit is checked by parity bits in different positions. The following table shows exactly which data bits are checked by each parity bit in a 21-bit code word:

Parity Bit	Checks the following Data Bits	Hint*
1	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21	Use 1, skip 1, use 1, skip 1, ...
2	2, 3, 6, 7, 10, 11, 14, 15, 18, 19	Use 2, skip 2, use 2, skip 2, ...
4	4, 5, 6, 7, 12, 13, 14, 15, 20, 21	Use 4, skip 4, use 4, ...
8	8, 9, 10, 11, 12, 13, 14, 15	Use 8, skip 8, use 8, ...
16	16, 17, 18, 19, 20, 21	Use 16, skip 16, ...

Table 9.2

To see the logic behind building this table, look at the binary representation of each bit position. The p1 bit is calculated using all bit positions whose binary representation includes 1 in the right most position. This is explained below:

Binary representation of 1 is 0001. Then we will check for all values from 1 to 21 whose binary representations include 1 in right most position. You can verify that binary representations of all data bits in row 1 of table 9.2 include 1 in right most position.(3–0011, 5–0101, 7 -0111 and so on)

Similarly, the *p*2 bit calculated using bit positions with a 1 in second position and p4 bit calculated using bit positions with a 1 in third position and so on.

It is useful to view each row in this table as a bit group. As we will see later, error-correcting using the Hamming encoding method is based on the intersections between these groups, or sets, of bits.

Encoding a Data Value

Now it's time to put all of this information together and create a code word. We will use even parity for each bit group, which is an arbitrary decision. We might just as easily have decided to use odd parity.

For the first example, let's use the 8-bit data value 1 1 1 0 1 0 0 1, which will produce a 12-bit code word. Let's start by filling in the data bits:

1	2	3	4	5	6	7	8	9	10	11	12
P	P	1	P	1	1	0	P	1	0	0	1

Next, we begin calculating and inserting each of the parity bits.

P1 : To calculate the parity bit in position 1, we sum the bits in positions 3, 5, 7, 9 and 11 : (1 + 1 + 0 + 1 + 0 = 3). This sum is odd (we are considering even parity), so parity bit should be assigned a value of 1. By doing this, we allow the parity to remain even:

1	2	3	4	5	6	7	8	9	10	11	12
1	P	1	P	1	1	0	P	1	0	0	1

P2 : To generate the parity bit in position 2, we sum the bits in positions 3, 6, 7, 10 and 11 : (1 + 1 + 0 + 0 + 0 = 2). The sum is even, so we assign a value of 0 to parity bit 2. This produces even parity for the combined group of bits 2, 3, 6, 7, 10, and 11:

1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	P	1	1	0	P	1	0	0	1

P4 : To generate the parity bit in position 4, we sum the bits in positions 5, 6, 7 and 12 : (1 + 1 + 0 + 1 = 3). The sum is odd, so we set parity bit 4 to 1, making the parity even:

1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	1	0	<u>P</u>	1	0	0	1

P8 : To generate the parity bit in position 8, we sum the bits in positions 9, 10, 11 and 12 : (1 + 0 + 0 + 1 = 2). This results in even parity, so we set parity bit 8 to zero, leaving the parity even:

1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	1	0	0	1	0	0	1

All parity bits have been created, and the resulting code word is :

1 0 1 1 1 1 0 0 1 0 0 1

Detecting a Single Error

When a code word is received, the receiver must verify the correctness of the data. This is accomplished by counting the 1 bits in each bit group (mentioned earlier) and verifying that each has even parity. Recall that we arbitrarily decided to use even parity when creating code words. Here are the bit groups for a 12-bit code value:

Parity Bit	Bit Group
1	1, 3, 5, 7, 9, 11
2	2, 3, 6, 7, 10, 11
4	4, 5, 6, 7, 12
8	8, 9, 10, 11, 12

If one of these groups produces an odd number of bits, the receiver knows that a transmission error occurred. As long as only a single bit was altered, it can be corrected. The method can be best shown using concrete examples.

Example 1 : Suppose that the bit in position 4 was reversed, producing 101011001001. The receiver would detect an odd parity in the bit group associated with parity bit 4. After eliminating all bits from this group that also appear in other groups, the only remaining bit is bit 4. The receiver would toggle this bit, thus correcting the transmission error.

Example 2 : Suppose that bit 7 was reversed, producing 101111101001. The bit groups based on parity bits 1, 2, and 4 would have odd parity. The only bit that is shared by all three groups (the intersection of the three sets of bits) is bit 7, so again the error bit is identified:

Parity Bit	Bit Group
1	1, 3, 5, $\underline{7}$, 9, 11
2	2, 3, 6, $\underline{7}$, 10, 11
4	4, 5, 6, $\underline{7}$, 12
8	8, 9, 10, 11, 12

Example 3 : Suppose that bit 6 was reversed, producing 101110001001. The groups based on parity bits 2 and 4 would have odd parity. Notice that two bits are shared by these two groups (their intersection): 6 and 7:

Parity Bit	Bit Group
1	1, 3, 5, 7, 9, 11
2	2, 3, $\underline{6}$, $\underline{7}$
4	4, 5, $\underline{6}$, $\underline{7}$, 12
8	8, 9, 10, 11, 12

But then, but 7 occurs in group 1, which has even parity. This leaves bit 6 as the only choice as the incorrect bit.

Alternate way of locating an error in a bit

The receiver takes the transmission and recalculates new values for $p1, p2, p4$ and $p8$ using the same sets of bits used by the sender plus the relevant parity (p) bit for each set. Then it assembles the new parity values into a binary number in order of p position:

$$(\boldsymbol{p8, p4, p2, p1})$$

Now consider the example 2 in which bit 7 is altered during transmission. Consider that the following data was transmitted with error in bit 7 .

1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	1	**1**	0	1	0	0	1

Now using this data we will compute new values for $p1, p2, p4$ and $p8$.

$p1$: 1 (Check bits 1, 3, 5, 7, 9, 11 and if odd take value as 1 else 0)
$p2$: 1 (Check bits 2, 3, 6, 7, 10, 11 and if odd take value as 1 else 0)
$p4$: 1 (Check bits 4, 5, 6, 7, 12 and if odd take value as 1 else 0)
$p8$: 0 (Check bits 8, 9, 10, 11, 12 and if odd take value as 1 else 0)

Arranging this in order (**$p8, p4, p2, p1$**) comes out to be 0111, which is binary representation for 7, and hence it gives the location of error. Once this bit is located, the receiver can reverse its value and correct the error.

Limitations of this scheme

- Can't correct multiple errors:

If two errors occur then this scheme could detect the presence of an error, but it would not be possible to correct the error. Consider, for example, that both bits 5 and 7 were incorrect. The bit groups based on parity bit 2 would have odd parity. Groups 1 and 4, on the other hand, would have even parity because bits 5 and 7 would counteract each other:

Parity Bit	Bit Group
1	1, 3, 5, 7
2	2, 3, 6, 7
4	4, 5, 6, 7

This would incorrectly lead us to the conclusion that bit 2 is the culprit, as it is the only bit that does not occur in groups 1 and 4. But toggling bit 2 would not fix the error—it would simply make it worse.

- Error correction is relatively expensive (computationally and in bandwidth). For example, 10 redundancy bits are required to correct 1 single-bit error in a 1000-bit message. Detection? In contrast, detecting a single bit error requires only a single-bit, no matter how large the message.

9.3.3 Burst Errors

Most of the techniques described do not handle burst errors well. LRC/VRC does reasonably well, since a burst no longer than word length will be detected by the LRC. In order to improve the performance of parity or Hamming codes in the presence of burst errors, we can buffer a frame of words, each with its own redundant bits, and then send out the bits of the frame column by column rather than by rows. If a burst is shorter than the number of rows in the frame, then each row will have one or fewer errors, and they will all be detected (or corrected). This is *interleaving* and may be done on the bit, byte, word or block level. Burst errors are distributed over the interleaved (time-division multiplexed) channels, so that if *K* channels are interleaved (i.e., an interleaving depth of *K* is used), then each channel suffers an error of length roughly 1/*K* of the original burst error. In this way, the LRC/VRC can detect bursts that are no longer than the number of rows in a column. LRC/VRC is a form of *concatenated coding*, another effective way to increase the power of error correction methods and resistance to burst errors.

9.4 Elementary Data Link Protocols

Before discussing elementary data link protocols we will have a brief introduction with the variables and procedures used in the implementation of data link protocols:

- **DLL Interaction with the Physical Layer**

- to_physical_layer to transmit frame.
- from_physical_layer to receive frame.

- **DLL Interaction with the Network Layer**
  - to_network_layer to pass packet upwards.
  - from_network_layer to get packet from network layer.
- **Wait-for-event (and event) :**

  This procedure only returns when something has happened, for example a frame arrives. Event says what happened (e.g., event = cksum_err).
- **Frame contents - kind, seq_no, ack, info**
  - First 3 contain control info and the last contains actual data.
  - Kind lets us know if frame contains only control information or if it contains control info. + data.
  - seq_no is used to number frames to differentiate them.
  - ack is used for acknowledgements.
  - info field of data frame contains a single packet. For a control frame info field is not used.
- **Packet and Frame:**
  - Network layer constructs a packet by taking a message from the transport layer and adds the network layer header to it.
  - Packet is then passed to DLL and put into the info field of the outgoing frame.
  - The destination DLL extracts the packet from the frame and sends packet to network layer (above it).
- Some more Procedures:
  - wait_for_event waits in a loop for events to occur. For example when a frame is sent by DLL, it starts a timer (in order to account for lost frames/replies). wait_for_event returns event = timeout.
  - to_network_layer, from_network_layer used to pass packets between network layer and DLL.
  - to_physical_layer, from_physical_layer used to pass frames between DLL and physical layer.
  - Frame number is counted from 0 - MAX_SEQ.

9.4.1 An Unrestricted Simplex Protocol/Utopia

In order to appreciate the step-by-step development of efficient and complex protocols such as SDLC, HDLC etc., we will begin with a simple but unrealistic protocol.

This is a simple data link protocol which makes certain assumptions that are far from true in the real world, but its sole purpose is to explain the working of the data link layer and then incrementally taking the assumptions off.

Assumption 1 :	Data is transmitted in only one direction.
Assumption 2 :	At both ends the network layers are always ready.
Assumption 3 :	Infinite buffer size of the sender, receiver.
Assumption 4 :	There is no error on the communication channel.
Assumption 5 :	Processing time can be ignored.

It is also known as utopia because network layer processes incoming data infinitely quickly and there is an infinite amount of buffer space. The following two procedures are implemented in this protocol:

- **Sender** (runs in DLL of the source machine):
  - ☐ Go get something to send (buffer)
  - ☐ Copy it into info. field for transmission
  - ☐ Send information—do forever (in infinite loop)
- **Receiver** (runs in the DLL of the destination machine)
  - ☐ Frame arrival
  - ☐ Get the inbound frame
  - ☐ Pass data to host—do forever (in infinite loop)

Receiver processes frames as they are sent. There are no control fields. The channel is also assumed to be perfect.

Using state diagrams to represent protocols

The Utopia protocol can be represented using a state diagram for the sender and receiver.

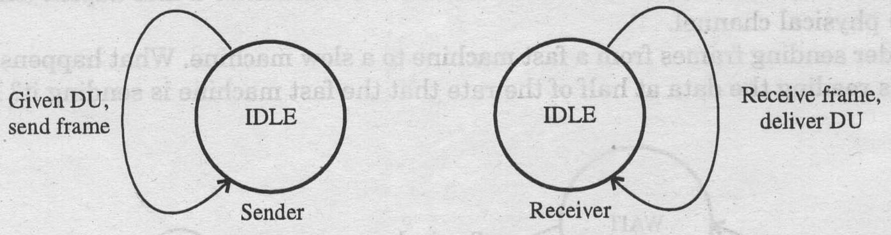


Fig. 9.7 State diagram for utopia protocol

About finite state machine diagrams

Finite state machines (FSM's) consist of states and arcs. The machine begins at a designated state. Arcs are directed arrows, each from a single state to another state (possibly the same one). Each arc has associated with it a list of conditions and actions. An arc is traversed, i.e., the machine may change state, if the conditions on the arc are met. When the arc is traversed the machine must perform the actions on the arc. It may not be feasible to use FSM's for all of the protocols that we describe.

The Utopia protocol provides a so-called best-effort service because it does the least amount of work to actually send the DU's (Data Units). In less than ideal circumstances Utopia may still be practical, e.g., when the probability of error is very small.

The summary of utopia is provided in the following table:

The Protocol:
• Sender collects message from network layer. • Prepares a frame buffer • Transfers to physical layer. • Receiver keeps waiting for frame arrival event. • On Event it collects frame from physical layer. • Transfers message to network Layer.
Analysis:
• Flow management is absent. • Works on a perfect channel only. • Unrealistic, thus little practical use.

9.4.2 Simplex Stop-and-Wait Protocol

Drop assumption that receiver can process incoming data infinitely fast. Stop-and-wait protocols are that where the sender sends one frame and then waits for acknowledgement. In this protocol, the contents of the acknowledgement frame are unimportant. Data transmission is one directional, but must have bidirectional line. It could have a half-duplex (one direction at a time) physical channel.

Consider sending frames from a fast machine to a slow machine. What happens if the slow machine is reading the data at half of the rate that the fast machine is sending it? Eventually

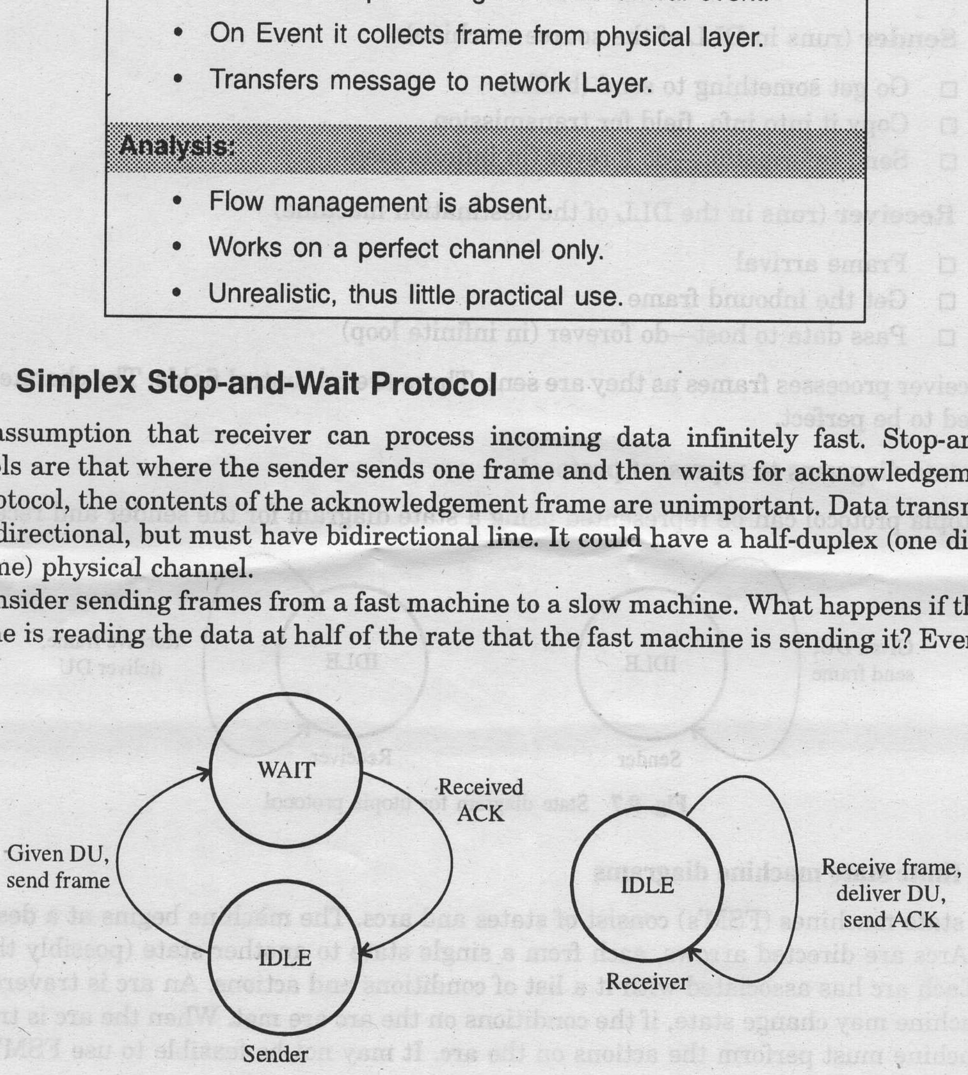


Fig. 9.8 State diagram for simplex stop-and-wait protocol

the sent data will be lost and never to be received. For this reason it is important to provide some kind of flow control.

The Stop-and-Wait protocol requires the receiver to send an acknowledgment PDU in return for every frame received. Such a PDU is often called an ACK. The sender will wait for the ACK before sending the next frame.

Time-lines

Time-lines are often used to demonstrate the operation of a protocol, e.g. the Stop-and-Wait protocol could be demonstrated as follows:

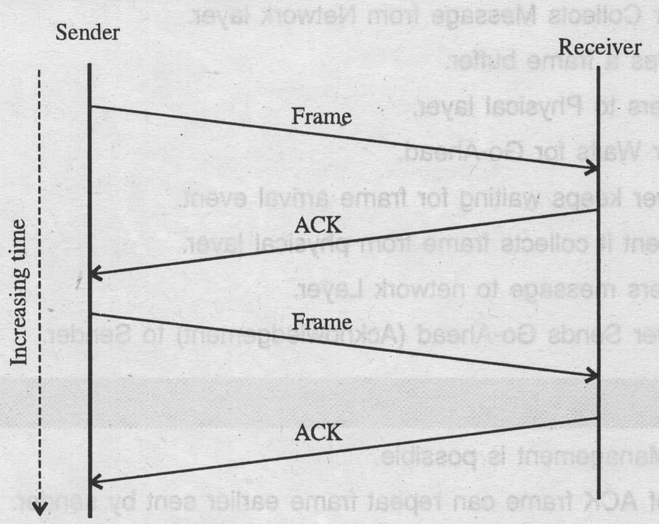


Fig. 9-9 Time-lines demonstrating stop-and-wait protocol

Properties of Stop-and-wait flow control

- Simplest form of flow control.
- Source entity transmits a frame.
- Destination entity sends back a signal to acknowledge receipt and to indicate that it can receive another frame.
- Source waits for acknowledgement before sending another frame.
- Destination can stop flow of data by withholding acknowledgement.
- Works well for sending a few large blocks.
- Preferable to send smaller blocks.
  - ☐ Buffer size of receiver may be limited.
  - ☐ Shorter transmission time; damaged frame can be retransmitted in short time.
  - ☐ One station does not monopolize shared transmission medium (LAN) for extended time.
- Stop-and-wait not very good for multiple frames for a single message.

- ☐ Only one frame at a time can be in transit.
- ☐ If bit length of link is greater than frame length, we can have serious line deficiencies.
- ☐ Bit length is the number of bits on the link when stream of bits fully occupies the link.

The summary of Stop-and-Wait protocol is provided in the following table:

The Protocol:
• Sender Collects Message from Network layer. • Prepares a frame buffer. • Transfers to Physical layer. • Sender Waits for Go-Ahead. • Receiver keeps waiting for frame arrival event. • On Event it collects frame from physical layer. • Transfers message to network Layer. • Receiver Sends Go-Ahead (Acknowledgement) to Sender.
Analysis:
• Flow Management is possible. • Loss of ACK frame can repeat frame earlier sent by sender. • Receiver cannot detect repeated frame. • Channel on Other Side is required for returning the control frame. • Half-Duplex Physical channel can suffice as traffic at a time is only in one direction.

9.4.3 A Simplex Protocol for a Noisy Channel

Consider assumption 4 is no longer true; frames can be lost or damaged. But the receiver can detect damaged frames. This protocol suffer from a number of flaws—frames submitted out of order, duplication of frames, if an ack frame is lost the sender is kept waiting. Protocols in which the sender waits for an acknowledgment before sending the next frame are called **ARQ (Automatic Repeat Request) or PAR (Positive Ack with Retransmission).**

The protocols have a timeout for retransmission to deal with lost frames. This still does not solve the problem of duplication that requires for the frames to have sequence numbers. Scenario of what could happen:

- A transmits frame one.
- B receives A1.
- B generates ACK.

- ACK is lost.
- A times out, retransmits.
- B gets duplicate copy (sending on to network layer).

To avoid duplicate frames, this protocol uses a sequence number. How many bits? 1-bit is sufficient because only concerned about two successive frames.

How long should the timer be? What if too long? (inefficient) What if too short? A problem because the ACK does not contain the sequence number of the frame which is being ACK'ed.

Scenario:

- A sends frame zero
- timeout of A
- resend frame A0
- B receives A0, ACKS
- B receives A0 again, ACKS again (does not accept)
- A gets A0 ACK, sends frame A1
- A1 gets lost
- A gets second A0 ACK, sends A2
- B gets A2 (rejects, not correct seq. number)
- will lose two packets before getting back on track (with A3,1)

This protocol differs from the previous two protocols in the respect that both sender and receiver have a variable whose value is remembered while the DLL is in the wait state.

Properties of Stop-and-Wait ARQ

- Very simple process.
- Based on stop-and-wait flow control technique.
- Source sends a frame and waits for ack.
- No other data frames are sent till receiver's ack is received.
- Error possibilities:

  1. *Frame damaged in transit*

     - ☐ Receiver detects damaged frame using error-detection techniques, and discards the frame.
     - ☐ No ack is sent to sender.
     - ☐ Sender retransmits after timeout expiration.
     - ☐ Sender must maintain a copy of what was sent till an ack is received.

  2. *Damaged ack*

     - ☐ Frame correctly received by receiver but ack damaged in transit.
     - ☐ Ack not recognized by sender who sends a second copy of same frame.
     - ☐ Receiver ends up with two copies of same frame.
     - ☐ Problem solved by labeling frames with 0 or 1.

The summary of ARQ is provided in the following table:

The Protocol:
• Sender Collects Message from Network layer. • Prepares a frame buffer with Info and Frame Sequence no. • Transfers to Physical layer. • Start a timer to time-out if ACK is not received. • Sender Waits for Next-Frame Event. • Receiver keeps waiting for frame arrival event. • On Event it collects frame from physical layer. • Check whether Frame Sequence is as expected. • Transfers message to network Layer. • Receiver Sends Go-Ahead (Acknowledgement) to Sender.
Analysis:
• Only Positive ACK is sent and timer controls retransmission. (PAR—positive ack with retransmission, ARQ—Automatic Repeat Request) • For a One to One ACK model 1 bit Frame Sequence number is sufficient. • May not be very efficient, as acknowledgement timer has to reset on every frame transaction. • Still a Half-Duplex Model.

9.5 Sliding Window Protocols

All the previously discussed protocols could possibly be used for a unidirectional data transfer; in the real world we need a bidirectional communication channel. One-way of doing this would be to have two separate channels, one for the data (forward) channel and one for its acknowledgments (reverse), but the reverse channel would almost be entirely wasted. A big improvement over this is using the same channel for transmitting data and control frames.

How do we achieve full duplex data transmission?

- **Possible solution**—Have two separate communication channels and use each one for simplex data traffic. **Disadvantage**—Bandwidth is almost entirely wasted.
- **Better solution**—Use the same circuit for data in both directions. By looking at the *kind* field in the header of an incoming frame, the receiver can tell whether the frame is data or ack.
- **Best solution—Piggybacking**—It is a technique of temporarily delaying outgoing acks so that they can be hooked onto the next outgoing data frame. **Advantage**—Better use of available channel bandwidth. Piggybacking introduces new problem. How long should the DLL wait for a packet onto which to piggyback the ack? Possible solution—If new packet arrives quickly, ack is piggybacked onto it. If no new packet has arrived by the end of a time period, DLL just sends separate ack frame.

Sliding Window Protocols assumes two-way communication (full duplex). It uses two types of frames:

1. Data.
2. Ack (sequence number of last correctly received frame).

To guarantee lossless delivery, a unique (increasing) sequence number (ID) is attached to every message. The combination of the message's sender address and its ID identifies every message uniquely. To overcome message loss, sender and receiver agree on some start ID. Now the IDs of messages received from a sender S always have to monotonically increase: if message 42 was received from S, the next message has to be 43. If a gap is detected, the receiver sends a retransmission request to the sender of the message. Upon receiving a retransmission request, the sender resends the message with the requested ID. Being able to resend old messages requires the senders to store recently sent messages for some time, usually until it is known that the receiver(s) has/have received all messages below a certain ID. In this case the messages below this ID may be deleted. When a receiver only sends retransmission requests when a gap is detected, the mechanism is called negative acknowledgment NAK. NAK is used in systems where message loss is infrequent, so NAKs would only rarely have to be used to retransmit lost messages.

When communication links are noisy and loss rate high, ACK schemes are preferred: a sender has to receive an acknowledgment from each receiver of a message. If ACKs are not received, the message is resent until an ACK has been received.

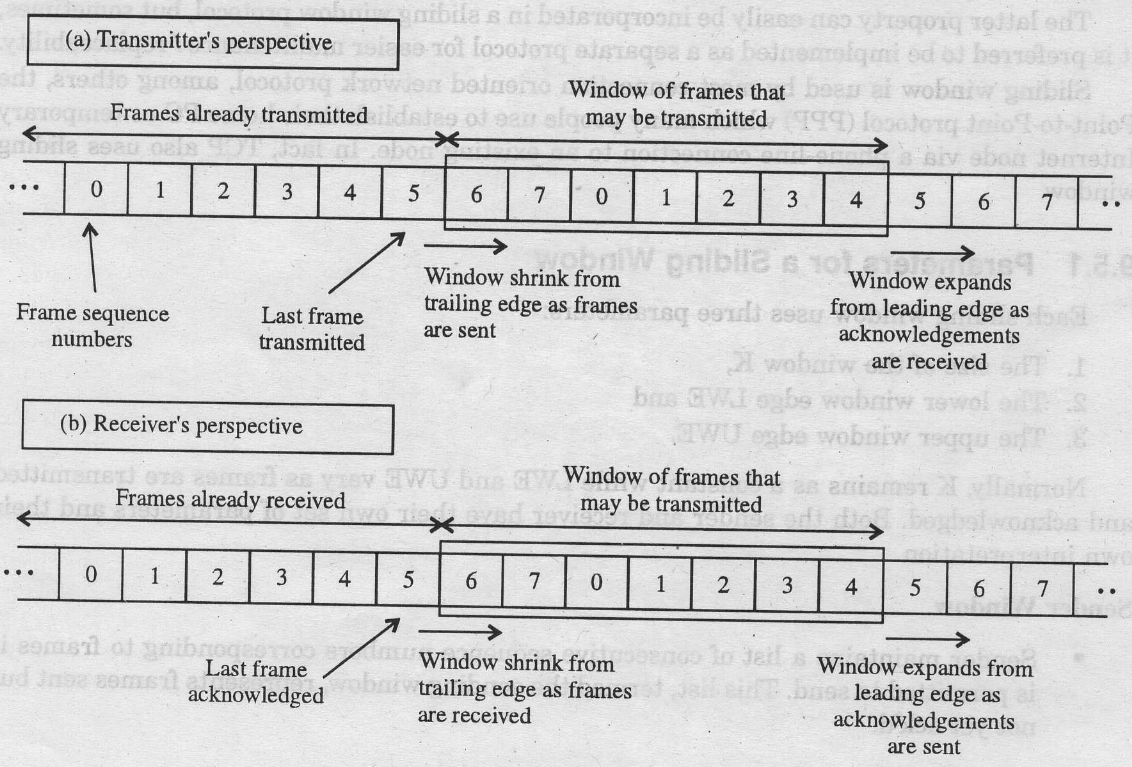

(a) Transmitter's perspective
Frames already transmitted
Window of frames that may be transmitted
... 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 ..
Frame sequence numbers
Last frame transmitted
Window shrink from trailing edge as frames are sent
Window expands from leading edge as acknowledgements are received
(b) Receiver's perspective
Frames already received
Window of frames that may be transmitted
... 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 ..
Last frame acknowledged
Window shrink from trailing edge as frames are received
Window expands from leading edge as acknowledgements are sent

Fig. 9-10 Sliding windows

The idea of sliding windows is to keep track of the acknowledgements for each ID. However, a scheme in which a sender send a single message (e.g., to multiple receivers in a group) and then waits for all ACKs is too slow: a sender should be able to send a number of messages and a separate thread should receive ACKs, and resend messages with ACKs missing.

The senders and receivers each maintain a window of messages for which no ACKs have been received : **a window** is essentially a sequence of message IDs, starting with a low water mark and bounded by a high water mark. The sender keeps the value of expected acknowledgement; while the receiver keeps the value of expected receiving frame. Whenever an ACK is received, the low and high water marks are advanced by 1; this allows 1 more ACK to be received, therefore sliding the window 1 to the right. When the window is full, an ACK is either discarded, or some kind of flow control is used to throttle the sender until there is more space available. When the receiver receives the expected frame, it advances the window.

Sliding windows usually start out with a given size, however, more sophisticated protocols will dynamically adapt the window size, trying to find an agreed-upon size between sender and receiver.

The characteristics of sliding windows used at the sender and receiver usually involve:

(*a*) Error correction (by retransmission),
(*b*) Flow control
(*c*) Message ordering by sender (FIFO).

The latter property can easily be incorporated in a sliding window protocol, but sometimes, it is preferred to be implemented as a separate protocol for easier maintenance / replaceability.

Sliding window is used by most connection oriented network protocol, among others, the Point-to-Point protocol (PPP) which many people use to establish their home PC as temporary Internet node via a phone-line connection to an existing node. In fact, TCP also uses sliding window.

9.5.1 Parameters for a Sliding Window

Each sliding window uses three parameters:

1. The size of the window K,
2. The lower window edge LWE and
3. The upper window edge UWE.

Normally, K remains as a constant while LWE and UWE vary as frames are transmitted and acknowledged. Both the sender and receiver have their own set of parameters and their own interpretation.

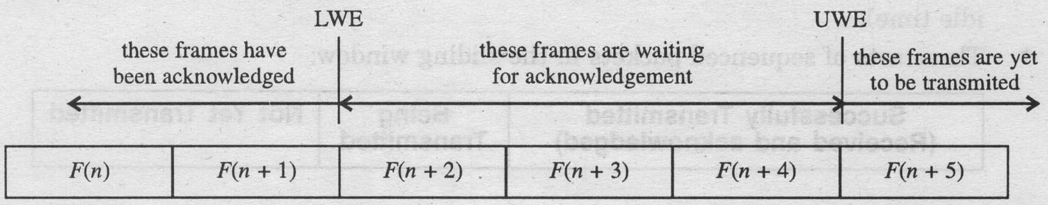
Sender Window

- Sender maintains a list of consecutive sequence numbers corresponding to frames it is permitted to send. This list, termed the sending window, represents frames sent but not yet ack'd.

- When an ack arrives, the lower edge of the window is advanced to the corresponding sequence number, thereby allowing the sender to transmit new frames.

Note : When receiver sends an ack for frame 's', this is understood to mean that all frames up to and including 's' have been received.)

General example window for sender

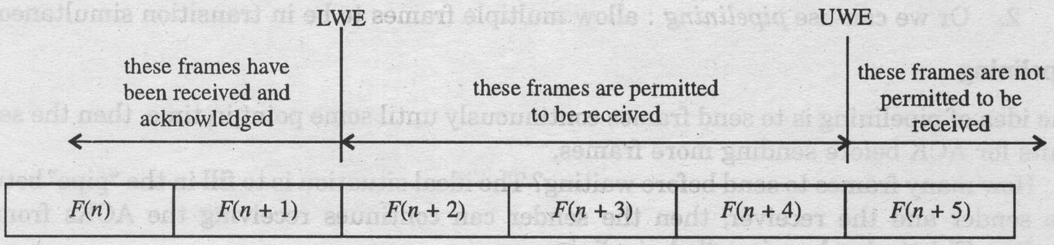

LWE
UWE
these frames have been acknowledged
these frames are waiting for acknowledgement
these frames are yet to be transmited
F(n)
F(n + 1)
F(n + 2)
F(n + 3)
F(n + 4)
F(n + 5)

The **LWE** and **UWE** are initialized to 0. A frame can be transmitted by the sender if **UWE—LWE < K**. When a frame is transmitted **UWE** is incremented by 1. The **LWE** is incremented as ACK'S 's are received. In this way, there can be no more than **K** frames in the send buffer.

Receiver Window

- The receiver also maintains a receiving window, corresponding to the number of out-of-order frames it can accept. Frames falling outside the window are discarded without ack.

General example window for receiver


LWE
UWE
these frames have been received and acknowledged
these frames are permitted to be received
these frames are not permitted to be received
F(n)
F(n + 1)
F(n + 2)
F(n + 3)
F(n + 4)
F(n + 5)

For the receiver, the semantics are slightly different. Frames within the window are those frames that are permitted to be received. Any other frames will be discarded. **LWE** can move forward any number of consecutive frames that have been acknowledged. Always **UWE = LWE + K.**

Properties of sliding window protocols:

- Used in TCP to make the transmission stream efficient
- Keep track of many acknowledged and un-acknowledged packets, instead of one at a time.
- Use network bandwidth better because they allow the sender to transmit multiple packets before waiting on each ACK.

- Transmit all packets in the window and slide it forward for each ACK received.
- Unacknowledged packets are constrained by the windows size (limited to a small fixed number).
- Sliding window performance depends upon the window size and speed in which the destination processes the packets.
- Well-tuned sliding window protocols keep the network saturated with packets (zero idle time).
- Three sets of sequenced packets in the sliding window:

Successfully Transmitted (Received and acknowledged)	Being Transmitted	Not Yet Transmitted

9.5.2 Stop-and-Wait—One Bit Sliding Window Protocol

One bit sliding window protocol is also called Stop-And-Wait protocol. In this protocol, the sender sends out one frame, waits for acknowledgment before sending next frame, thus the name Stop-And-Wait. This protocol requires little buffer space but provides poor line utilization.

Problem with Stop-And-Wait protocol is that it is very inefficient. At any one moment, only one frame is in transition. The sender will have to wait at least one round trip time before sending next. The waiting can be long for a slow network such as satellite link.

How to fix it?

1. We can use large frames;
2. Or we can use *pipelining* : allow multiple frames to be in transition simultaneously.

Pipelining

The idea of pipelining is to send frames continuously until some point in time, then the sender waits for ACK before sending more frames.

How many frames to send before waiting? The ideal situation is to fill in the "pipe" between the sender and the receiver, then the sender can continues receiving the ACKs from the receiver. This technology is called pipelining.

How to deal with the problem of errors? Two possible solutions exist.

1. Go Back n.
2. Selective Repeat.

9.5.3 Go-Back-n

What to do when a frame in the middle of a long stream is bad? Large number of succeeding frames may come to receiver before sender is told of error. On discovering error, a solution is go back *n*. Here receiver discards all subsequent frames (upto *n*) and sends no ack for the discarded frames. Eventually sender will retransmit all frames including lost/damaged one.

If there is one frame *k* missing, the receiver simply discards all subsequent frames *k* + 1,

$k + 2$, ..., sending no acknowledgments. So the sender will retransmit frames from k onwards. This can waste a lot of bandwidth if there is a high error rate.

The receiver detects receipt of an out-of-sequence frame and requests sender to retransmit all outstanding unACKd frames from the last correctly received frame. Or, the receiver sends a NAK to the primary when it first detects an out-of-sequence data frame. Thereafter the receiver discards all data frames until it receives the missing frame.

If an ACK is corrupted (say N and $N + 1$), then the sender will infer from receipt of the ACK for $N + 2$ that both N and $N + 1$ were received correctly, and will take the $N + 2$ ACK as sufficient for all three frames.

Since the receiver doesn't have to store frames out-of-sequence, it doesn't require much buffer space. Remember that it throws away the out-of-sequence frames it receives. Go-Back-*N* handles duplicate (lost ACKs) frames easily by simply discarding them. Selective-Retransmission can only know to discard a duplicate by maintaining a history list of received frames.

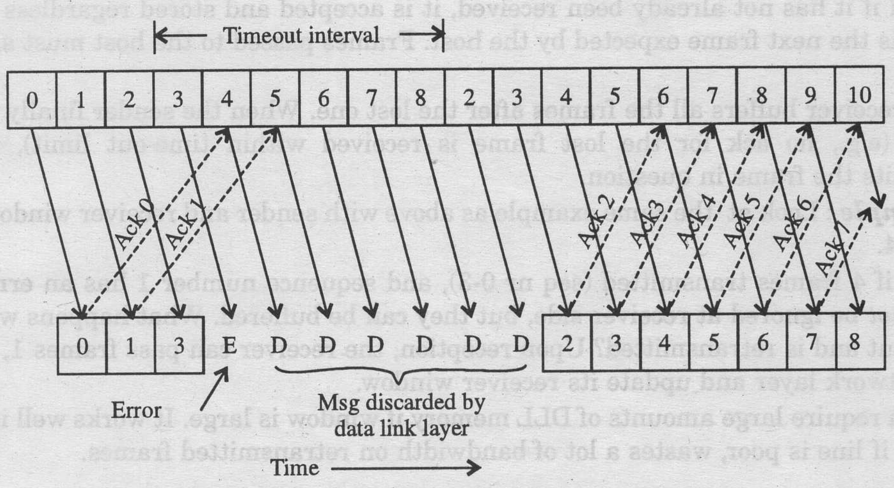


Fig. 9.11 Time line diagram for go back N

Most data communications is full duplex, i.e., you have as much bandwidth in one direction as you do in another. If you only use the reverse channel to send ACK/NAKs, then you will be wasting the bandwidth of the reverse channel (low utilization). One idea is to piggyback ACK/NAK traffic on the information frames that are flowing on the reverse channel.

Disadvantage of Go Back N

The error handling is not as efficient for Go Back N, since valid received frames are discarded.

Example : Look at an example with sender and receiver window size each equal to 4.

Now if 4 frames transmitted (seq nr 0–3), and sequence number 1 has an error. Subsequent frames 2–3 will also be discarded by receiver side. When Frame 1 times out, sender retransmits frame 1 as well as frames 2, 3. Upon reception, now the receiver can pass frames 1, 2 and 3 up to the network layer and update its receiver window.

9.5.4 Selective Repeat

Another strategy is to re-send only the ones that are actually lost or damaged. This protocol employs **the selective reject technique**. The protocol does not discard good frames because an earlier frame was damaged or lost provided that these good frames fall within the receiving window. Associated with each outstanding frame is a timer. When the timer goes off, (or when the transmitter is notified of any error), only that one frame is retransmitted, not all the outstanding frames, as in protocol go back *n*.

In this protocol, the receiver's window size is fixed, the maximum of which is (MaxSeq+1)/2. The maximum number is thus chosen to ensure that there will not be an overlapping between the new window and the previous window. The overlapping of windows means that the receiver would not be able to differentiate between a new frame and a retransmitted frame.

The receiver has a buffer reserved for each sequence number within its window. Whenever a frame arrives, its sequence number is checked to see if it falls within the receiver's window. If so, and if it has not already been received, it is accepted and stored regardless of whether or not it is the next frame expected by the host. Frames passed to the host must always be in order.

The receiver buffers all the frames after the lost one. When the sender finally noticed the problem (e.g., no ack for the lost frame is received within time-out limit), the sender retransmits the frame in question.

Example : Look at the same example as above with sender and receiver window size each equal to 4.

Now if 4 frames transmitted (seq nr 0-3), and sequence number 1 has an error. Frames 2-3 will *not* be ignored at receiver side, but they can be buffered. What happens when Frame 1 times out and is retransmitted? Upon reception, the receiver can pass frames 1, 2 and 3 up to the network layer and update its receiver window.

It can require large amounts of DLL memory if window is large. It works well if errors are rare, but if line is poor, wastes a lot of bandwidth on retransmitted frames.

Selective Repeat is more efficient than Go Back *n* in that

Selective repeat employs an auxiliary timer to prevent delay in piggybacking. If no reverse traffic has presented itself before the timer goes off, a separate acknowledgement is sent.

Whenever the receiver has reason to suspect that an error has occurred it sends a negative acknowledgement (NAK) frame back to the sender. Such a frame is a request for retransmission of the frame specified in the NAK.

9.6 Examples Of DLL Protocols

9.6.1 HDLC: High Level Data Link Control

Introduction

In this section we describe the International Standards Organization data link protocol HDLC (High-level Data Link Control). The HDLC protocol, an ISO data link layer protocol based on the IBM SDLC (Synchronous Data Link Control), is to ensure that data passed up to the next layer has been received exactly as transmitted (i.e., error free, without loss and in the correct order). Another important function of HDLC is flow control, which ensures that data is transmitted only as fast as the receiver can receive it). HDLC is a discipline for the management of information transfer over a data communication channel.

There are two distinct HDLC implementations: HDLC NRM (also known as SDLC) and HDLC Link Access Procedure Balanced (LAPB), the later is a more popular implementation. HDLC is usually used by X.25.

LAPB is a bit-oriented synchronous protocol that provides complete data transparency in a full-duplex point-to-point operation. It supports a peer-to-peer link in that neither end of the link plays the role of the permanent master station. HDLC NRM, on the other hand, has a permanent primary station with one or more secondary stations.

HDLC LAPB is a very efficient protocol, which requires a minimum of overhead to ensure flow control, error detection and recovery. If data is flowing in both directions (full duplex), the data frames themselves carry all the information required to ensure data integrity.

The concept of a frame window is used to send multiple frames before receiving confirmation that the first frame has been correctly received. This means that data can continue to flow in situations where there may be long "turn-around" time lags without stopping to wait for an acknowledgement. This kind of situation occurs, for instance in satellite communication.

Structure of HDLC

HDLC has a basic structure that governs the function and the use of control procedures. **The basic structure includes:**

- The definitions of primary and secondary station responsibilities.
- The design of information grouping for control and checking.
- The design of the format for transfer of information and control data.

Primary and secondary stations

A data link involves two or more participating stations. For control purposes one station on the link is designated a primary station, the others are secondary stations. The primary station is responsible for the organization of data flow and for the link level error recovery procedures.

A frame sent from a primary station is referred to as a command frame. A frame from a secondary to a primary is referred to as a response frame. Normally when a command is sent, a response or a string of responses is expected in reply.

On a point-to-point link either station could be the primary. On multidrop links and where polling is employed, the station that polls the other is the primary.

Frame structure

In HDLC the input data string to be transmitted over the link is broken up into data frames that are then transmitted sequentially. The input data string could be command, response or information.

1 byte	1-2 bytes	1 byte	variable	2 byte	1 byte
Flag	Address field	Control field	Information	FCS	Flag

Fig. 9.12 Frame format of HDLC

The header

Flag (01111110) : A frame is identified by this beginning flag *F* and contains only non-*F* bit patterns. HDLC uses bit-stuffing to achieve data transparency.

Address : Defines the address of the secondary station that is sending the frame or the destination of the frame sent by the primary station. It contains Service Access Point (6-bits), a Command/Response bit to indicate whether the frame relates to information frames (I-frames) being sent from the node or received by the node, and an address extension bit which is usually set to true to indicate that the address is of length one byte. When set to false it indicates an additional byte follows:

Extended address—HDLC provides another type of extension to the basic format. The address field may be extended to more than one byte by agreement between the involved parties.

Control Field : The control field contains information for controlling the link. It serves to identify the type of the frame. In addition, it includes sequence numbers, control features and error tracking according to the frame type. There are 3 types of frames:

I-frame.

Information frame. This type of frame carries user's data.

S-frame

Supervisory frame : This is used for supervisory control functions such as acknowledgements, requesting transmission and requesting a temporary suspension of transmission.

U-frame

Unnumbered frame or Unsequenced frame: This is used to provide additional link control functions like:

- Have a variety of purposes, but most often used for establishing the link setup and disconnect.
- Sets up the data transfer mode, sequence number size. Also used to reset the link and other miscellaneous stuff

The Trailer

Frame Check Sequence : The FCS field contains a 16-bit Cyclic Redundancy Check

(CRC) error detecting code. The Frame Check Sequence (FCS) enables a high level of physical error control by allowing the integrity of the transmitted frame data to be checked.

Flag : This flag indicates the end of the frame. When this flag is detected the receiver examines the preceding two bytes and executes its error detection algorithm.

A flag may be followed by a frame, by another flag, or by an idle condition.

Data link channel states

Active channel state

A channel is in an ACTIVE state when the primary or a secondary is actively transmitting a frame, a single abort sequence or interframe time fill. In this state the right to continue transmission is reserved.

Abort : A station can abort a frame by transmitting at least seven contiguous ones. The receiving station will ignore the frame.

Interframe time fill : In this state continuous flag bytes are sent between frames.

Idle channel state : A channel is defined to be in an IDLE state when the transmission of 15 or more contiguous one bits is detected. In this state a primary must repoll a secondary before transmitting an I-frame to it.

9.6.2 Data Link Layer in the Internet

Two protocols used:

- SLIP—Serial Line IP
- PPP—Point-to-Point Protocol

9.6.2.1 SLIP- Serial Line IP

SLIP stands for Serial Line IP. While we will discuss IP in the network layer (because IP involves in routing), SLIP is used at datalink layer for serial line connection to the Internet. SLIP assembles frames by adding flag byte (character stuffing) to a raw IP packet. Some SLIP protocls just add the flag at the end; while others add at both front and rear.

The protocol was devised by Rick Adams in 1984. Some problems with SLIP:

- SLIP does not do any error detection or correction. It is upto the higher layers (transport layer) to accomplish this.
- SLIP supports only IP, so it won't talk to Novell and other protocols.
- In earlier version of SLIP, each side must know the other's IP address in advance. This limits the use of it.
- SLIP does not provide any form of authentication, neither party knows whom it is really talking to.
- Not an approved standard, many versions exist.

9.6.2.2 PPP—Point-to-Point Protocol

To overcome the problems in SLIP, the Internet Engineering Task Force (IETF) set up a group to develop PPP. PPP is a multiprotocol framing mechanism.

- It is suitable for use over modems, HDLC bit-serial lines, SONET and other physical layers.
- It supports error detection, option negotiation, header compression, and optionally reliable transmission using HDLC framing.

The datalink layer for the TCP/IP protocol stack is part of the ill-defined bottom "host to network" layer that also includes the physical layer. The current Internet standard for a data link protocol is PPP (Point-to-Point Protocol).

PPP consists of LCP (link control protocol) for bringing lines up and down, negotiating features like compression, etc. and NCP (network control protocol) which allows for different sorts of network layers (and thus makes PPP capable of being used for more than just IP).

PPP also supports authentication and the dynamic assignment of a network layer address, as well as the option to provide reliable service via sequence numbering of frames. PPP uses character stuffing to avoid problems with the flag showing up in the data field.

It can be used for dial-up and leased router-router lines. It provides:

1. Framing method to delineate frames. Also handles error detection.
2. Link Control Protocol (LCP) for bringing lines up, negotiation of options, bringing them down. These are distinct PPP packets.
3. Network Control Protocol (NCP) for negotiating network layer options.

Dial-up scenario

1. PC user initiates connection to ISP router, modems synchronize and negotiate, channel is established.
2. LCP packets are sent via PPP frames to router and negotiation of link parameters is done.
3. NCP packets are sent; dynamically assigned IP address is returned from router.
4. PC now a part of Internet, higher-level Internet protocols will work (FTP, HTTP, TELNET, etc).
5. User is done with connection.
6. NCP tears down the network connection, freeing the assigned IP address for re-use.
7. LCP tears down the data link connection.
8. Modem is told to hang-up.

PPP frame format

The PPP frame is character oriented (8-bytes per character).

Flag	Address	Control	Protocol	Payload	Checksum	Flag
1 byte	1	1	1 or 2	Variable	2 or 4	1
01111110	11111111	00000011				0 1111110
Identifies the start of the frame; character stuffing used for this value in payload	There are no addresses, always all 1s; assumed point-to-point	This value indicates an unnumber-ed frame; sequence number could be used over noisy medium	Tells what the payload contains (LCP, NCP, IP, IPX, Appletalk, etc)	Negotiated max by LCP, usually 1500 bytes	Size is negotiated	Marks end of frame

Fig. 9.13 Frame format of PPP

LCP includes the ability to negotiate away some of the fields (like address and control) which aren't used in most cases. Authentication takes place and must be successful before the NCP even begins.

The strength of PPP is that it supports multiple network protocols, and can be used harmoniously with various physical mediums, including PSTN/modems and SONET.

● ● ●

10 Multiple Access Protocols

Introduction

In the IEEE 802 protocols for shared multi-access LANs, the data link layer is divided into two sublayers:

1. LLC (Logical Link Control) layer
2. MAC (Media Access Control) layer

The upper LLC layer provides a way to address a station on LAN and exchange information with it. The lower MAC layer provides the interface between the LLC and the particular network medium that is in use (Ethernet, token ring, and so on).

In general, the **Media Access Control** topics are about **sharing** and **connecting** to a

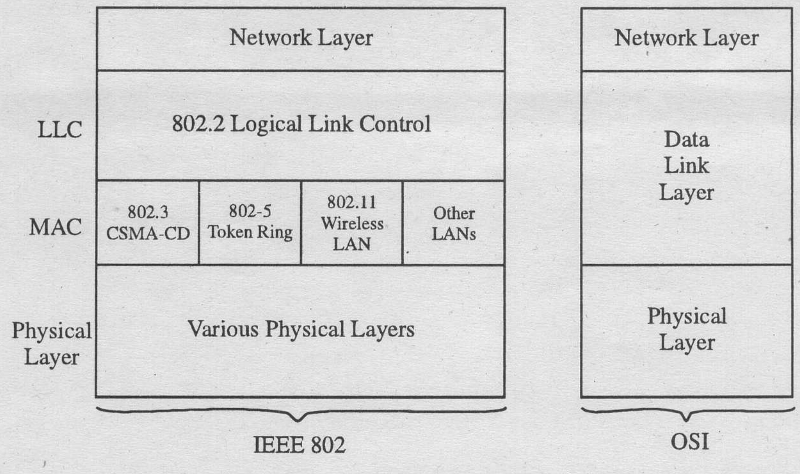


Fig. 10.1 IEEE standards mapped to the OSI model

medium. The **Logical Link Control** topics are about **setting up** and **maintaining** the connections for transmitting frames. (Already discussed in chapter 9). The MAC sublayer regulates the access to the channel shared by the nodes on a LAN. The LLC sublayer supervises the packet links between nodes, and presents a uniform interface to the network layer.

MAC involves controlling, which workstations can talk when. Easy part of layer 2 MAC is when a single wire directly connects two computers. More interesting part is when a number of workstations on a LAN share a single common wire or channel for communication; only one can use it at a time

"Garble" results when two workstations try to transmit at the same time. The goal of this layer is to maximize use of the channel & minimize waiting time for a workstation that wants to send.

FDM may work for wireless LANs or on some cables; TDM can be used in some cases; generally, they don't work well because computer communications are inherently bursty. The result is often that lots of channel capacity is wasted.

The MAC layer frames data for transmission over the network, and then passes the frame to the physical layer interface where it is transmitted as a stream of bits. Framing packages information into distinct units that are transmitted one at a time on the network. If a frame is corrupted during transmission, only it needs to be resent-not the entire transmission.

Individual LAN addresses are defined in the MAC layer. The MAC sublayer carries the physical address of each device on the network. This address is more commonly called a device's MAC address. The MAC address is a 48-bit address that's encoded on each network device by its manufacturer. It's the MAC address that the Physical layer uses to move data between nodes of the network. Workstations on the same LAN use the MAC address to forward packets to one another. When LANs are connected in an internetwork, higher-level addressing schemes such as IP (Internet Protocol) identify networks and nodes across the internetwork.

Assumptions about the LAN Environment

1. Network consists of N independent workstations, each generating FRAMES (sometimes called PACKETS) for transmission. For ease of analysis, we assume that all frames are the same size. (They don't have to be in real life, but it makes the analysis easier.)
2. There is only a single channel available for communication. All workstations can transmit on it, and all workstations can listen to it for frames addressed to them.
3. If two frames are transmitted simultaneously, they overlap and resulting signal is garbled. No errors on the communications channel other than from collisions between two or more frames.
4. There is an agreed-upon time rule for frame transmission. Either everybody is using continuous time (you can start whenever you are ready), or everybody is using slotted time (you can start transmitting only at the beginning of a predefined "slot").
5. There is an agreed-upon rule for carrier sense. Either all workstations are listing to the channel to see if it is in use, or nobody is.

The two entities of interest are as follows:

- **"Delay under light loading conditions"**—How long do I have to wait to transmit if I am ready and nobody else is?
- **"Channel utilization under heavy loading"**—When all work stations want to transmit data, how much of the time is the channel using to transmit data, and how much of the time is wasted for one reason or another?

10.1 Channel Allocation

The allocation of the channel can be done statically or dynamically. Static allocation is used in the telephone industry where the needs for the different stations (in this case telephone sets) are always the same. Thus it makes sense to split the available bandwidth and assign a fixed portion to each user. This strategy doesn't work for computer networks where the traffic is very busy. Thus in computer networks the channel allocation is done dynamically to adapt to the rapidly changing traffic.

Access Methods

Local area networks (LANs) are typically shared by a number of attached systems, and only one system at a time may use the network cable to transmit data. An access method defines how a system gains access to a shared network in a cooperative way so its transmissions do not interfere with the transmissions of other systems. Simultaneous access to the cable is either prevented by using a token-passing method or controlled with a carrier sensing and collision detection method.

Media Access is accomplished in three general ways:

- Contention—all devices send when they wish.
- Token-passing—devices send when it is their turn.
- Polling—devices are asked if they have anything to send.

Contention systems work by letting each device try to send a message on the net as needed, contending or competing with all the other devices for the bandwidth. Contention based access controls can work well for bursty traffic. They are asynchronous, distributed, random-access control techniques, simple to implement, and are generally efficient unless the network load is very high. The basic idea is to allow any node that wishes to do so, at any time, to transmit a frame. If it is successful in getting its frame through without interference from another node's transmission during its frame duration, then everything is fine. Otherwise, there is a mechanism for re-transmitting frames. The protocols that support such systems are known also as **Random Access Protocols.**

Two examples of protocols that support such systems are:

- **CSMA/CD** (Carrier Sense, Multiple Access, with Collision Detection)
- **CSMA/CA** (Carrier Sense, Multiple Access, with Collision Avoidance)

A **collision** occurs when two signal collide on the medium, causing signal loss. These are bursty protocols, best supporting intermittent transmissions. Only about 60 to 70 users can be supported on a segment. Time sensitivity is good, as users do not often have to wait for media access.

- **CSMA/CD (Carrier Sense Multiple Access/Collision Detection) :** Carrier sensing implies that network nodes listen for a carrier tone on the cable and send information when other devices are not transmitting. Multiple access means that many devices share the same cable. If two or more devices sense that the network is idle, they will attempt to access it simultaneously (contention), causing collisions. Each station must then back off and wait a certain amount of time before attempting to retransmit. Contention may be reduced by dividing networks with bridges or using switches. This is used by Ethernet.
- **CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) :** This access method is a variation on the CSMA/CD method. Nodes estimate when a collision might occur and avoid transmission during that period. This method is cheaper to implement, since collision detection circuitry is not required; however, it imposes more delay and can slow network throughput. In a CSMA/CA system, devices can be assigned time slices or can be required to ask permission to send, avoiding collisions. Apple LocalTalk is as an example of this.

Token Passing involves passing a token, a small data frame, from station to station. When a station has the token, it is that station's turn to access the medium. Examples of this method are:

- Token ring.
- FDDI.
- Token bus.

This type of media access is predictable and consistent, allowing large or small transmissions. It is not the best for time sensitive data since waits are built in, but it will support more devices than contention. Contention is best when the load is light and token passing is better with heavier loads and that both schemes crash under too much load.

Carrier sensing methods tend to be faster than token-passing methods, but collisions can bog down the network if it is heavily populated with devices. Token ring does not suffer from collision problems, but the current throughput rates on token ring networks are restrictive when compared to new high-speed Ethernet networks.

Polling is too slow and controlled to give the users the speed they expect from other network methods.

10.2 The ALOHA Random Access Protocol

The ALOHA protocol is an interesting example of a MAC protocol of the contention-type. It is the precursor of Ethernet and its subsequent standardization as IEEE 802.3 (CSMA-CD). To control which NICs are allowed to transmit at any given time, a protocol is required. This simplest protocol is known as ALOHA (this is actually an Hawaiian word, meaning "hello"). ALOHA allows any NIC to transmit at any time, but states that each NIC must add a checksum/CRC at the end of its transmission to allow the receiver(s) to identify whether the frame was correctly received.

The concept of the ALOHA system is applicable with modifications in other situations, such as a LAN of nodes attached to a common coaxial cable bus (the Ethernet concept). The actual implementations may be different, but the underlying concepts are the same for all ALOHA systems. The basic **characteristics of ALOHA** are the following:

- Fixed length frames, shared channel or medium.
- Frames carry address of destination node.
- Nodes transmit on common channel and listen to common channel for transmissions from other nodes.
- A node starts a frame transmission whenever it is ready to send, regardless of the state of the channel.
- A node can tell if its frame arrived at the destination node without collision, or if a collision occurred with another frame in an overlapping time interval thus corrupting both frames. (An ACK may be used, or the node listens to the downlink broadcast channel in the original ALOHA system for its own frame).

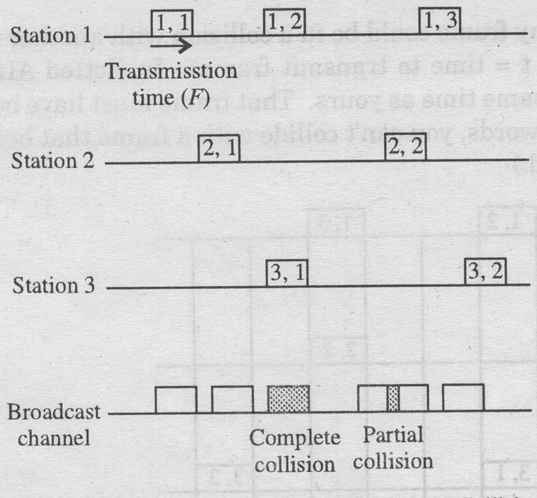
ALOHA is therefore a best effort service, and does not guarantee that the frame of data will actually reach the remote recipient without corruption. It therefore relies on ARQ protocols to retransmit any data, which is corrupted. An ALOHA network only works well when the medium has a low utilization, since this leads to a low probability of the transmission colliding with that of another computer, and hence a reasonable chance that the data is not corrupted. There are two categories of aloha:

1. Pure Aloha
2. Slotted Aloha

In pure ALOHA a node can start transmission at any time. In slotted ALOHA, all nodes have synchronized clocks marking frame boundary times (the clock period is the time for one frame transmission) and a node wishing to transmit does so at the start of the next frame slot. In both cases, a node transmits without checking the state of the channel. (In ALOHA, a node cannot necessarily "listen" to the uplink transmissions of another node, because these transmissions are directed by the transmitting antennae to the satellite station and not to the other nodes.)

10.2.1 Pure (Unslotted) ALOHA - 1

In this protocol a station transmits a frame whenever it has data to send. When the frame collides with other frame(s), retransmit it by waiting a random amount of time.



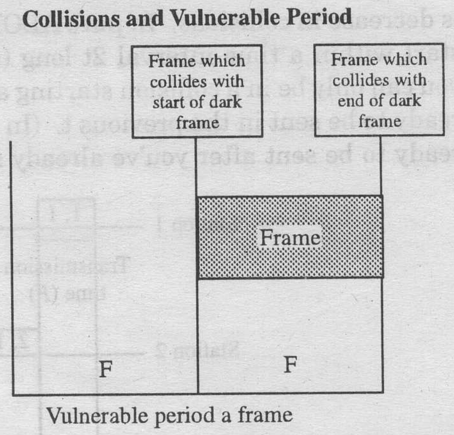

Collisions and Vulnerable Period
Frame which collides with start of dark frame
Frame which collides with end of dark frame
Frame
F
F
Vulnerable period a frame

Fig. 10.2 Collisions in pure ALOHA

A frame (dark frame) collides whenever another transmission begins in the vulnerable period of the frame. Vulnerable period has length of 2 frame times.

Salient features of Pure ALOHA

- Simpler, no synchronization.
- Station sends data immediately.
- If a collision occurs, the data is resent after a random amount of time (to reduce probability of another collision).
- Contention—the channel is shared in a way that can lead to conflicts.
- No carrier sensing.
- Vulnerability—based on the time it takes to transmit a frame.
- Send packets without waiting for beginning of slot. As a result collision probability increases: packet sent at t0 collide with other packets sent in $[t_0 - 1, t_0 + 1]$.

10.2.2 An improvement : "Slotted ALOHA"

Rather than continuous time, use slotted time. Define a slot to be the time it takes to transmit one frame. (This is where assuming all frames are the same size simplifies things). Everybody is kept synchronized on slot times by signal from main campus. You can only start at the beginning of a slot. If you become ready to transmit after a slot has begun, you have to wait until the beginning of the next slot.
e.g., slot is 30 seconds, if you're ready 5 seconds in; you have to wait 25 seconds, even if nobody else is transmitting.

"Delay under light loading" goes up to one-half * slot time, on average, under "slotted ALOHA".

"Utilization under heavy loading" goes to 36.8% - double than pure ALOHA. The reason is decrease in collisions. In pure ALOHA, any frame could be in a collision with another frame start within a time interval 2t long (where t = time to transmit frame). In slotted ALOHA, you can only be in a collision starting at the same time as yours. That frame must have become ready to be sent in the previous t. (In other words, you can't collide with a frame that becomes ready to be sent after you've already started.)

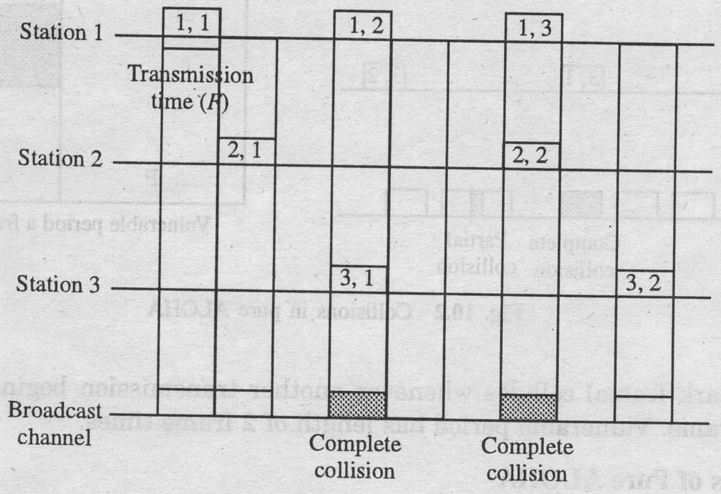


Fig. 10.3 Collisions in slotted ALOHA

Salient features of Slotted ALOHA

- Time is discrete. It is divided into equal size slots (= pkt trans. time).
- Timing required (cannot transmit frame until the beginning of the next time slot). A node with new arriving packet transmits at beginning of next slot.
- Increases throughput because of the way time is handled.
- Load dependent (increases in the load reduces performance).
- If collision occurs it retransmits packet in future slots with probability *p*, until successful.

Protocols can be improved by adding "carrier sense". In ALOHA, stations did not listen to the channel to see if anybody else was using it - they just transmitted.

10.3 Carrier Sense Multiple-Access (CSMA) Protocols

The fundamental reason for low channel utilization of Aloha protocol is that senders don't defer to each other even when another transmission is in progress. CSMA rectifies this problem by carrier sensing.

CSMA protocols are widely used in local area networks (LAN's) to control access to a shared communications channel such as a coaxial cable or radio frequency band. CSMA protocols are designed to exploit the property that the signal propagation time across the LAN is much smaller than the packet transmission time. Consequently, their performance depends on the exact timing of a sequence of asynchronous events originating at different points in the network.

In some shorter distance networks, it is possible to listen to the channel before transmitting. In radio networks, this is called "sensing the carrier". The CSMA protocol works just like Aloha except : If the channel is sensed busy, then the user waits to transmit its packet and a collision is avoided. This really improves the performance in short distance networks!

"Persistent" and "Non-Persistent" CSMA

Persistent CSMA (Most greedy) : A station listens to the carrier when it is ready to send; if it senses "empty", it transmits. If it senses "busy", it continues to listen until it senses "empty" and then immediately tries to transmit. (Called "1-persistent CSMA", because station transmits with probability 1 when it senses the carrier being empty)

Algorithm of 1-persistent CSMA

1. Station has data to send.
2. Listen to carrier.
3. If carrier is busy, wait until idle.
4. Transmit data.
5. If there is a collision, wait a random amount of time.
6. Goto 2.

1-persistent transmits as soon as the carrier becomes idle with a probability of 1. Because of propagation delay, a channel may send at the same time of another.

Two problems:

- **Obvious case :** Two workstations listening to carrier; will both sense "empty" at approximately the same time and will collide.
- **Less obvious :** All real transmission media have a finite propagation delay—it takes some time before a signal sent by one workstation is sensed by all other workstations on the wire. So a station listens to wire; senses it empty and begins transmission, only to find that somebody else had already been transmitting but signal hadn't made it that far yet. ***Result*** : collision.

Non-persistent CSMA (Least greedy) : A station listens to carrier; senses it busy; quits listening and waits a random time before starting to listen again. Each waiting station generates its own random wait time; chance of two stations being ready to transmit again at same time is small.

Algorithm of Non-persistent CSMA

1. Station has data to send.
2. Listen to carrier.
3. If available, send; else wait a random amount of time and goto 2.

Non-persistent does not wait constantly for an idle channel.

Advantage of non-persistent CSMA : Lower probability of collisions.

Disadvantage of non-persistent CSMA : Higher delay under light loading; you may be waiting for no reason (nobody else is going to transmit anyway).

Middle ground : p-persistent CSMA (Adjustable greedy) : Station listens to carrier; if it senses "empty" generates a random number and transmits with probability = p. With probability

$q = 1 - p$, station does not transmit, figuring the carrier is really busy, but the signal just hasn't reached this station yet.

Algorithm of *p*-persistent CSMA

1. Station has data to send.
2. Listen to carrier.
3. If idle, transmit with probability p; else wait until next time slot and goto 2.

All three of these strategies have higher utilization under heavy load than even slotted ALOHA but all except 1-persistent CSMA also have higher delay under light load.

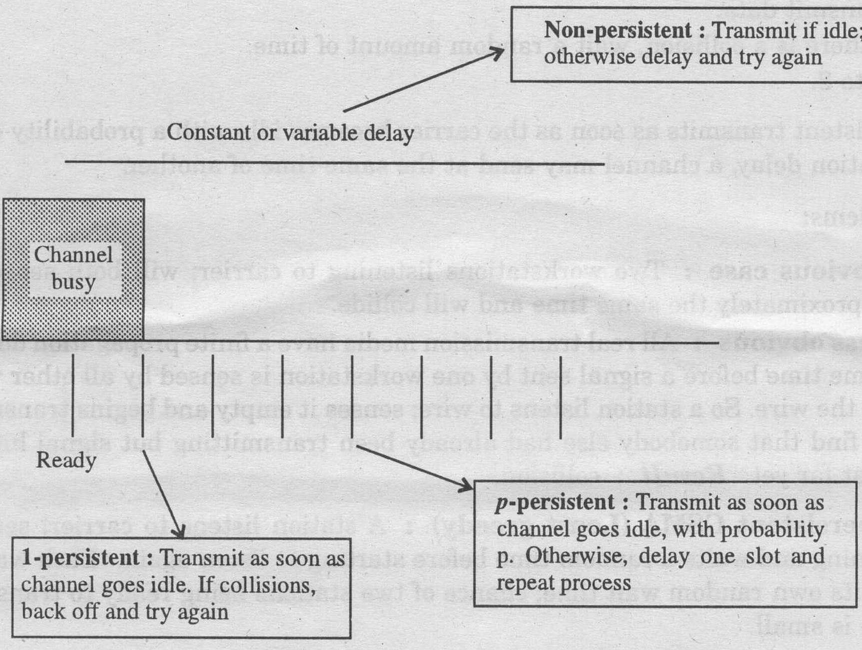

Non-persistent : Transmit if idle; otherwise delay and try again
Constant or variable delay
Channel busy
Ready
1-persistent : Transmit as soon as channel goes idle. If collisions, back off and try again
p-persistent : Transmit as soon as channel goes idle, with probability p. Otherwise, delay one slot and repeat process

Fig. 10.4 Comparison of CSMA strategies

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

CSMA has inefficiency that if a collision occurred, the channel is unstable until colliding packets have been fully transmitted. CSMA/CD overcomes this as follows :

While transmitting, the sender is listening to medium for collision. Sender stops if collision has occurred. So we can say that :

CSMA : Listen Before Talking.
CSMA/CD : Listen While Talking.

CSMA/CD is the method used in Ethernet networks for controlling access to the physical media by network nodes. CSMA is a network access method used on shared network topologies such as Ethernet to control access to the network. Devices attached to the network cable listen (carrier sense) before transmitting. If the channel is in use, devices wait before transmitting. MA (multiple access) indicates that many devices can connect to and share the same network. All devices have equal access to use the network when it is clear. Even though devices attempt to sense whether the network is in use, there is a good chance that two stations will attempt to access it at the same time. On large networks, the transmission time between one end of the cable and another is enough that one station may access the cable even though another has already just accessed it. There are two methods for avoiding these so-called collisions, listed here :

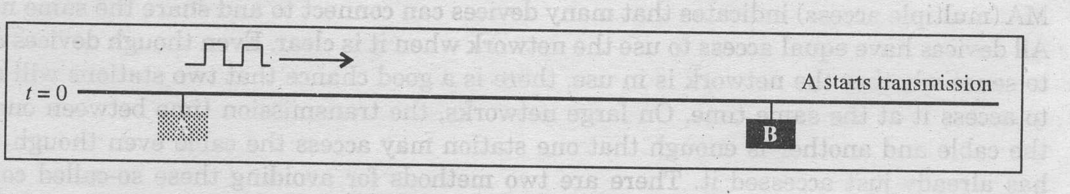
- **CSMA/CD (Carrier Sense Multiple Access/Collision Detection) :** CD (collision detection) defines what happens when two devices sense a clear channel, and then attempt to transmit at the same time. A collision occurs, and both devices stop transmission, wait for a random amount of time, and then retransmit. This is the technique used to access the 802.3 Ethernet network channel. This method handles collisions as they occur, but if the bus is constantly busy, collisions can occur so often that performance drops drastically. It is estimated that network traffic must be less than 40 percent of the bus capacity for the network to operate efficiently. If distances are long, time lags occur that may result in inappropriate carrier sensing, and hence collisions.
- **CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) :** In CA (collision avoidance), collisions are avoided because each node signals its intent to transmit before actually doing so. This method is not popular because it requires excessive overhead that reduces performance.

Collision Detection (CD)—It can be used to modify CSMA. Listen to the carrier, even as you're transmitting. When you detect a collision, stop transmitting (it's a waste of time to send the rest of the frame; it will be thrown away anyway). It saves wasted time, thus increasing utilization under heavy load.

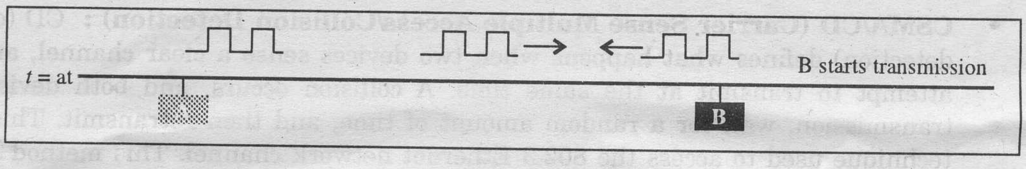
Collision strategy—Collisions can occur because 2 or more stations try to transmit simultaneously, or because one is already transmitting but its signal has not yet reached another station. When there is data waiting to be sent, each transmitting NIC also monitors its own transmission. If it observes a collision, it stops transmission immediately and instead transmits a 32-bit jam sequence. The purpose of this sequence is to ensure that any other node, which may currently be receiving this frame, will receive the jam signal in place of the correct 32-bit MAC CRC; this causes the other receivers to discard the frame due to a CRC error.

To ensure that all NICs start to receive a frame before the transmitting NIC has finished sending it, Ethernet defines a minimum frame size (i.e. no frame may have less than 46 bytes of payload). The minimum frame size is related to the distance, which the network spans, the type of media being used and the number of repeaters that the signal may have to pass through to reach the furthest part of the LAN. Together these define a value known as the Ethernet Slot Time, corresponding to 512 bit times at 10 Mbps.

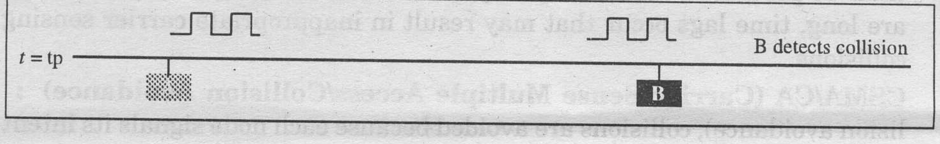
When two or more transmitting NICs each detect a corruption of their own data (i.e. a collision), each responds in the same way by transmitting the jam sequence. The following sequence depicts a collision :



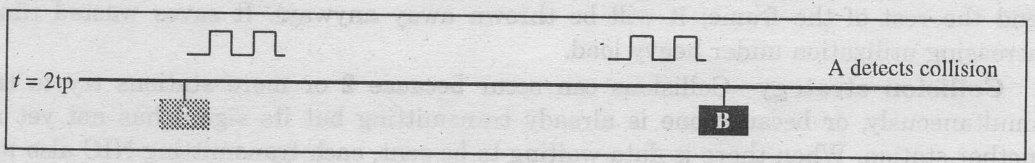
At time t=0, a frame is sent on the idle medium by NIC A.



A short time later, NIC B also transmits. (In this case, the medium, as observed by the NIC at B happens to be idle too).



After a period, equal to the propagation delay of the network, the NIC at B detects the other transmission from A, and is aware of a collision, but NIC A has not yet observed that NIC B was also transmitting. B continues to transmit, sending the Ethernet Jam sequence (32 bits).



After one complete round trip propagation time (twice the one way propagation delay), both NICs are aware of the collision. B will shortly cease transmission of the Jam Sequence, however A will continue to transmit a complete Jam Sequence. Finally the cable becomes idle.

Performance of CSMA / CD

It is simple to calculate the performance of a CSMA/CD network where only one node attempts to transmit at any time. In this case, the NIC may saturate the medium and near 100% utilization of the link may be achieved, providing almost 10 Mbps of throughput on a 10 Mbps LAN.

However, when two or more NICs attempt to transmit at the same time, the performance of Ethernet is less predictable. The fall in utilization and throughput occurs because some bandwidth is wasted by collisions and back-off delays. In practice, a busy shared 10 Mbps Ethernet network will typically supply 2-4 Mbps of throughput to the NICs connected to it.

As the level of utilization of the network increases, particularly if there are many NICs competing to share the bandwidth, an overload condition may occur. In this case, the throughput of Ethernet LANs reduces very considerably, and much of the capacity is wasted by the CSMA/CD algorithm, and very little is available for sending useful data. This is the reason why a shared Ethernet LAN should not connect more than 1024 computers. Many engineers use a threshold of 40% Utilization to determine if a LAN is overloaded. A LAN with a higher utilization will observe a high collision rate, and likely a very variable transmission time (due to back off). Separating the LAN in to two or more collision domains using bridges or switches would likely provide a significant benefit (assuming appropriate positioning of the bridges or switches).

Salient features of CSMA with Collision Detection

- When 2 stations transmit and the collision is detected, the stations stop immediately.
- This saves bandwidth and time.
- Used in 802.3 and Ethernet.

10.5 Collision Free Protocols

While collisions do not occur with CSMA when a station is controlling the channel, they do occur during the contention period between one station releasing the channel and the next claiming it. These collisions affect system performance, especially when the cable is long and the frames are short. Collision free protocols aim to resolve the contention issues for the channel, and so result in no collisions at all.

Collision-free protocols divide the users into disjoint groups, such that a collision is impossible. An example of a collision-free protocol is the TDM protocol in which each user has a dedicated time slot. Collision-free protocols have high delay at low load, but the channel efficiency at high load is much better than that of contention protocols. The simplest collision-free methods are TDM and FDM. These allocation schemes are efficient when the number of users is small, and the traffic is continuous. Under conditions of bursty traffic, or when the number of users is large and variable, collision-free protocols are poor choices.

"Contention slots"—the time intervals spent deciding which station gets to transmit next.

There are two types of collision free protocols that are:

1. A Bit-Map Protocol
2. Binary Countdown

10.5.1 A Bit-Map Protocol

It is a Linear Search Approach as the channel is divided into reservation period and data transfer period. Each contention period consist of exactly *N* slots. If station *X* has a frame to send, it transmits a 1 bit during the *X*th slot. The stations are only allowed to transmit during their allocated slot. Regardless of what station *X* does, station *X* + 1 gets the opportunity to transmit a 1 during slot *X* + 1, and so on (But it should only do so if it has some data queued, ready to be sent). An algorithm is then used to decide which station (out of those requesting the channel) is given access to it, and the data is then sent.

Simplest strategy : When one station finishes sending, each station on the network (in some pre-defined numerical order) broadcasts a "1" if it wants to transmit and a "0" otherwise. When everybody has done this, the stations transmit one at a time in numerical order. When everybody who wanted to send has done so, you start over.

Example :

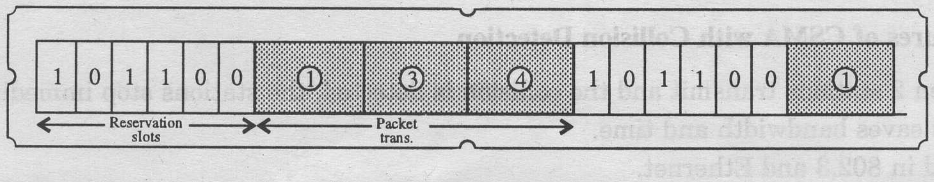


Fig. 10.5 Bit-map protocol

Here we have 6 users. Each user has a designated reservation slots. It favors the high numbered stations.

10.5.2 Binary Countdown

The problem with the basic bitmap protocol is that the overhead is 1 bit per station, so it does not scale well to networks with thousands of stations. Binary station addresses address this. A station wanting to use the channel broadcasts its address as a binary bit string, starting with the high-order bit. All addresses are assumed to be the same length. The bits in each position from different stations are 'ORed' together. As soon as a station attempting to transmit detects a bit which it did not send, it stops attempting to claim the channel; thus the station with the highest number is granted access to the channel. The following figure explains this protocol:

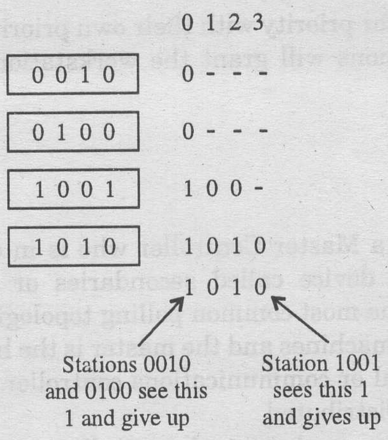

0 1 2 3
0 0 1 0
0 - - -
0 1 0 0
0 - - -
1 0 0 1
1 0 0 -
1 0 1 0
1 0 1 0
1 0 1 0
Stations 0010 and 0100 see this 1 and give up
Station 1001 sees this 1 and gives up

Fig. 10.6 Binary Countdown

Different, more complex algorithms include:

BRAP (Broadcast Recognition with Alternating Priorities)—As soon as a station inserts its "1" indicating it wants to talk, it starts sending-start the cycle of 1 and 0 with the next station, rather than the first each time.

MLMA (Multi-level Multi-Access)—Stations wanting to transmit contend using the "digits" in their station number, going from most-significant to least-significant-improves delay under light loading from the BRAP results

Adaptive Tree Walk protocol—Basically, a binary search over the set of stations. Organize the stations in a binary tree, and do a depth-first search to determine who gets to talk first.

10.6 Token and Token-Passing Access Methods

A token is a special control frame on token ring, token bus, and FDDI (Fiber Distributed Data Interface) networks that determines which stations can transmit data on a shared network. The node that has the token can transmit. Unlike contention-based networks, such as Ethernet, workstations on token-based networks do not compete for access to the network. Only the station that obtains the token can transmit. Other stations wait for the token rather than trying to access the network on their own. On Ethernet networks, "collisions" occur when two or more workstations attempt to access the network at the same time. They must back off and try again later, which reduces performance-especially as the number of workstations attached to a network segment increases.

In token ring networks, a station takes possession of a token and changes one bit, converting the token to a SFS (start-of-frame sequence). A field exists in the token in which workstations can indicate the type of priority required for the transmission. The priority setting is basically a request to other stations for future use of the token. The other stations compare a workstation's request for priority with their own priority levels. If the workstation's priority is higher, the other stations will grant the workstation access to the token for an extended period.

10.7 Polling Systems

Polling designates one device as a Master Controller who is in charge of who can transmit. The master queries each other device called secondaries or slaves to see if they have information to transmit. One of the most common polling topologies is a star where the points of the star are secondary or slave machines and the master is the hub. This is a popular method to connect terminals to a terminal or communications controller. There are two categories of polling systems: Centralized or Distributed

Centralized polling systems : A central controller transmits polling messages to stations in a certain order.

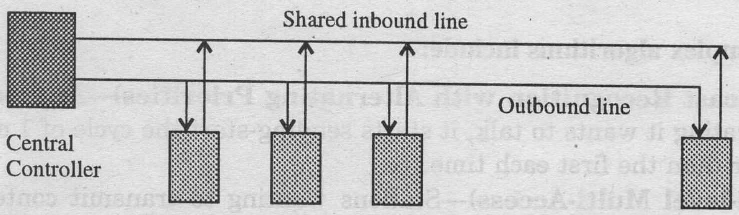


Fig. 10.7 Centralized polling systems

Distributed reservation systems : The permit for packet transmission is passed from a station to another in a certain order.

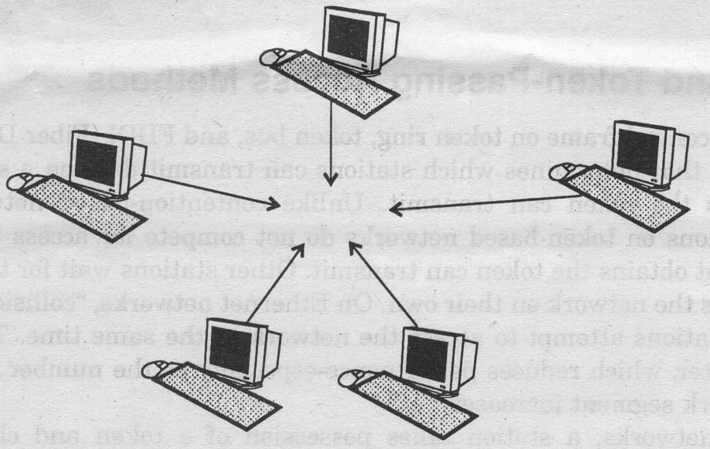


Fig. 10.8 Distributed reservation systems

Efficiency of general polling systems

Unlike the case of random access or reservation systems, the efficiency of a polling system can be arbitrarily close to 1.

Assumption : No bound for the number of packets that can be transmitted over a polling period.

What happens? The portion of walk times in a polling cycle becomes negligible when the system is heavily loaded and the polling cycle becomes extremely long as a result.

In practice, as the system load approaches 1, the length of polling cycles increases very fast, which in turn increase delays. The number of packets that a station can transmit over a single polling cycle is limited.

● ● ●

11

Introduction To LAN And MAN Protocols

Introduction

This chapter introduces the various LAN protocols and media-access methods used in a local-area network (LAN). Topics addressed focus on the methods and devices used in Ethernet/IEEE 802.3, Token Bus/IEEE 802.4, Token Ring/IEEE 802.5, CSMA/CA medium access protocol (IEEE 802.11), a wireless LAN protocol and Fiber Distributed Data Interface (FDDI). Also in this chapter we will discuss DQDB, a protocol designed to be used in MANs.

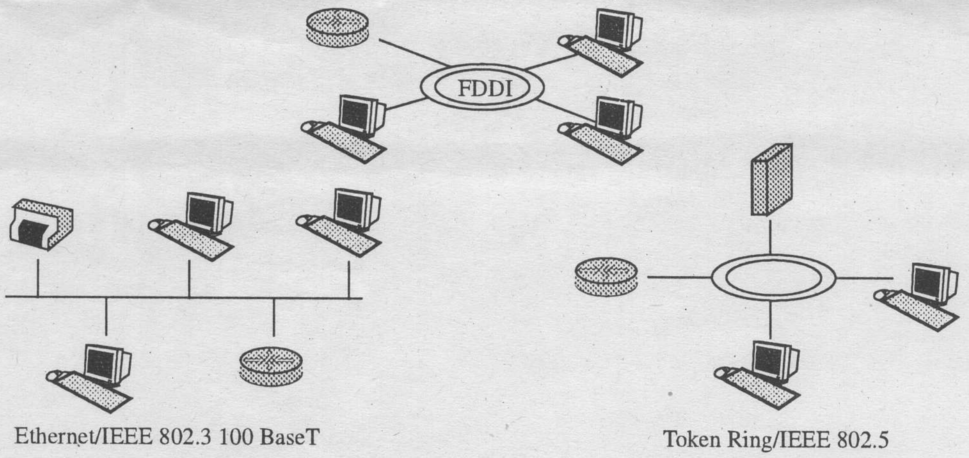


Fig. 11.1 Three LAN implementations used most commonly

What is a LAN?

A LAN is a high-speed data network that covers a relatively small geographic area. It typically connects workstations, personal computers, printers, servers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communication between users via electronic mail and other applications.

Definition

A local area network (LAN) is:
- *A data communications medium.*
- *Characterized by having all sites on the same link.*
- *Confined to a small area, does not cross public right of way.*
- *Characterized by relatively high data rates.*

11.1 IEEE 802 Standards

The IEEE (Institute of Electrical and Electronic Engineers) is a technical association of industry professionals with a common interest in advancing all communications technologies. The IEEE or Institute of Electrical and Electronics Engineers define industry-wide standards that promote the use and implementation of technology. These committees are named for the standards they publish. The IEEE 802 committee sets the standards for networking. However, this committee is split into several smaller sub-committees such as IEEE 802.3 and IEEE 802.5. In general, IEEE 802 standards define physical network interfaces such as network interface cards, bridges, routers, connectors, cables, and all the signaling and access methods associated with physical network connections.

The 802 standards describe the three lowest layers of the network architecture:

- **Physical layer (layer 1)**—Describes the transmission medium, type of electrical signals used and how nodes are attached to the network (cables and connectors).
- **Medium Access Control (MAC) sublayer (layer 2)**—Implements methods that control a node's access to the shared transmission medium.
- **Data link layer (layer 3)**—Provides for reliable transmission (including error checking), delineates beginning and ending of frames (headers and trailers).

These standard LANs differ in their physical layer and their MAC sublayer but are compatible at their data link layer.

A set of network standards developed by the IEEE include:

- 802.1 Higher Layer LAN Protocols Working Group.
- 802.2 Logical Link Control Working Group.
- 802.3 Ethernet Working Group.

- 802.4 Token Bus Working Group.
- 802.5 Token Ring Working Group.
- 802.6 Metropolitan Area Network Working Group.
- 802.7 Broadband TAG.
- 802.8 Fiber Optic TAG.
- 802.9 Isochronous LAN Working Group.
- 802.10 Security Working Group.
- 802.11 Wireless LAN Working Group.
- 802.12 Demand Priority Working Group.
- 802.14 Cable Modem Working Group.
- 802.15 Wireless Personal Area Network (WPAN) Working Group.
- 802.16 Broadband Wireless Access Working Group.
- 802.17 Resilient Packet Ring Working Group.

Following are the some standards that we will discuss in this chapter:

IEEE 802.1 : Standards related to network management. Defines routing, bridging, and internetwork communications

IEEE 802.2 : General standard for the data link layer in the OSI Reference Model. Allows Network layer protocols to link to Physical layer and MAC sublayer protocols

IEEE 802.3 : Defines the MAC layer for bus networks that use CSMA/CD. This is the basis of the Ethernet standard.

IEEE 802.4 : Defines the MAC layer for bus networks that use a token-passing mechanism (token bus networks).

IEEE 802.5 : Defines the MAC layer for token-ring networks.

IEEE 802.6 : Standard for Metropolitan Area Networks (MANs).

LAN Protocols and the OSI Reference Model

LAN protocols function at the lowest two layers of the OSI reference model, between the

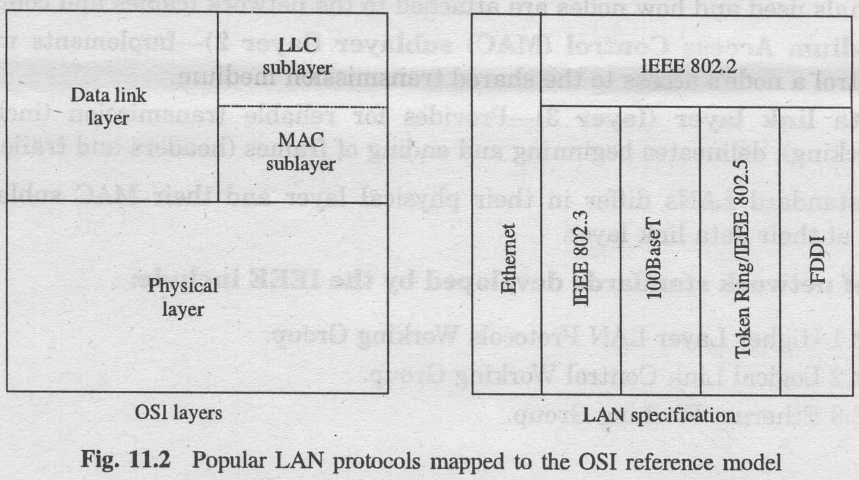


Fig. 11.2 Popular LAN protocols mapped to the OSI reference model

physical layer and the data link layer. Figure 11.2 illustrates how several popular LAN protocols map to the OSI reference model.

LAN Transmission Methods

LAN data transmissions fall into three classifications:

- Unicast.
- Multicast.
- Broadcast.

In each type of transmission, a single packet is sent to one or more nodes.

In a **unicast transmission**, a single packet is sent from the source to a destination on a network. First, the source node addresses the packet by using the address of the destination node. The package is then sent onto the network and finally, the network passes the packet to its destination.

A **multicast transmission** consists of a single data packet that is copied and sent to a specific subset of nodes on the network. First, the source node addresses the packet by using a multicast address. The packet is then sent into the network, which makes copies of the packet and sends a copy to each node that is part of the multicast address.

A **broadcast transmission** consists of a single data packet that is copied and sent to all nodes on the network. In these types of transmissions, the source node addresses the packet by using the broadcast address. The packet is then sent on to the network, which makes copies of the packet and sends a copy to every node on the network.

11.2 Higher Layer LAN Protocols (IEEE 802.1)

The IEEE 802.1 Working Group is chartered to concern itself with and develop standards and recommended practices in the following areas: 802 LAN/MAN architecture, internetworking among 802 LANs, MANs and other wide area networks, 802 Link Security, 802 overall network management, and protocol layers above the MAC & LLC layers.

The 802.1 working group has two active task groups : Internetworking and Link Security.

11.3 LLC : Logic Link Control (IEEE 802.2)

Logic Link Control (LLC) is the IEEE 802.2 LAN protocol that specifies an implementation of the LLC sublayer of the data link layer. IEEE 802.2 LLC is used in IEEE 802.3 (Ethernet) and IEEE 802.5 (Token Ring) LANs to perform these functions:

(a) Managing the data-link communication
(b) Link Addressing
(c) Defining Service Access Points (SAPs)
(d) Sequencing

LLC was originated from the High-Level Data-Link Control (HDLC) and it uses a subclass of the HDLC specification. LLC defines three types of operation for data communication:

Type 1 : Connectionless—The connectionless operation is basically sending but no guarantee of receiving.

Type 2 : Connection Oriented—The connection-oriented operation for the LLC layer provides these 4 services: Connection establishment; Confirmation and acknowledgement that data has been received; Error recovery by requesting received bad data to be resent; Sliding Windows, which is a method of increasing the rate of data transfer.

Type 3 : Acknowledgement with connectionless service.

The Type 1 connectionless service of LLC specifies a static-frame format and allows network protocols to run on it. Network protocols that fully implement a transport layer will generally use Type 1 service. The Type 2 connection-oriented service of LLC provides reliable data transfer. It is used in LAN environment that do not invoke network and transport layer protocols.

Protocol Structure—LLC : Logic Link Control (IEEE 802.2)

Recall from the OSI Model module that all packets deal with:

- What is being sent.
- Where is it being sent.

We will describe all packet structures in terms of these two parameters. In the case of the LLC PDU (Protocol Data Unit) the data field is what is being sent, and the source and destination access points (SSAP and DSAP) describe where it is being sent in terms of the multiplexing peer protocol.

Logic Link Control Layer (LLC) Header

8	16	24 or 32bits	Variable
DSAP	SSAP	Control	Data or LLC information

Fig. 11.3 LCC header

The various fields of LLC header are stated below:

- **Destination Service Access Point (DSAP)**—The DSAP is a seven-bit address with an eighth bit to indicate if it is a specific address (0) or a group (broadcast) address (1). The DSAP is not a station or device address; rather it designates the service control point where the message should be routed.
- **Source Service Access Point (SSAP)**—The SSAP is also a seven-bit address, but in this case the eighth bit is used to indicate if the message is a command (0) or a response (1). Like the DSAP, the SSAP designates a control point and not a station address. In the case of the SSAP, this is the control point from which the message originated.
- **Control**—The control field is either 8 or 16 bits long, with the length indicated by the

first two bits. The 16-bit fields are used to exchange sequence numbers, while the 8-bit variation is used for unsequenced information.

- **LLC information**—LLC data or higher layer protocols.

Features of IEEE Standard 802.2 : Logical Link Control

- Runs on top of 802.3-802.5 and provides error control and flow control.
- Hides differences among the 802 protocols and presents a single interface to the network layer.
- Three service options :
  - ☐ Unreliable datagram service.
  - ☐ Acknowledged datagram service.
  - ☐ Reliable connection-oriented service.

Below the 802.2 LLC are the MACs for the various physical LAN implementations. These standards are known as 802.3 for CSMA/CD, 802.4 for token passing bus, and 802.5 for token passing ring.

11.4 Ethernet (IEEE 802.3)

Ethernet is a networking technology defined by the Institute for Electrical and Electronic Engineers (IEEE) as IEEE standard of 802.3. This Physical Layer technology is the most popular Data Link Layer protocol because of its speed, lower cost, and worldwide acceptance. Ethernet can support nearly every protocol in use today, and can operate with any networking equipment that adheres to the IEEE standard. This openness, combined with the ease of use, has made Ethernet dominant in the local area network arena.

In the Ethernet standard, there are two modes of operation: half-duplex and full-duplex modes. In the half duplex mode, data are transmitted using the popular Carrier-Sense Multiple Access/Collision Detection (CSMA/CD) protocol on a shared medium. The main disadvantages of the half-duplex are the efficiency and distance limitation, in which the link distance is limited by the minimum MAC frame size. This restriction reduces the efficiency drastically for high-rate transmission. Therefore, the carrier extension technique is used to ensure the minimum frame size of 512 bytes in Gigabit Ethernet to achieve a reasonable link distance.

The Ethernet system consists of three basic elements:

1. The physical medium used to carry Ethernet signals between computers.
2. A set of medium access control rules embedded in each Ethernet interface that allow multiple computers to fairly arbitrate access to the shared Ethernet channel.
3. An Ethernet frame that consists of a standardized set of bits used to carry data over the system.

11.4.1 General Ethernet Features

There are a number of adaptations to the IEEE 802.3 Ethernet standard, including adaptations with data rates of 10 Mbits/sec; 100 Mbits/sec (Fast Ethernet); 1,000 Mbits/sec (Gigabit Ethernet); and, most recently, 10 Gigabit/sec Ethernet. The original forms of Ethernet used coaxial cable, but today most Ethernet networks are connected with Category 5 twisted-pair cable. Gigabit Ethernet and 10 Gigabit Ethernet run on fiber optic cables and are being deployed in metropolitan area networks as "metro Ethernets".

- **Shared media :** All nodes attached to the network take turns using the media. If the network is not busy, a shared scheme is very efficient; but if the network gets busy, nodes must wait to transmit or, worse, collisions occur that reduce performance.
- **Broadcast domains :** Just like a TV broadcast, transmitted frames are "heard" by all, but nodes receive only the frames that are addressed to them. Special broadcast frames are addressed to all nodes. These are used for multicasting and network maintenance.
- **CSMA/CD (Carrier Sense Multiple Access/Collision Detection) access method:** On a shared media, some method is required to prevent two or more nodes from transmitting at the same time, and to detect "collisions" that occur if two stations begin transmitting at the same time.
- **Collision domains :** A collision domain consists of a set of networked nodes that share the same medium and that contend for access to the cable. In other words, their transmissions can collide. Suppose you have a busy 50-node network, and the number of collisions is reducing performance. You divide the network with a bridge to create two 25-node networks. Now you have two separate collision domains and one broadcast domain (the bridge will propagate broadcasts and appropriate traffic from one segment to the other).
- **Framing method :** Several framing methods were specified early on, but most systems now conform to IEEE 802.3 framing. More recently, the framing method was changed to support higher data rates (Gigabit Ethernet). The 802.3 frame is discussed later.
- **Full-duplex mode :** The twisted-pair Ethernet adaptations support full-duplex mode, in which signals are transmitted on one wire and received on another. Coaxial cable does not support full-duplex mode. Since both end stations are transmitting on separate wires, there is no contention (in a switched environment). This greatly improves transmission speeds.
- **Cable types :** The earliest Ethernet adaptations support coaxial cable in a bus topology configuration (the cable runs from one node to the next). Later versions supported twisted-pair cable in a star configuration (each node is individually wired to a hub). Combinations of cabling schemes are possible.
- **Cable length :** There are cable-length limitations, depending on the media type. Overextended networks are subject to signal propagation delays that can cause a failure in the collision detection mechanism. When a cable is too long, nodes at far ends may not detect that another has begun transmission. Collisions will go undetected, and there will be continuous network errors until the cable is shortened and the end stations are able to detect one another's transmissions.