Chapter 3

The Data Link Layer

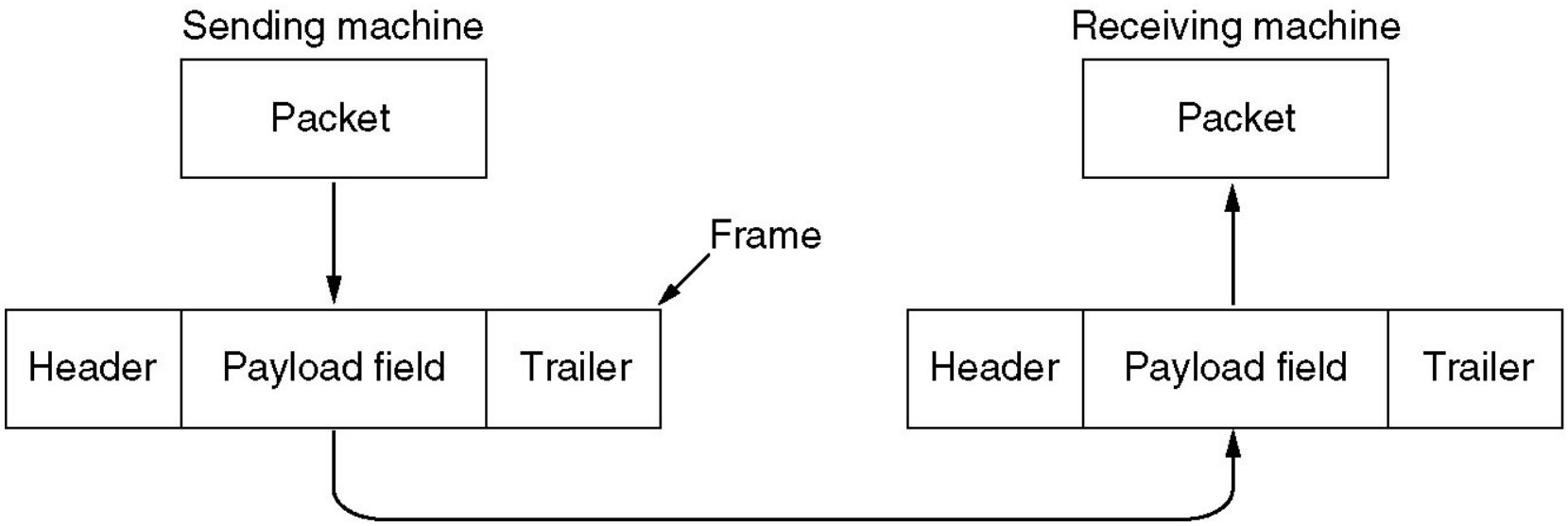
Data Link Layer Design Issues

- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

Functions of the Data Link Layer

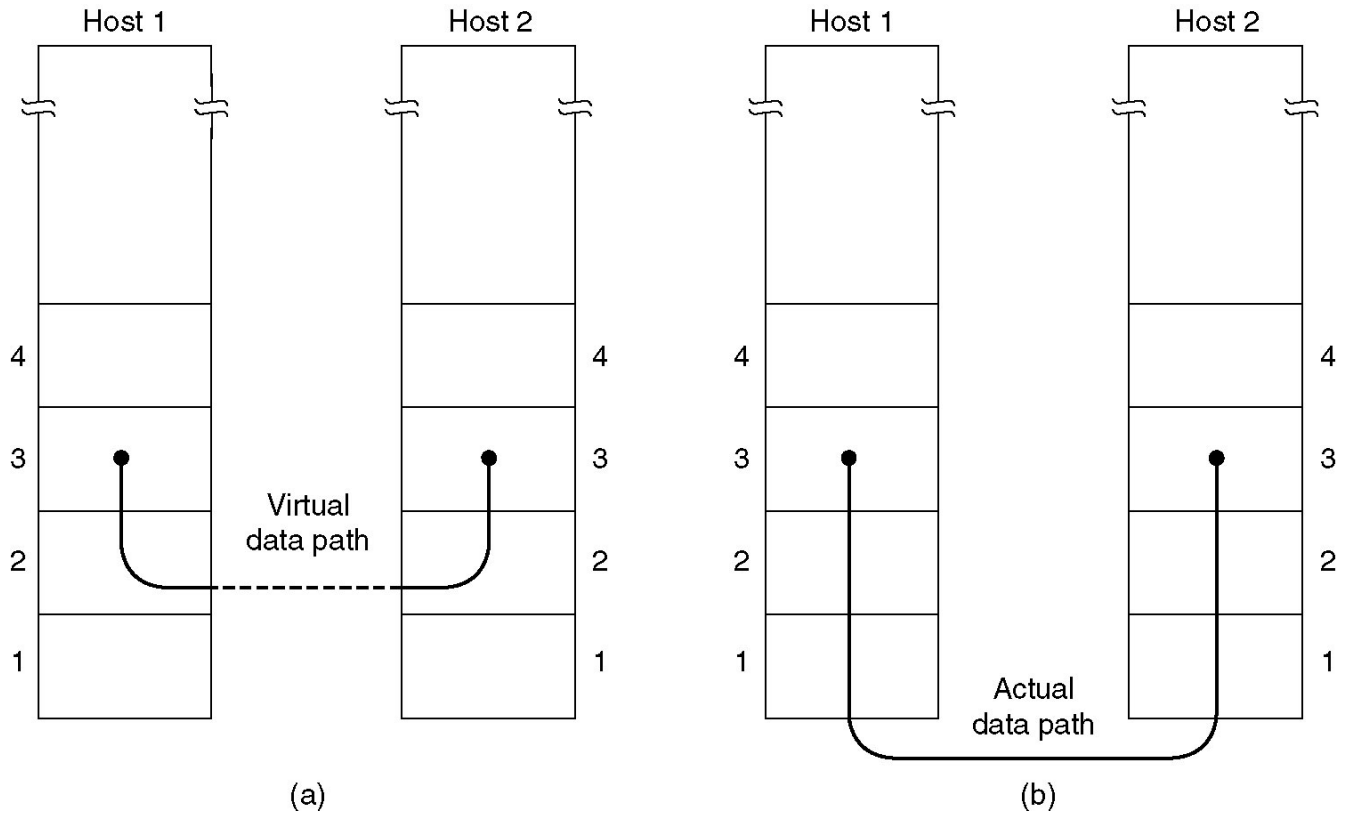
- Provide service interface to the network layer
- Dealing with transmission errors
- Regulating data flow
  - Slow receivers not swamped by fast senders

Functions of the Data Link Layer (2)



Relationship between packets and frames.

Services Provided to Network Layer



(a) Virtual communication.
(b) Actual communication.

Services Provided to Network Layer(2)

1) Unacknowledged connectionless service
2) Acknowledged connectionless service
3) Acknowledged connection oriented service

Services Provided to Network Layer(3)

1) Unacknowledged connectionless service

- The source machine send independent frames to the destination machine without having the destination machine acknowledge them;

- no logical connection is established;

- appropriate for real time traffic (voice,..) when the error rate is very low;

- most LAN's used unacknowledged connectionless service in the DLL;

Services Provided to Network Layer(4)

2) Acknowledged connectionless service

- There is still no logical connection used but each frame sent is individually acknowledged;

- the source knows whether a frame has arrived correctly and if it has not arrived within a specified time interval, it can be sent again;

- the service is *useful over unreliable channels* such as wireless systems;

OBS

Providing acknowledgments in the DLL is an optimization never a requirement

Services Provided to Network Layer(5)

3) Acknowledged connection-oriented service

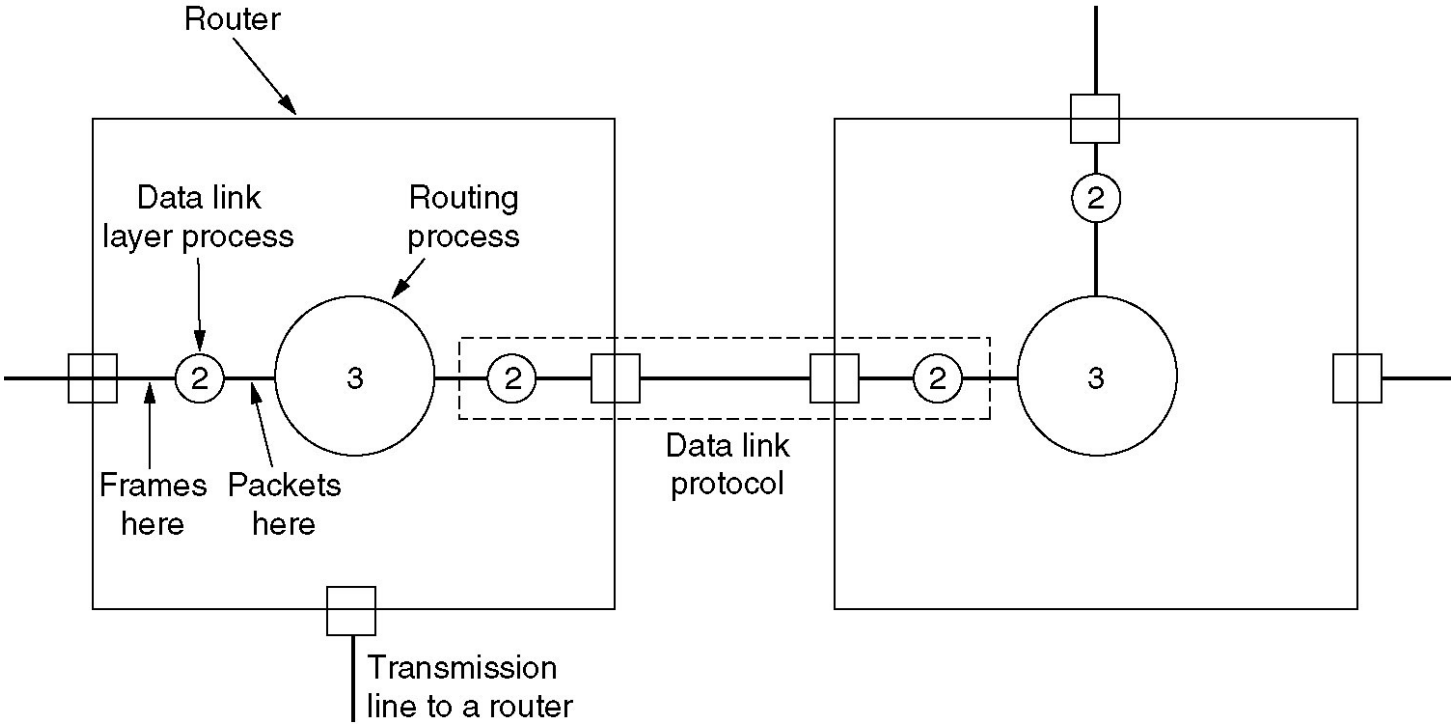
- the source and destination machines establish a connection before any data are transferred;
- each frame sent is numbered and the DLL guarantees that:
  - it is indeed received;
  - it is received once and all the frames are received in the right order;
- provides the network layer processes with the equivalent of a reliable stream;
- transfer phases:
  - connection established;
  - frames actually transmitted;
  - connection released;

Services Provided to Network Layer(6)

3) Acknowledged connection-oriented service

Example: A WAN subnet consisting of routers connected by point-to point leased telephone lines;

Services Provided to Network Layer (7)



Placement of the data link protocol.

Framing (incadrarea)

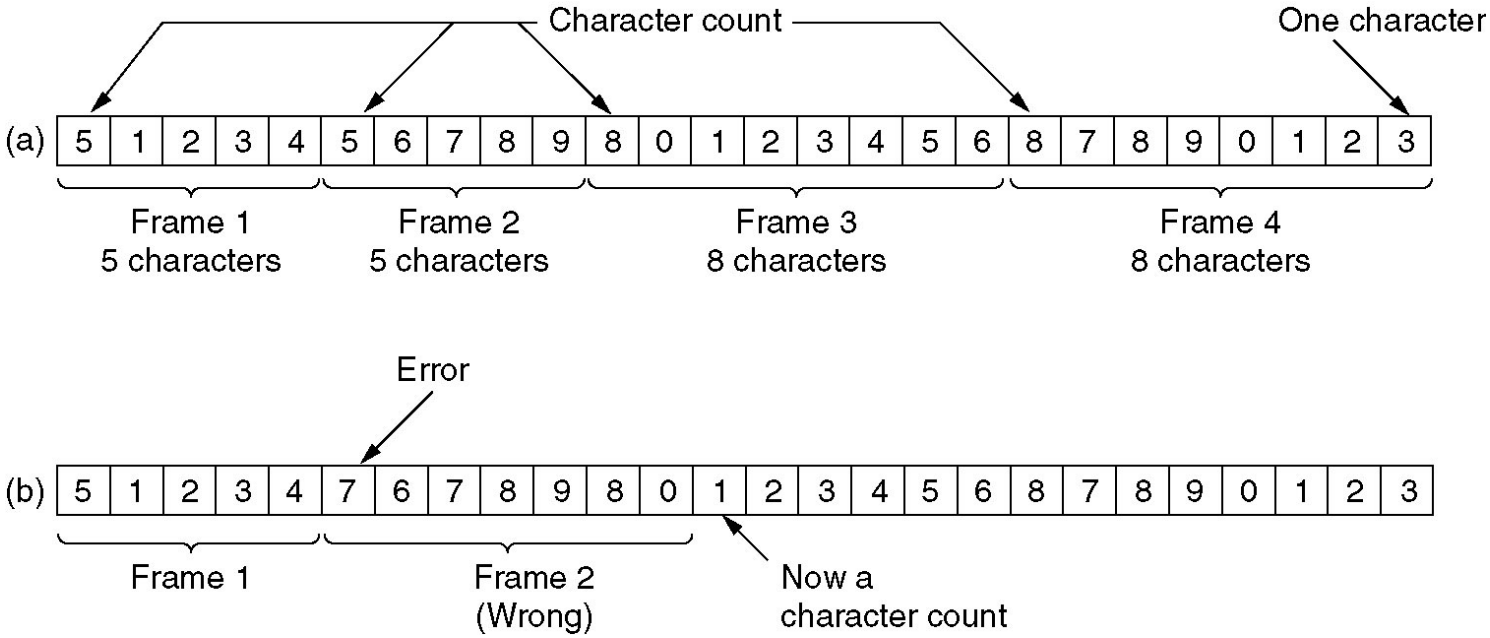
The DLL

- breaks the bit streams into discrete frames
- compute the ***checksum*** for each frame

Methods for breaking the bit stream

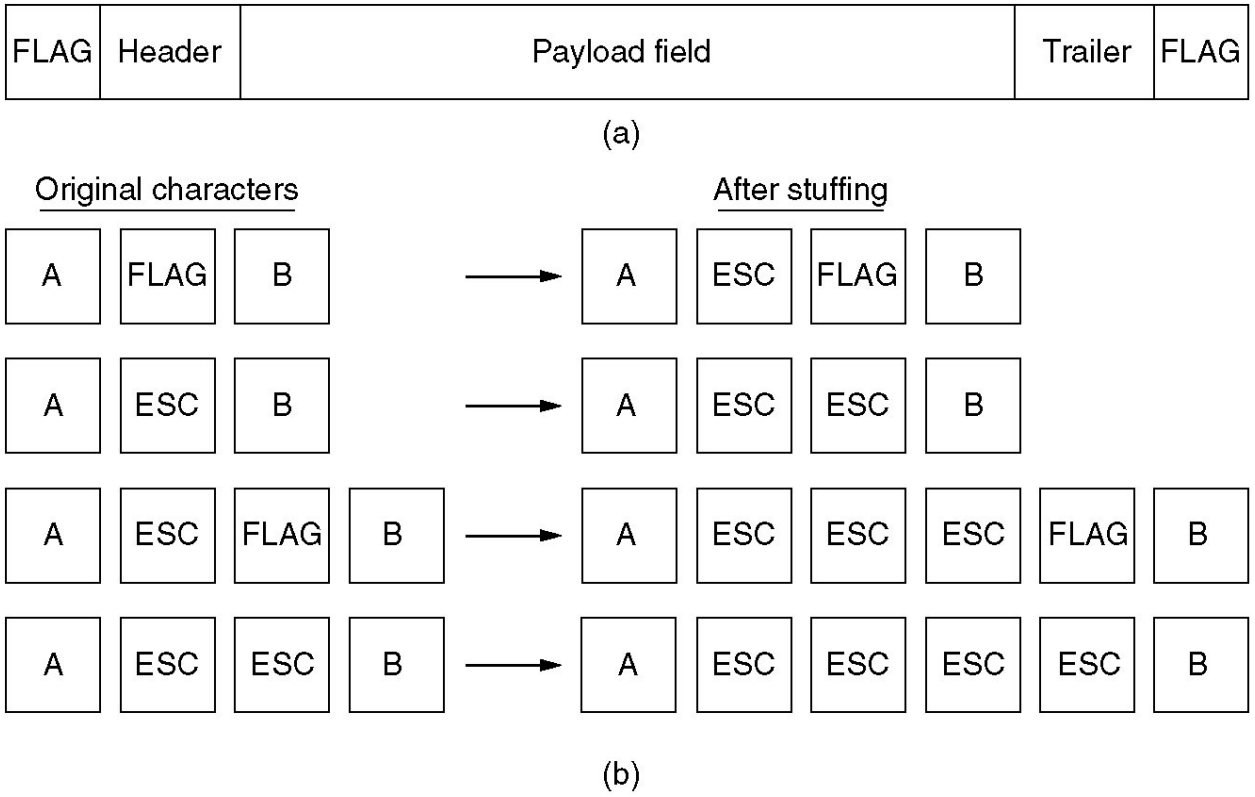
- character count;
- fly bites with byte stuffing;
- starting and ending flags with bit stuffing
- physical layer coding violations

Framing(2)- Character count



A character stream. (a) Without errors. (b) With one error.

Framing (3)- Flag bytes with byte stuffing



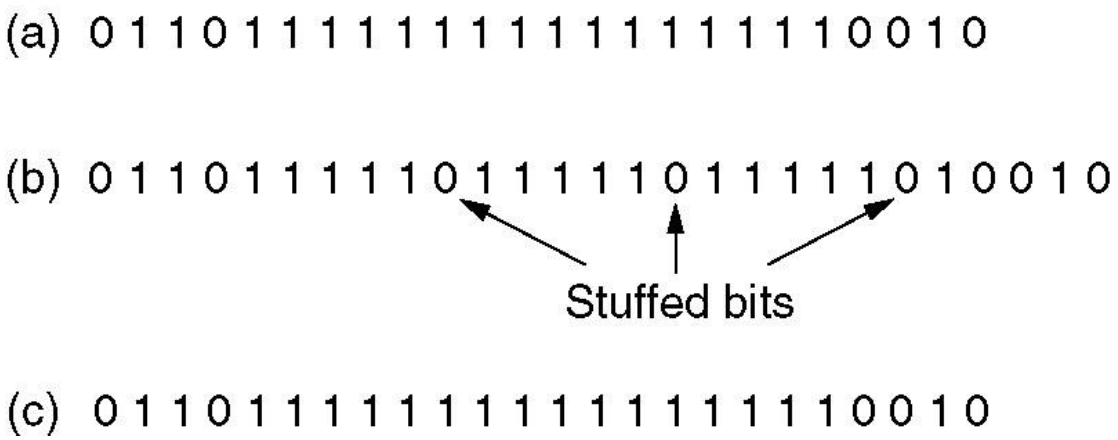
(a) A frame delimited by flag bytes.

(b) Four examples of byte sequences before and after stuffing.

Framing (4) – Starting and ending flags with bit stuffing

Each frame begins and end with a special bit pattern 01111110

01111110 *is transmitted* 011111010



Bit stuffing
(a) The original data.
(b) The data as they appear on the line.
(c) The data as they are stored in receiver's memory after de-stuffing.

Error control

In the connection- oriented service networks the usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line;

- the protocols calls for the receiver to send back special control frames bearing positive or negative acknowledgments about the incoming frames;

    - a positive ack. means: the frame has arrived safely;

- introducing timers into the data link layer;

    - if either the frame or the ack is lost the timer will go off alerting the sender to a potential problem;

    - solution is to transmit the frame again;

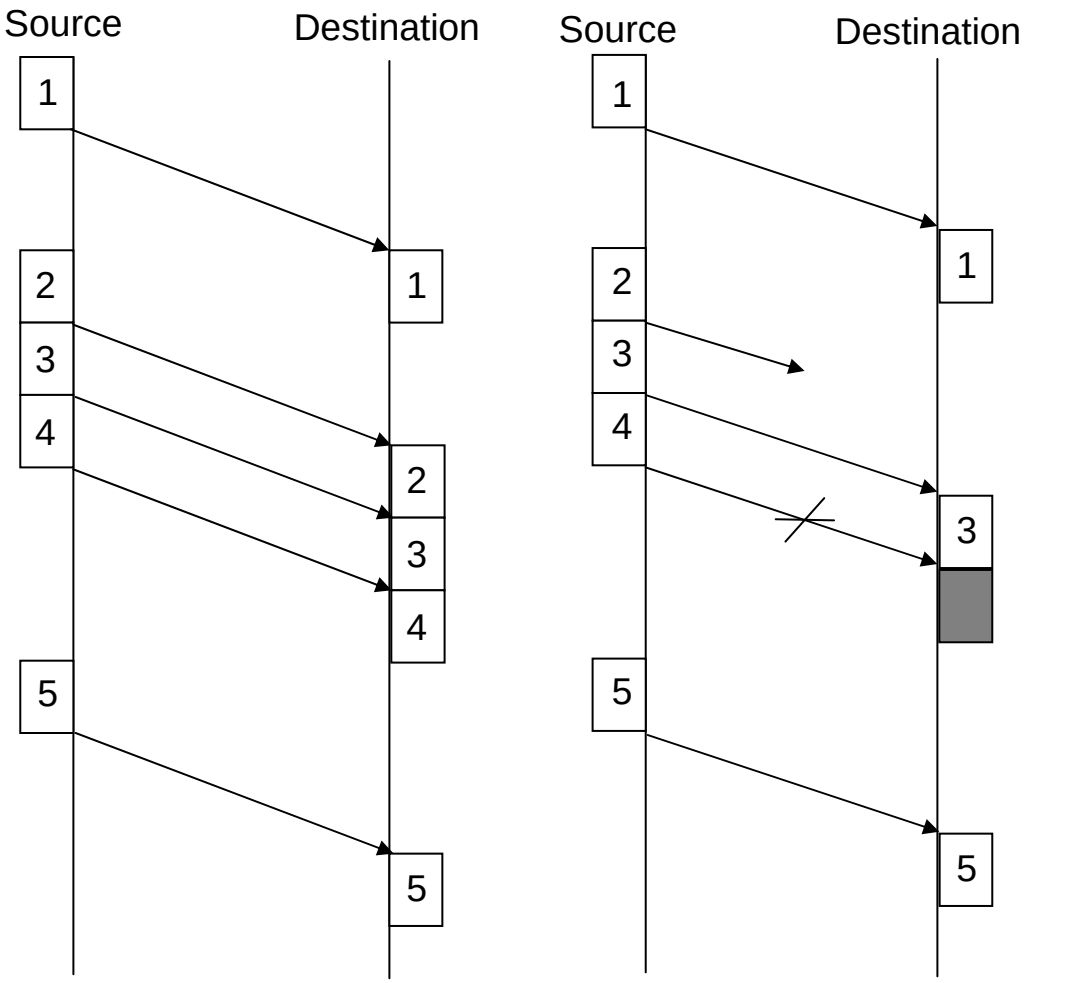
    - to assign sequence numbers to outgoing frames.

Flow control

Approaches used:

- feed-back based flow control: the receiver sends back information to the user giving it permission to send more data;
- rate-based flow control: the protocol has a built-in mechanism that limits the rate at which senders may transmit data without using feedback from the receiver

Flow control techniques



(a) Error-free transmission

(b) Transmission with errors and losses

Flow control techniques(2)

Stop-and-wait flow control:

- source entity transmits a frame;
- after reception, the destination indicates its willingness to accept another frame by sending back an ack. to the frame just received;
- the source must wait until it receive the ack.;
- the destination can stop the flow by simply with holding ack.

OBS

With the use of multiple frames for a single message the stop-and-wait procedure may be inadequate.

The essence of the problem is that only one frame at a time can be in transit.

Flow control techniques(3)

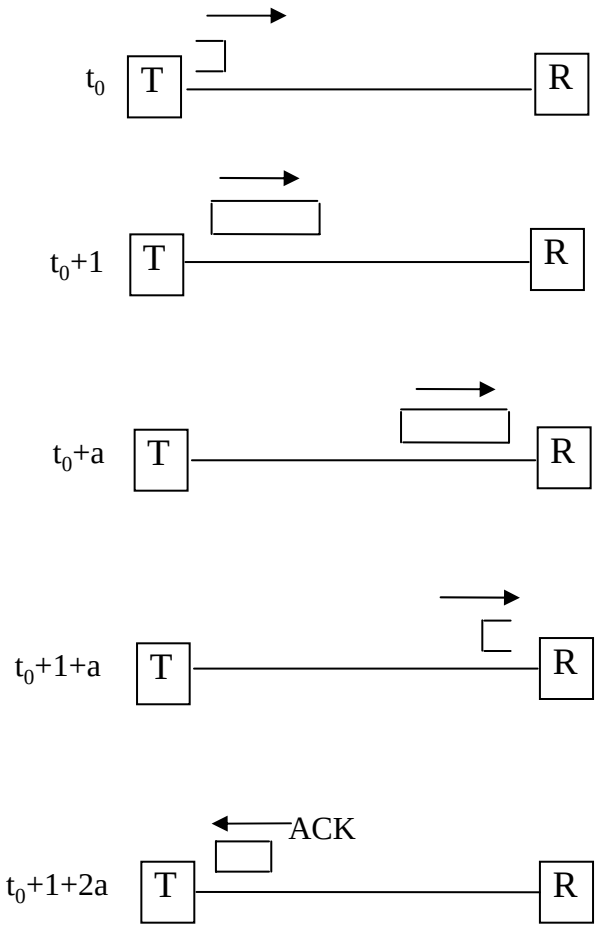
1 = the transmission time (the time it takes for a station to transmit a frame)

a = the propagation delay (the time it takes for a bit to travel from sender to receiver)

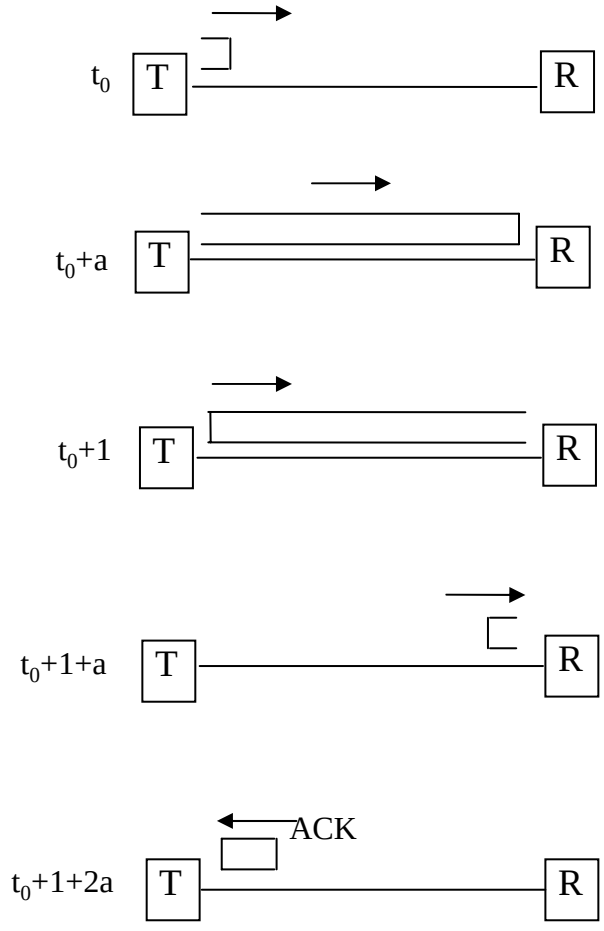
a < 1 – the frame is sufficient long that the fist bits have arrived at the destination before the source has completed the transmission of the frame;

a >1 – the sender completes the transmission of the entire frame before the leading bits of the frame arrives at the receiver;

Flow control techniques(4)



(a) a>1



(b) a<1

Flow control techniques(5)

Conclusion stop-and-wait flow control

1. larger values of a are consistent with higher data rates and/or longer distances between stations
2. For very high data rates and long distances between sender and receiver stop-and-wait flow control provide inefficient utilization of the link.
3. Efficiency can be greatly improved by allowing multiple frames to be in transit at the same time.

Flow control techniques(6)

Sliding window flow control

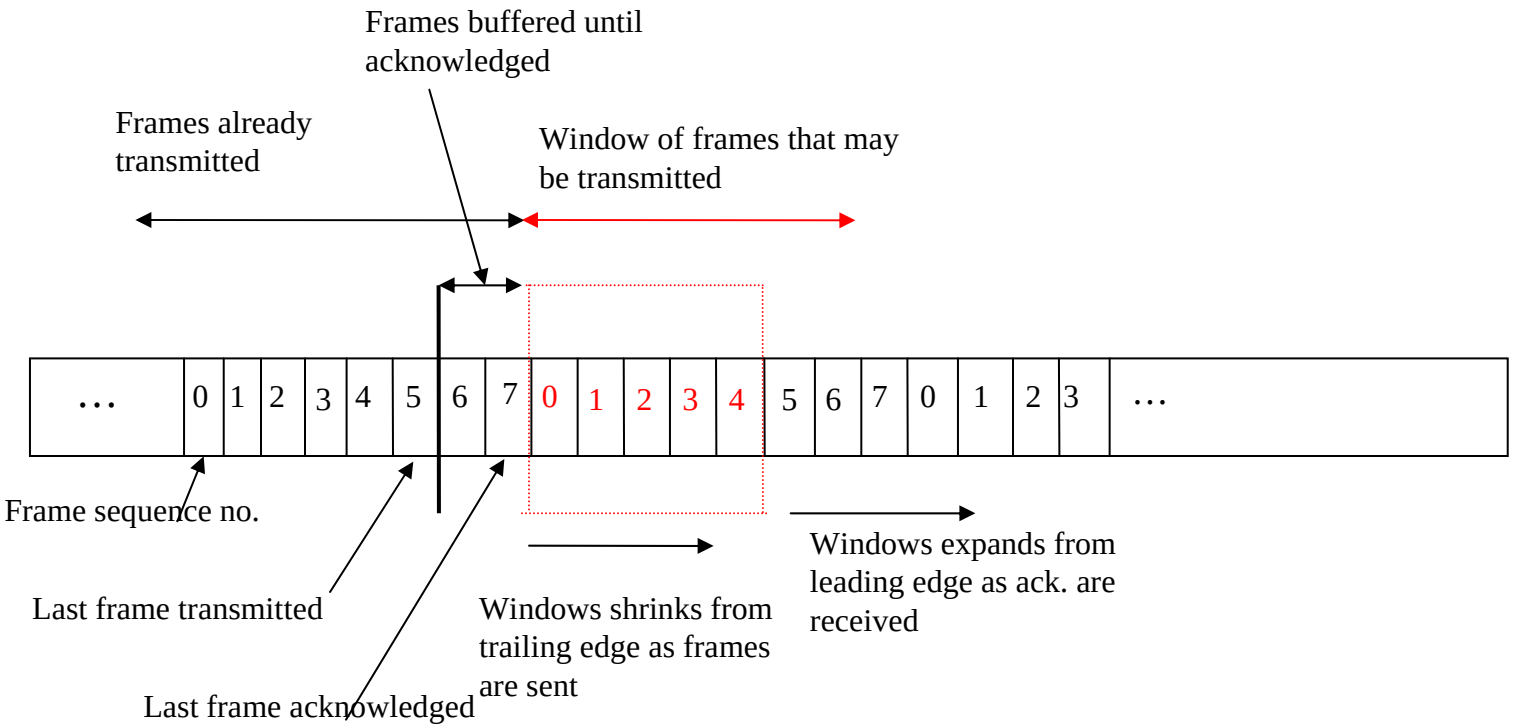
Source system A	Destination system B
- A is allowed to sent ***n frames*** without waiting for ACK	- B allocates buffer space for n frames, thus B can accept ***n frames***
- Each frame is labeled with a sequence number	- B acknowledges frames by sending back an ACK that includes the sequence no of ***the next frame*** expected
	-This ACK also implicitly announces that B is prepared to receive the ***next n frames*** beginning with the number specified
- A maintains a list of sequence numbers that it is allowed to send	-B maintains a list of sequence numbers that it is prepared to receive

Flow control techniques(7)

Sliding window flow control

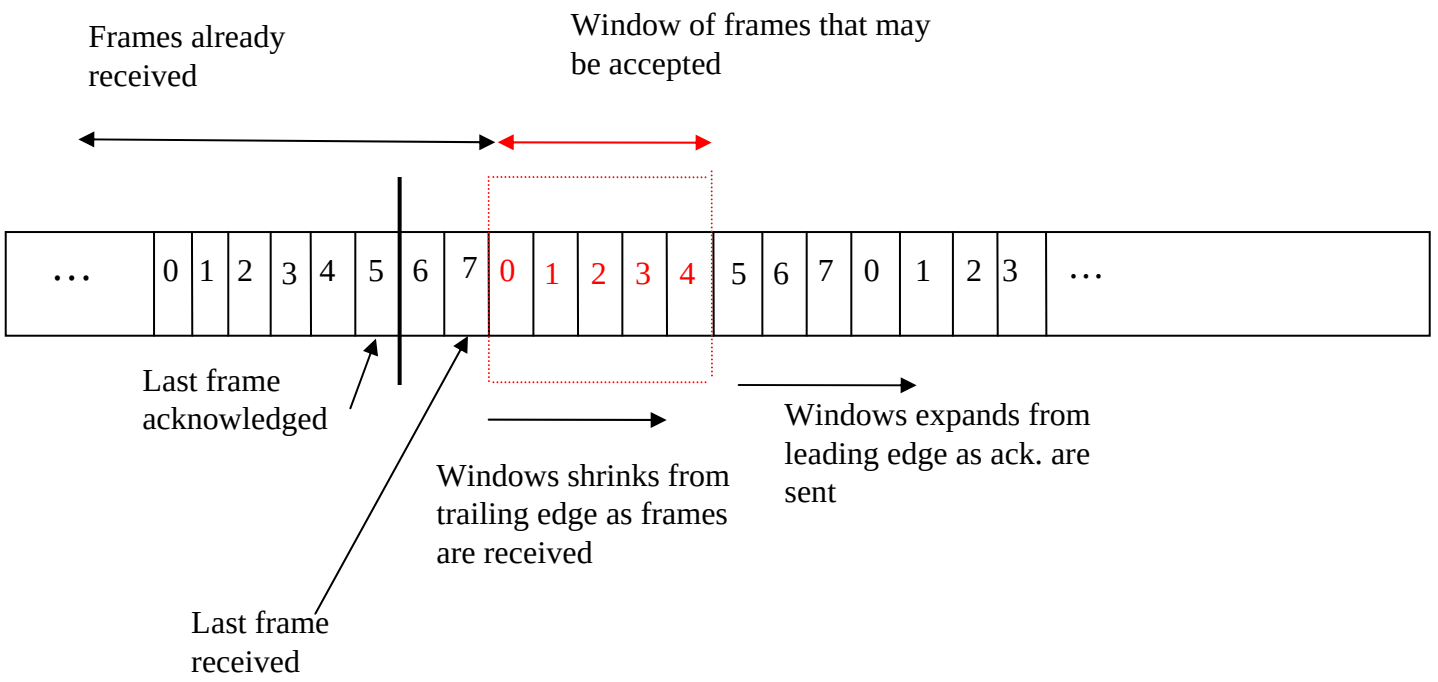
- each of these lists can be thought of as a window of frames
- the operation is referred to ***as sliding window flow control***
- the sequence number to be used occupies a field in the frame and is of bounded size.

Flow control techniques(8)



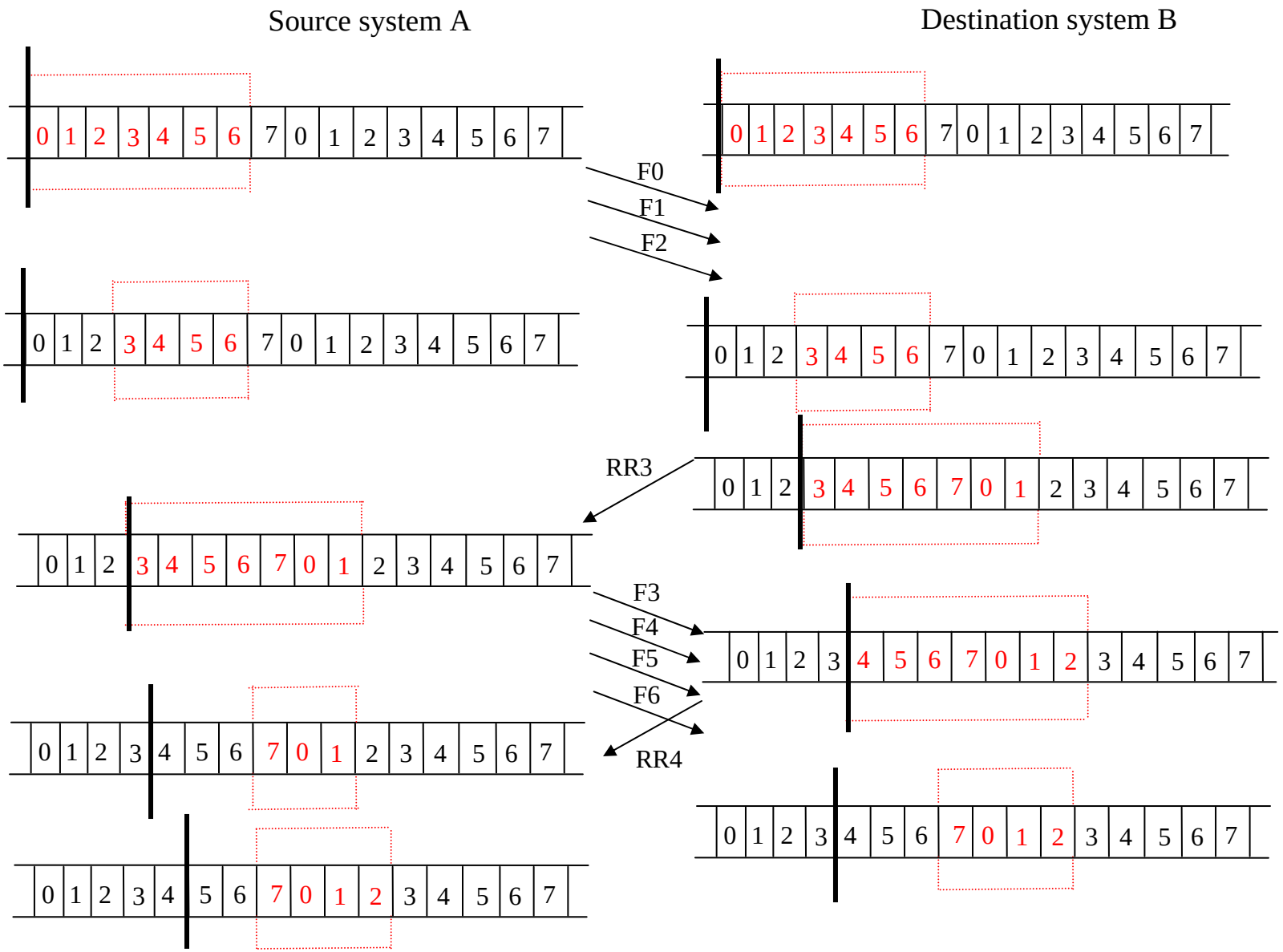
Sliding - window depiction – Transmitter's perspective

Flow control techniques(9)



Sliding - window depiction – Receiver's perspective

Flow control techniques(10)



Flow control techniques(11)

Sliding window flow control

- most protocols also allow a station to completely cut off the flow of frames from the other side by sending a Receive-Not – Ready message which ack former frames but forbids transfer of future frames;
- at some subsequent point the station must send a normal ack to reopen the window

Flow control techniques(12)

Sliding window flow control – transmission in both directions

- each station needs to maintain 2 windows: one for transmit and one for receive;
- each side needs to sends the data and ack's to the other;

To provide sufficient support for this requirement a feature known as piggybacking is typically provided.

- each data frame includes a field that holds the sequence number of that frame plus a field that holds the sequence no used for ack.

How it works:

- if a station has data + ack to send it sends both together in one frame;
- if a station has an ack to send, it sends a separate ack frame;
- if a station has data to send but no new ack it must repeat the last ack; when a station receive a duplicate ack it simple ignores it.

Errors detection and correction principles

Strategies developed by network designers:

1) To include enough redundant information along with each block of data sent, to enable the receiver to deduce what transmitted data must have been;
2) To include only enough redundancy to allow the receiver to deduce that an error occurred, but ***not which error*** and have it request a retransmission;

1) ERROR-CORRECTING-CODES(Forward error correction)
   - Used on wireless links, ..
2) ERROR-DETECTING-CODES
   - Used on channels such as fiber that are highly reliable

Errors detection and correction principles(2)

What an error really is?

- a frame consists of *m*- data bits and *r* redundant or *check bits*;
- total length of a frame: $n = m + r$
- an *n*-bits unit containing data and check bits = ***code word***

Given 2 codes words:

10001001 10110001

Determine how many corresponding bits differ?

10001001

10110001

00111000 *SAU EXCLUSIV*

The number of bit position in which two code words differ is called ***The Hamming distance.***

Significance: if 2 codewords are a Hamming d distance apart, it will require d single- bit errors to convert one into the other.

Errors detection and correction principles(3)

- all 2^n possible data messages are legal, but not ***all are code words***;

- given the algorithm to compute the check bits it is possible to construct the complete list of ***legal code words***;

- find the 2 code words with minimum Hamming distance » Hamming distance of the complete code

- The error detecting and error correcting properties of a code depends on its Hamming distance;

- To detect *d* errors you need a *d*+1 code.

- To correct *d* errors you need a distance 2*d*+1

Expl: error detecting code

Consider a code in which a single parity bit is appended to data

- When data is sent:

$$1011010 \xrightarrow{\textit{even parity}} 1011010$$

$$1011010 \xrightarrow{\textit{odd parity}} 1011011$$

A code with a single parity bit has a distance 2, since any single-bit error produces a code word with a wrong parity. It can be used to *detect single* errors.

Errors detection and correction principles(4)

Expl: error correcting code

Consider a code with only 4 valid codewords: When data is sent:

0000000000; 0000011111; 1111100000; 1111111111

This code has a distance 5 which means that it *correct double errors*.

If the code word

0000000111

arrives, the receiver knows that the original signal must have been

0000011111

If a triple error changes : 0000000000 ***into*** 0000000111 ***the error will not be corrected properly .***

The lower limit of the number of check bits needed to correct single errors:

$$(m + r + 1) \leq 2^r ; m = no\ of\ \ message\ bits$$

$$r = no\ of\ \ check\ bits$$

Errors detecting codes

Probabilities defined with respect to error transmitted frames:

P_b - probability of a single bit error; known as the ***bit error rate***

P_1 - probability that a frame arrives with no bit errors

P_2 - probability that a frame arrives with one or more undetected bit errors

P_3 - probability that a frame arrives with one or more detected bit errors but no undetected bit errors

Consider the case when no means are taken to detect errors ($P_3 = 0$).

$$P_1 = (1 - P_b)^F$$

$$P_2 = 1 - P_1$$

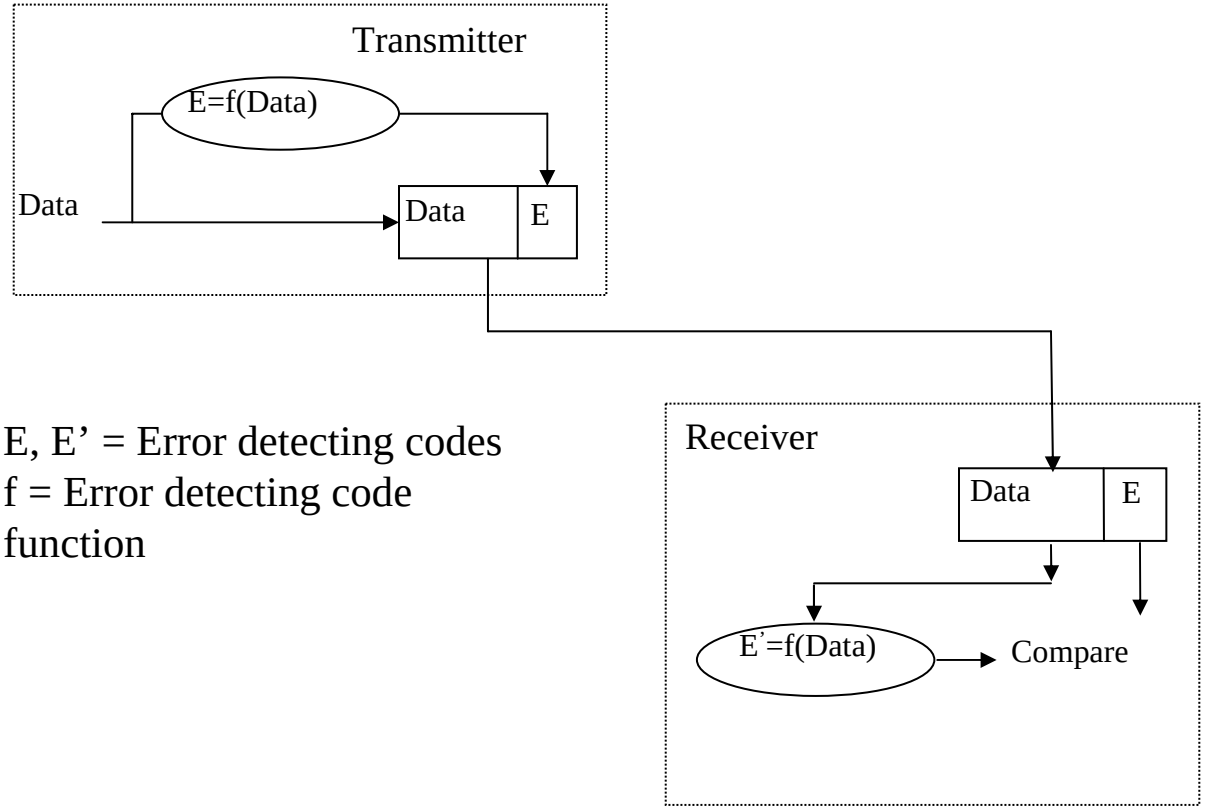
F = number of bits per frame

- The probability that a frame arrives with no bits errors decreases when then probability of a single bit error increases.

- the longer the frame the more bits it has and the higher the probability that one of these is in error

Errors detecting codes(2)

Error detection techniques operates on the following principle:



E, E' = Error detecting codes
f = Error detecting code function

P_3 - the probability that if a frames contains errors, the error detection scheme will detect that fact

P_2 - is known as the residual error rate

Errors detecting codes(3)

Cyclic redundancy check: (CRC)

k = the bit block dimension, or message

n = the length of the bit sequence(known as frame check sequence- FCS) generated by the transmitter;

The resulting frame consisting of $k+n$ bits, is exactly divisible by some predetermined number;

The receiver then divides the incoming frame by that number and, if there is no remainder, assume there was no error;

The procedure can work:

- modulo 2 arithmetic
- polynomials
- digital logic

Errors detecting codes(3)

Cyclic redundancy check: (CRC)

Modulo 2 Arithmetic –uses binary addition without carries which is just the exclusive-or (XOR) operation:

$$\begin{array}{r} 1111 \\ +\,1010 \\ \hline 0101 \end{array}$$

$$\begin{array}{r} 11001 \\ \times \quad 11 \\ \hline 11001 \\ \underline{11001} \\ 101011 \end{array}$$

Define:

$T = (k+n)$ - bit frame to be transmitted $n < k$

$M = (k)$ - bit message, the first k bits of T

$F = (n-k)$ - bit FCS, the last $n-k$ bits of T

$P = (n-k+1)$ - pattern; the predetermined divisor

We would like T / P has no remainder.

Errors detecting codes(4)

Cyclic redundancy check: (CRC) -Modulo 2 Arithmetic

T has the form:

$$T = 2^{n-k} M + F$$

$$\frac{2^{n-k} M}{P} = Q + \frac{R}{P}$$

- we will use the remainder as our FCS

$$T = 2^{n-k} M + R$$

$$\frac{T}{P} = \frac{2^{n-k} M + R}{P} = Q + \frac{R}{P} + \frac{R}{P} = Q$$

- results: *T* exactly divisible by *P*

- *FCS is easily generated: divide* $2^{n-k} M$ *by P and use the (n-k) bit remainder as FCS;*

- *at the reception the receiver will divide T by P and will get no remainder if there has been no errors;*

Errors detecting codes(5)

Exemple:

1) given $\quad Message\ M = 1010001101 \quad (10 bits)$

$Pattern\ P = 110101 \quad (6 bits)$

$FCS\ R = to\ be\ calculated \quad (5 bits)$

2) the message is multiplied by 2^5 yielding: 101000110100000

3) this product is divided by P

$$
\begin{array}{r|l}
 & 1101010110 \leftarrow Q \\
\hline
P \rightarrow 110101 & 101000110100000 \rightarrow 2^{n-k} M \\
 & \underline{110101} \\
 & \quad 111011 \\
 & \quad \underline{110101} \\
 & \quad\quad 111010 \\
 & \quad\quad \underline{110101} \\
 & \quad\quad\quad 111110 \\
 & \quad\quad\quad \underline{110101} \\
 & \quad\quad\quad\quad 101100 \\
 & \quad\quad\quad\quad \underline{110101} \\
 & \quad\quad\quad\quad\quad 110010 \\
 & \quad\quad\quad\quad\quad \underline{110101} \\
 & \quad\quad\quad\quad\quad\quad 01110 \leftarrow R
\end{array}
$$

Errors detecting codes(6)

4) the remainder is added to $2^5 M$ to give :

$$T = 1010001101 \mid 01110$$

5) if there are no errors the receiver receives T intact. The receiver frame is divided by P

$$\begin{array}{r|l} & 1101010110 \leftarrow Q \\ \hline P \rightarrow 110101 & 1010001101011110 \rightarrow T \end{array}$$

$$\begin{array}{r}
110101 \\
\hline
111011 \\
110101 \\
\hline
111010 \\
110101 \\
\hline
111110 \\
110101 \\
\hline
101111 \\
110101 \\
\hline
110101 \\
110101 \\
\hline
00000 \leftarrow R
\end{array}$$

Errors detecting codes(7)

The occurrence of an error is easily expressed: an error results in the reversal of a bit. This is equivalent to taking the exclusive or of the bit and 1.

The error in an $n+k$ frame can be represented by an $n+k$ bit field with 1's in each error position.

The resulting frame T_r can be expressed as:

$$T_r = T \oplus E$$

T – transmitted frame;

E - error pattern with 1's in positions where errors occur;

T_r - received frame;

OBS The receiver will fail to detect an error if and only if T_r is divisible by P, which is equivalent to E divisible by P.