0.1 Data Link layer functions and services

0.1.1 Main functions

- Provide service interface to the network layer.
- Eliminate/reduce transmission errors.
- Regulate data flow: Slow receivers not swamped by fast senders.

0.1.2 Services provided

Principal service: Transfer data from the network layer on the source machine to the network layer on the destination machine.

There are three reasonable possibilities that we will consider:

- **Unacknowledged connectionless service:**
  - No logical connection is established beforehand or released afterwards.
  - Transmitter sends independent frames without having the destination machine acknowledge them.
  - If a frame is lost due to noise on the line, no attempt is made to detect or recover from that loss.
  - Appropriate when the error rate is very low and for real-time traffic.
- **Acknowledged connectionless service:**
  - No logical connections used.
  - Each frame sent is individually acknowledged so the sender knows if a frame arrived correctly or has been lost.
  - If it has not arrived within a specified time interval, it can be sent again.
  - This service is useful over unreliable channels, such as wireless systems.(i.e. Wi-Fi).
- **Acknowledged connection-oriented service:**
  - The source and destination machines establish a connection before any data are transferred.
  - Each frame is numbered, and the data link layer guarantees that each frame sent is indeed received.
  - Guarantees that each frame is received exactly once and that all frames are received in the right order.
  - Appropriate over long, unreliable links (satellite channel, long-distance telephone circuit).

– Divided in 3 phases:
  * **First phase:** The connection is established (initialize variables and counters needed to keep track of which frames have been received and which ones have not).
  * **Second phase:** One or more frames are actually transmitted.
  * **Third phase:** The connection is released (free the variables, buffers, and other resources used to maintain the connection).

0.2 Framing

Breaking up the bit stream into discrete frames, computing a short token called a checksum for each frame, and including the checksum in the frame when it is transmitted. When a frame arrives at the destination,the checksum is recomputed. If it is different from the one contained in the frame, the data link layer knows that an error occurred.

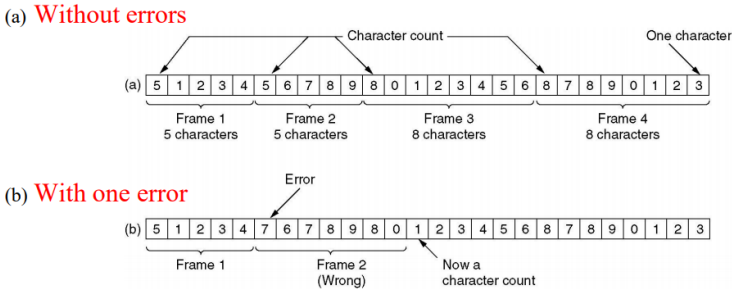
A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth. We will look at three methods:

0.2.1 Byte count

Uses a field in the header to specify the number of bytes in the frame. When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is.

Issues

- The count can be garbled by a transmission error.
- A single bit flip, may trigger the destination to get out of synchronization.
- If an out-of-sync occurs it is unable to locate the correct start of the next frame.



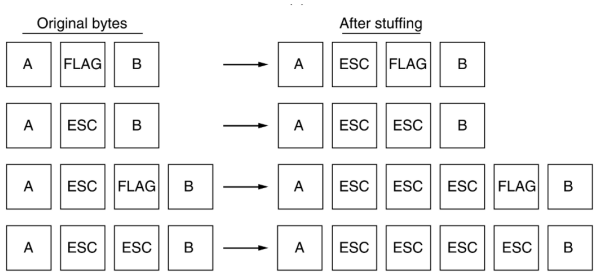
0.2.2 Flag bytes with byte stuffing

This method gets around the problem of resynchronization by having each frame start and end with special bytes (flag bytes).

However, this flag may occur in the middle of the data and induce the receiver in error by thinking the end of the frame was reached. This issue can be solved using **byte stuffing**.

Byte Stuffing

- Inserting a special escape byte (ESC) before each flag byte in the data.
- Makes framing flag bytes distinguishable from the ones in the data.
- Escape bytes present in the data also need to be escaped.



0.2.3 Flag bits with bit stuffing

- Each frame begins and ends with a special bit pattern (01111110 / 0x7E).
- When the sender finds five consecutive 1 bits in the data, it stuffs a 0 bit into the outgoing bit stream.
- When the receiver finds five consecutive incoming 1 bits, followed by a 0 bit, it destuffs the 0 bit.

Advantages:

- The boundary between two frames is unambiguously recognized by the flag pattern (flag sequences can only occur at frame boundaries and never within the data).
- Frames can contain an arbitrary number of bits made up of units of any size.
- Ensures a minimum density of transitions that help the physical layer maintain synchronization.

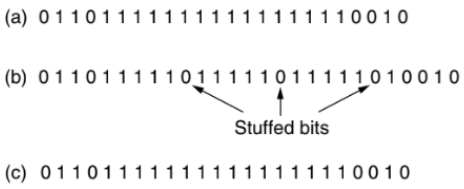


Figure 5. Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

Both byte and bit stuffing:

- Are completely transparent to the network layer in both computers.
- Have a frame length that depends on the contents of the data.

Many data link protocols use a combination of these three methods for safety.

0.3 Error detection

0.3.1 Types of Errors

- **Simple Error:** Random and independent from previous error.
- **Errors in burst:**
  - Not independent.
  - Affect neighbour bits.
  - Burst lenght defined by the first and last bits in error.

0.3.2 Counting Errors

Frame Error Probability(FER):

$$FER = 1 - (1 - BER)^n \tag{1}$$

BER = Bit Error Ratio
n = frame length

No Error Probability: $P = (1-p)^n$
Error Probability: $P = 1 - (1-p)^n$
i Error Probability: $P = \binom{n}{i} p^i (1-p)^{n-i}$

p = bit error probability
n = frame length

0.3.3 Error Detection Techniques

Used by the receiver to determine if a packet contains errors. If a packet is found to contain errors, the receiver may request the transmitter to re-send the packet.

0.3.4 Parity Check

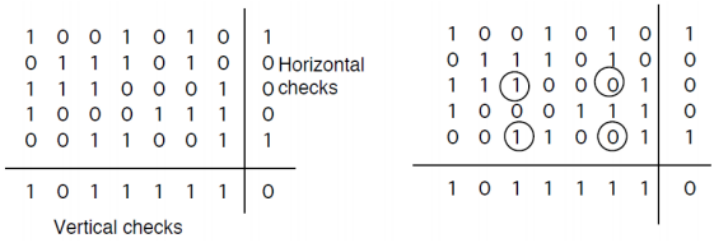
Simple Parity Check: One parity bit added to every k information bits so that:

- The total number of bits 1 even (even parity).
- The total number of bits 1 odd (odd parity).

Allows the detection of simple errors and any number of odd errors in a block of k + 1 bits. However, does not detect even number of errors in a block of k + 1 bits.

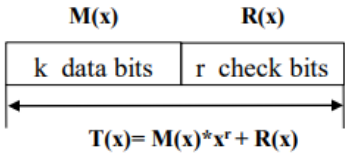
Bi-dimensional Parity

- Parity per row.
- Parity per column.
- Minimum code distance = 4.



0.3.5 Cyclic Redundancy Check (CRC)

A fixed number of check bits are appended to the message to be transmitted. Data receivers check on the check value attached by finding the remainder of the polynomial division of the contents transmitted. If it seems that an error has occurred, a negative acknowledgement is transmitted asking for data retransmission.



The bit string is represented as a polynomium (i.e. $110011 \Rightarrow x^5+x^4+x+1$)

How to compute the check bits: R(x)?

- Choose a generator string G(x) of length r+1 bits.
- Choose R(x) such that T(x) is a multiple of $G(x) : T(x) = A \times G(x)$.

Generating R(x):

$$R(x) = M(x)x^r \textbf{ mod } G(x) \tag{2}$$

Choice of G(x) is very important! ($G(x) = x^r + \ldots + 1$)

Generating R(x) example:

Assume for example:

- r=3.
- $G(x) = x^3 + 1 \Rightarrow 1001$.
- $M(x) = x^5 + x^4 + x^2 + 1 \Rightarrow 110101$.

Then:

- $x^r = x^3$.
- $M(x) \times x^3 = x^8 + x^7 + x^5 + x^3 \Rightarrow 110101000$.
- $R(x) = M(x)x^3$ **mod** $G(x) = 110101000$ **mod** $1001 = 011 = x^1 + 1$

Checking at the Receiver

- Divide T(x) by G(x):
  - If the remainder $R(x) = 0 \Rightarrow$ no errors.
  - If the remainder $R(x)! = 0 \Rightarrow$ errors have occurred.

Performance:

For r check bits per frame the following can be detected

- All patterns of 1, 2, or 3 errors (d $>$ 3).
- All bursts of errors of r or fewer bits.
- All errors consisting of an odd number of inverted bits.

0.4 Automatic Repeat reQuest (ARQ)

An error-control method for data transmission that uses acknowledgements (messages indicating whether or not the message has been correctly received) and timeouts to achieve reliable data transmission over an unreliable service. This mechanisms automatically request the retransmission of:

- Missing packets.
- Packets with errors.

There are three common ARQ schemes:

0.4.1 Stop and Wait

- Sender transmits information frame I and waits for positive confirmation ACK from receiver.
- Receiver receives I frame:
  - If I frame has no error sends ACK.
  - If I frame has error sends NACK.
- Sender receives I frame:
  - If ACK, proceeds and transmits new frame.
  - If NACK, retransmits frame I.
- If I, ACK or NACK is lost a timeout is required!

Issue: If the transmitter times-out and sends a packet twice, the receiver cannot tell whether the second frame is a retransmission or a new frame transmission.
Solution: Define a 1 bit sequence number in the header of the frame.

This sequence number alternates (from 0 to 1) in subsequent frames. The transmitter sends a frame with a sequence number attached to it so the receiver can check if it matches the expected. When the receiver sends an ACK, it includes the sequence number of the next packet it expects. This way, the receiver can detect duplicated frames by checking if the frame sequence numbers alternate.

Efficiency(S):

$$S = \frac{T_f}{T_f + 2 \times T_{prop}} = \frac{1}{1 + 2a} \tag{3}$$

where:

T_f = Data transmission time
T_{prop} = Propagation Delay

Probability of k Attempts required to transmit a frame with success

$$P[A = k] = {p_e}^{k-1}(1 - p_e) \tag{4}$$

where:

p_e = frame error probability(FER)

Expected number of Attempts to transmit a frame with success

$$E[A] = \frac{1}{1 - p_e} \tag{5}$$

where:

p_e = frame error probability(FER)

Efficiency with Errors

$$S = \frac{T_f}{E[A](T_f + 2 \times T_{prop})} = \frac{1 - p_e}{1 + 2a} \tag{6}$$

where:

p_e = frame error probability(FER)

0.4.2 Go Back N

Allows the transmission of new packets before earlier ones are acknowledged.

Sender:

- May transmit up to W frames without receiving RR(Receiver Ready = ACK).
- I frames are numbered sequentially I(NS): I(0), I(1), I(2), etc.
- Cannot send I(NS=i+W) until it has received the RR(NR=i).

Receiver:

- Does not accept frames out of sequence.
- Sends RR(NR) to sender indicating:
  - That all the packets up to NR-1 have been received in sequence.
  - The sequence number, NR, of the next expected frame.

Behaviour under Errors

- Frames with errors are silently discarded by the Receiver.
- If Receiver receives Data frame out of sequence:
  - First out-of-sequence-frame: Receiver sends REJ(NR) where NR = next in-sequence frame expected.
  - Following out-of sequence-frames: Receiver discards them; no REJ sent.
- When Sender receives REJ(NR=x), the Sender:
  - Goes-Back and retransmits I(x), I(x+1), etc.
  - Continues using Sliding Window mechanism.
- If timeout occurs, the Sender:
  - Requests the Receiver to send a RR message.
  - Sends a special message (RR command message).

Maximum Window Size(W):

$$W = M - 1 = 2^k - 1 \tag{7}$$

where:

M = Number of sequence numbers
k = Number of bits used to code sequence numbers

Efficiency:

- If $W \geq 1 + 2a \Rightarrow S = 1$.
- If $W < 1 + 2a \Rightarrow S = \frac{W}{1+2a}$.

Efficiency with Errors:

$$S = \begin{cases} \dfrac{1-p_e}{1+2ap_e} & ,W \geq 1+2a \\ \dfrac{W(1-p_e)}{(1+2a)(1-p_e+Wp_e)} & ,W < 1+2a \end{cases}$$

Figura 1: pe - frame error probability (ratio, FER)

0.4.3 Selective Repeat

Similar to **Go Back N**, however it does not discard successful frames when errors occur.

Receiver:

- Accepts out of sequence frames.
- Confirms negatively, SREJ, a frame not arrived.
- Uses RR to confirm blocks of frames arrived in sequence.

Sender: Retransmits only the frames signaled by SREJ.

Maximum window size(W):

$$W = \frac{M}{2} = 2^{k-1} \tag{8}$$

where:

M = Number of sequence numbers
k = Number of bits used to code sequence numbers

Efficiency:

$$S = \begin{cases} 1-p_e & ,W \geq 1+2a \\ \dfrac{W(1-p_e)}{1+2a} & ,W < 1+2a \end{cases}$$

Figura 2: pe - frame error probability (ratio, FER)

0.4.4 Useful Formulas for All Methods

Data transmission time (T_f):

$$T_f = \frac{L}{R} \tag{9}$$

where:

L = Frame Size
R = Data Rate

Propagation Delay(T_{prop}):

$$T_{prop} = \frac{d}{V} \tag{10}$$

where:

d = Distance between sender and receiver
V = Propagation Velocity

SUMETHIN(a)

$$a = \frac{T_{prop}}{T_f} \tag{11}$$

where:

T_{prop} = Propagation Delay
T_f = Data transmission time

Maximum Rat(R_{max}):

$$R_{max} = S \times R \tag{12}$$

where:

S = Efficiency
R = Data rate

Round Trip Time(RTT - Time of transmission and acknowledgement of a frame):

$$RTT = 2 \times T_{prop} + T_f \tag{13}$$

where:

T_{prop} = Propagation Delay
T_f = Data transmission time

0.5 Framing, Error detection and ARQ in common networks

0.6 Reliability in the Protocol Stack