HAMMING CODES

Hamming codes provide another method for error correction. Error bits, called Hamming bits, are inserted into message bits at random locations. It is believed that the randomness of their locations reduces the odds that these Hamming bits themselves would be in error. This is based on a mathematical assumption that because there are so many more message bits compared with Hamming bits, there is a greater chance for a message bit to be in error than for a Hamming bit to be wrong. Determining the placement and binary value of the Hamming bits can be implemented using hardware, but it is often more practical to implement them using software. The number of bits in a message (M) are counted and used to solve the following equation to determine the number of Hamming bits (H) to be used:

$2^H \geq M + H + 1$

Once the number of Hamming bits is determined, the actual placement of the bits into the message is performed. It is important to note that despite the random nature of the Hamming bit placements, the exact sample placements must be known and used by both the transmitter and receiver. Once the Hamming bits are inserted into their positions, the numerical values of the bit positions of the logic 1 bits in the original message are listed. The equivalent binary numbers of these values are added in the same manner as used in previous error methods by discarding all carry results. The sum produced is used as the states of the Hamming bits in the message. The numerical difference between the Hamming values transmitted and that produced at the receiver indicates the bit position that contains a bad bit, which is then inverted to correct it.

Ex. The given data

10010001100101(14- bits)

The number of hamming codes

$2^H \geq M + H + 1$

H = ? M = 14 to satisfy this equation H should be 5 i.e. 5 hamming code bits should be incorporated in the data bits.

1 0 0 1 0 0 0 1 1 0 H 0 H 1 H 0 H 1 H

Now count the positions where binary 1's are present. Add using mod 2 operation (Ex-OR). The result will give the Hamming code at the transmitter end.

1's position	Binary equivalent

2	-	0	0	0	1	0
6	-	0	0	1	1	0
11	-	0	1	0	1	1
12	-	0	1	1	0	0
16	-	1	0	0	0	0
19	-	1	0	0	1	1
Hamming code =		0	0	0	0	0

This Hamming code will be incorporated at the places of 'H' in the data bits and the data will be transmitted.

How to find out there is an error in the data?

Let the receiver received the 12th bit as zero. The receiver also finds out the Hamming code in the same way as transmitter.

1's position		Binary equivalent				
2	-	0	0	0	1	0
6	-	0	0	1	1	0
11	-	0	1	0	1	1
16	-	1	0	0	0	0
19	-	1	0	0	1	1
Hamming code at the receiver		0	1	1	0	0

Hamming code at the Tx	0	0	0	0	0
Hamming code at the Rx	0	1	1	0	0
	0	1	1	0	0

The decimal equivalent for the binary is **12** so error is occurred at 12th place.

Data Link Protocols

1. Unrestricted Simplex Protocol:

In this the following assumptions are made

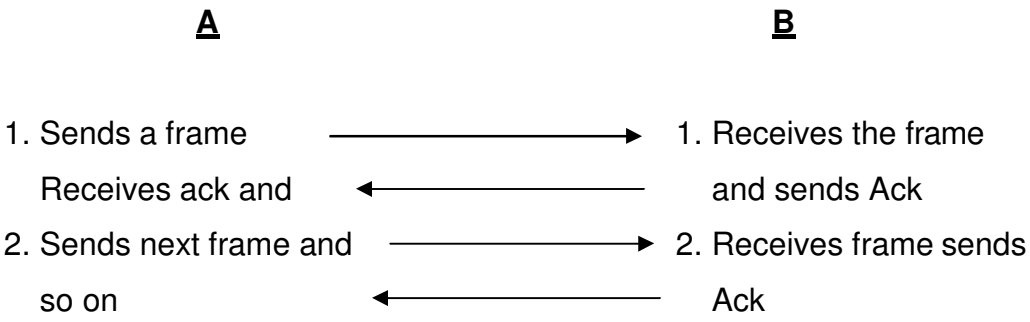
a. Data transmission is simplex i.e. transmitted in one direction only.
b. Both transmitting and receiving network layers are ready.
c. Processing time is ignored.
d. Infinite buffer space is available.
e. An error free channel.

This is an unrealistic protocol, which has a nickname “Utopia”.

2. A simplex stop and wait protocol:

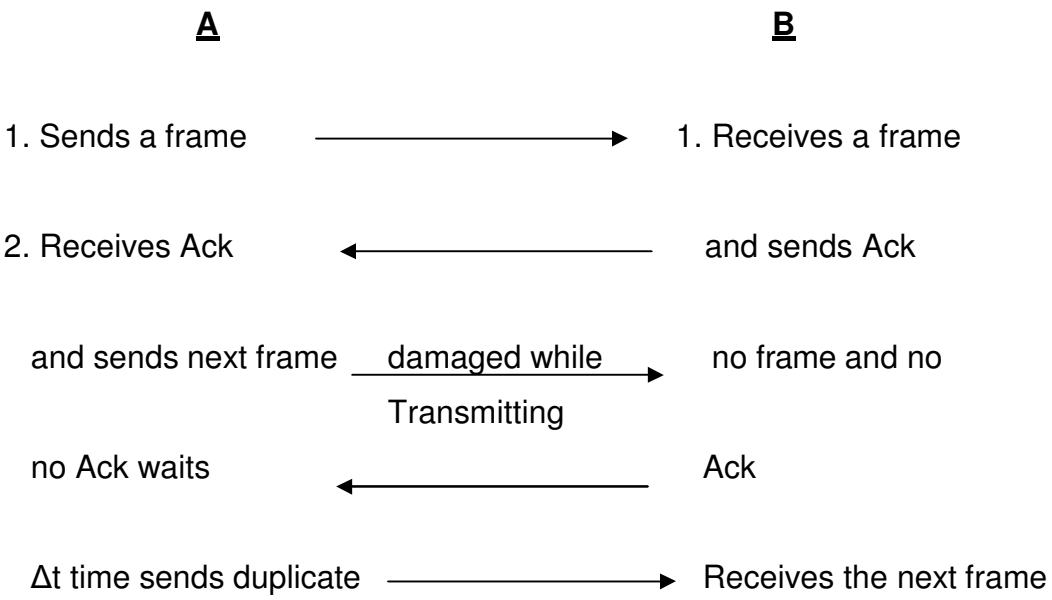
The following assumptions are made

a. Error free channel.
b. Data transmission simplex.

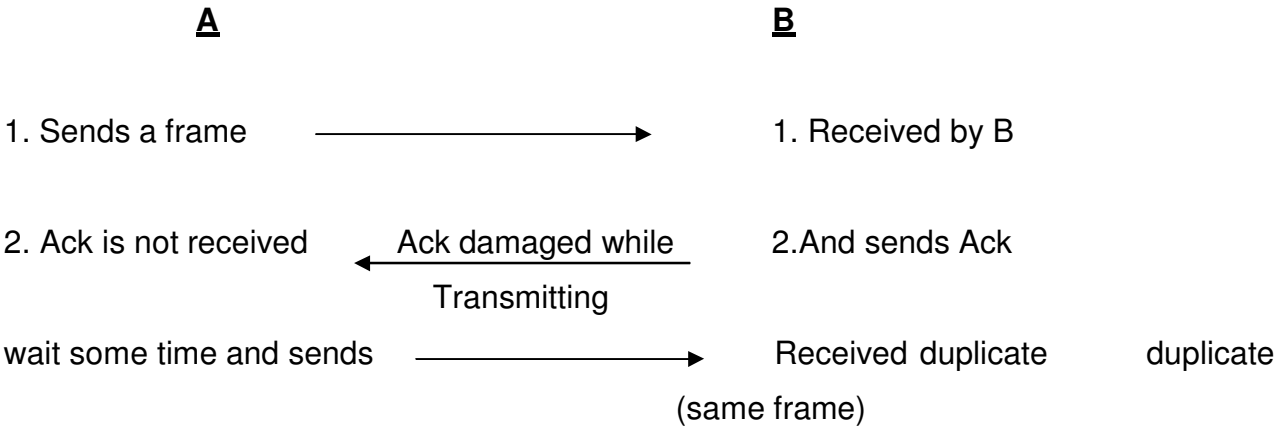


Since the transmitter waits for Δt time for an Ack this protocol is called stop and wait protocol.

3. A simplex protocol for a noisy channel



When this protocol fails?

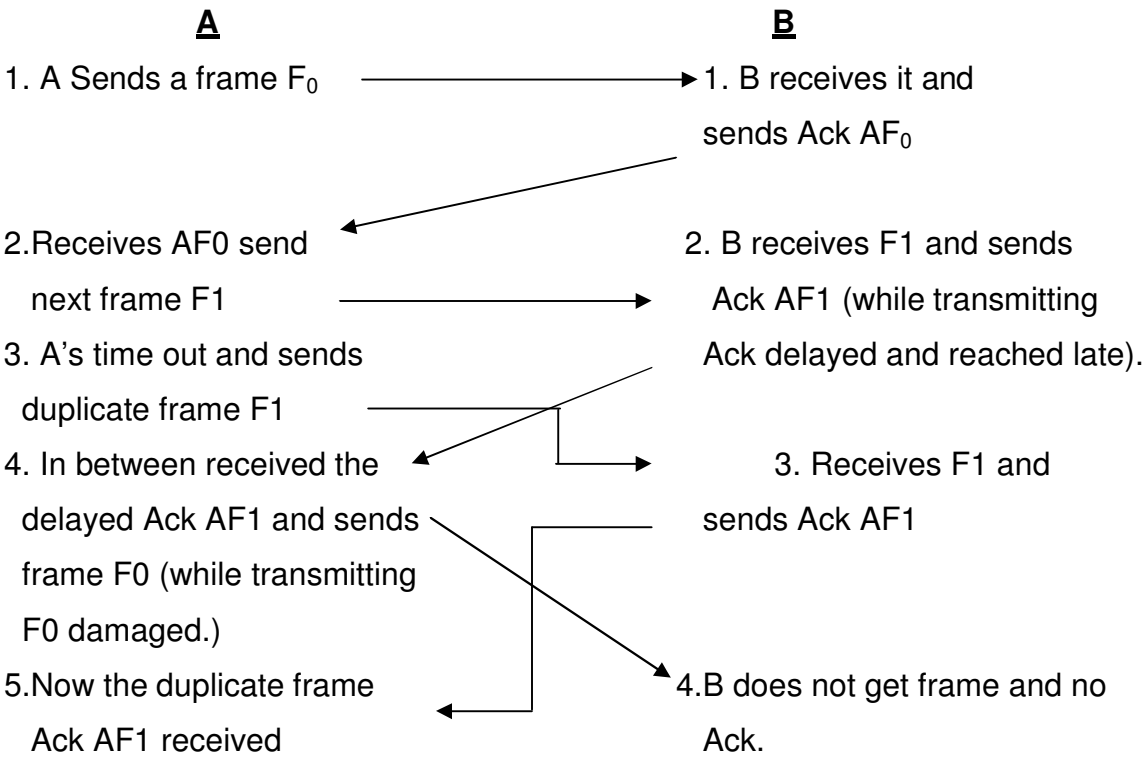


At this situation protocol fails because the receiver receives a duplicate frame and there is no way to find out whether the receiver frame is original or duplicate. So the protocol fails at this situation.

Now what is needed is some way for the Rx to distinguish a frame and a duplicate. To achieve this, the sender has to put a sequence number in the header of each frame it sends. The Rx can check the sequence number of each arriving frame to see if it is a new frame or a duplicate.

Here a question arises: What is the minimum number of bits needed for the sequence number? The ambiguity is between a frame and its successor. A 1-bit sequence number (0 or 1) is therefore sufficient. At each instant of time, the receiver excepts a particular sequence number next. Any arriving frame containing wrong sequence number is rejected as a duplicate. When a frame containing the correct sequence number arrives, it is accepted, passed to the network layer and then expected sequence number is incremented i.e. 0 becomes 1 and one becomes 0. Protocols in which a sender waits for a positive ack before advancing to the next data item are often called PAR (positive ack with retransmission) or ARQ (automatic repeat request).

When this protocol fails?



6. Now A thinks that the Ack received is the ack of new frame F0 and A sends next frame F1. So a frame F0 is missed. At this situation this protocol fails.

PIGGY BACKING

In most practical situations there is a need of transmitting data in both directions. This can be achieved by full duplex transmission. If this is done we have two separate physical circuits each with a 'forward ' and 'reverse' channel. In both cases, the reverse channel is almost wasted. To overcome this problem a technique called **piggy backing** is used.

The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as **piggy backing**.

However, piggybacking introduces a complication not present with separate acknowledgements. How long should the data link layer wait longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements. Of course, the data link layer cannot foretell the future, so it must resort to some ad hoc scheme, such as waiting a fixed number of milli seconds. If a new packet arrives quickly, the acknowledgement is piggy backed onto it; otherwise, if no new packet has arrived by the end of this time period, the data link layer just sends a separate acknowledgement frame.