DATALINK LAYER

Introduction

The Data Link Layer break the bit stream into discrete frames and compute the checksum for each frame. When a Frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from one computed contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

FRAMING METHODS

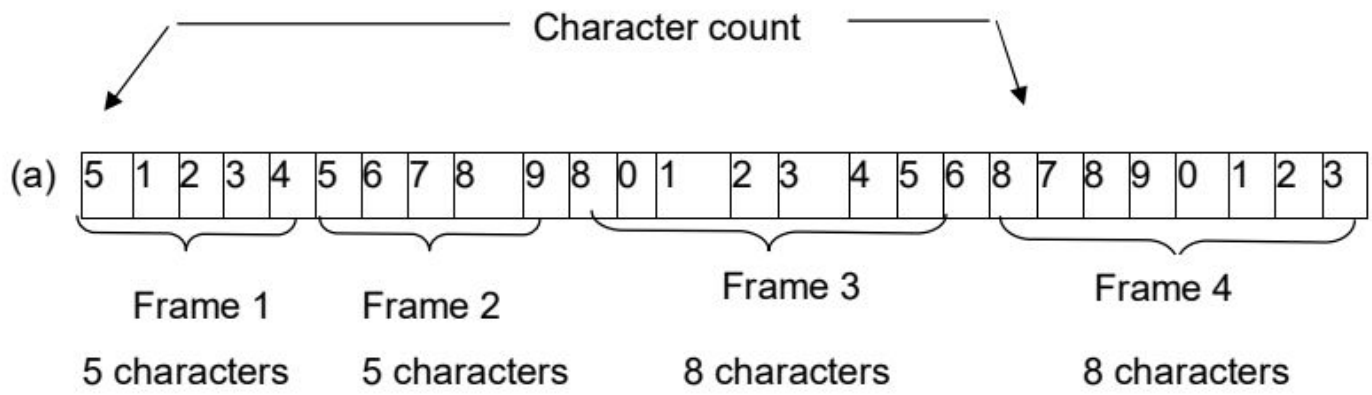
1. CHARATER COUNT METHOD

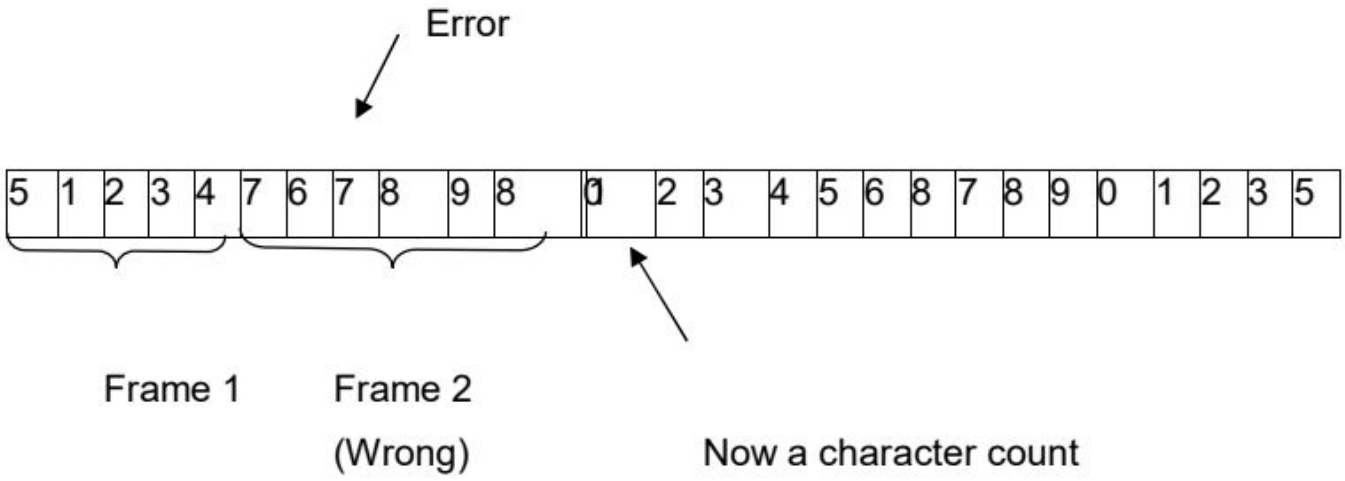
2. STARTING AND ENDING CHARACTERS, WITH CHARATER STUFFING

3. STARTING AND ENDING FLAGS, WITH BIT STUFFING

CHARATER COUNT METHOD:

In this method a field in the header will be used to specify the number of CHARACTERS in the frame. When data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is.
The trouble with this algorithm is that the count can be garbed by a transmission error resulting the destination will get out of synchronization and will be unable to locate the start of the next frame. There is no way of telling where the next frame starts. For this reason this method is rarely used.





A Character Stream (a) Without errors (b) With one error

CHARATER STUFFING METHOD:

In this method each frame will start with a FLAG and ends with a FLAG.

The starting flag is **DLE STX ---- Data Link Escape Start of Text**

The ending flag is **DLE ETX ---- Data link Escape End of Text.**

Ex 1. The given Data ABRFCXDGJHKK12435ASBGXRR

The Data will be sent

DLE STX ABRFCXDGJHKK12435ASBGXRR DLE STX

Ex 2. The given Data **ASHGTRDXZBNHG DLE STX %$#54378**

The data will be sent as

DLE STX ASHGTRDXZBNHG DLE DLE STX %$#54378 DLE ETX

Dis Adv:

1.24 bits are unnecessarily stuffed.

2. Transmission delay.

BIT STUFFING METHOD

In this method every frame will start with a **flag 01111110.**

In the data if there are **FIVE** consecutive ONE 's are there then a ZERO will be stuffed.

Ex. The given data is 01111000011111110101001111110 01111101100

The data will be sent as

01111110 01111000011111**0**1101010011111**0**10 01111**0**01100

Stuffed bits

Advantages:

1. Only one bit is stuffed.
2. No transmission delay

ERROR – CORRECTING AND DETECTING CODES

Network designers have developed two basics strategies for dealing with errors. One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been .The other way is 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. The former strategy uses **Error – correcting codes** and the latter uses **Error- detecting codes**.

The **Error – correcting and Error- detecting methods are**

1. PARITY METHOD
2. LRC METHOD (Longitudinal redundancy check)
3. CRC METHOD (Cyclic redundancy check)
4. HAMMING CODE METHOD

PARITY METHOD

- appends a parity bit to the end of each word in the frame

- Even parity is used for asynchronous Transmission

- Odd parity is used for synchronous Transmission

	Character code	even parity	odd parity
Ex 1.	1100100	1100100 <u>1</u>	1100100 <u>0</u>
2.	0011000	0011000 <u>0</u>	0011000 <u>1</u>

If one bit or any odd no bits is erroneously inverted during Transmission, the Receiver will detect an error. How ever if two or even no of bits are inverted an undetected error occurs.

Ex 3. The Transmitted data is 10011010. The received data is 11011010.

Let both the transmitter and receiver are agreed on EVEN parity.
Now an error will be detected, since the no of ones received are ODD

4. The Transmitted data is 10011010. The received data is 01011010

The received data is wrong even though the no of ones are EVEN.

Science two bits are inverted error can't be detected.

Longitudinal Redundancy Check(LRC)

The frame is viewed as a block of characters arranged in 2-dimensions. To each character is appended a parity bit. In addition a parity bit is generated for each bit position across all characters i.e., an additional character is generated in which the I^{th} bit of the character is parity bit for the I^{th} bit of all other characters in the block. This can be expressed mathematically using exclusive OR(+) operation. The parity bit at the end of each character of row parity

$$R_j = b_{1j} + b_{2j} + \text{------} b_{nj}$$

Where R_j=Parity bit of jth character

b_{ij}=ith bit in jth character

This equation generates even parity.

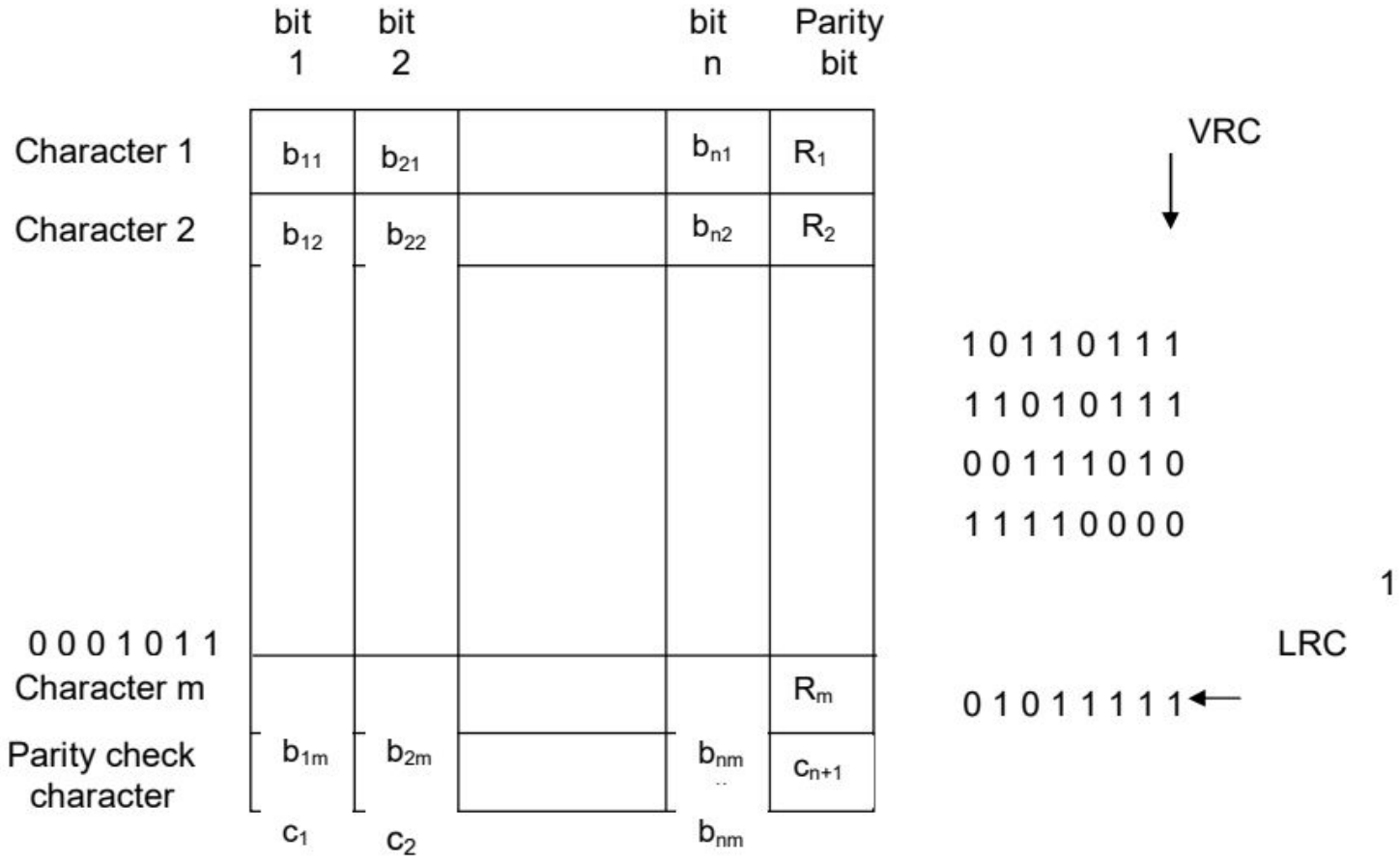
$$C_i = b_{i1} + b_{i2} + \text{-----} + b_{in}$$

Where C_i=ith bit of parity check character

m=number of characters in a frame

In this format the parity bits at the end of each character are referred to as

The Vertical Redundancy Check (VRC) and the Parity check character is referred to as the Longitudinal Redundancy Check (LRC).



CRC Method

1. The frame is expressed in the form of a Polynomial F(x).0 1 1 1 1 1 1 0

2. Both the sender and receiver will agree upon a generator polynomial G(x) in advance.

3. Let 'r' be the degree of G(x).Append 'r' zero bits to the lower – order end of frame now it contains m+r bits.

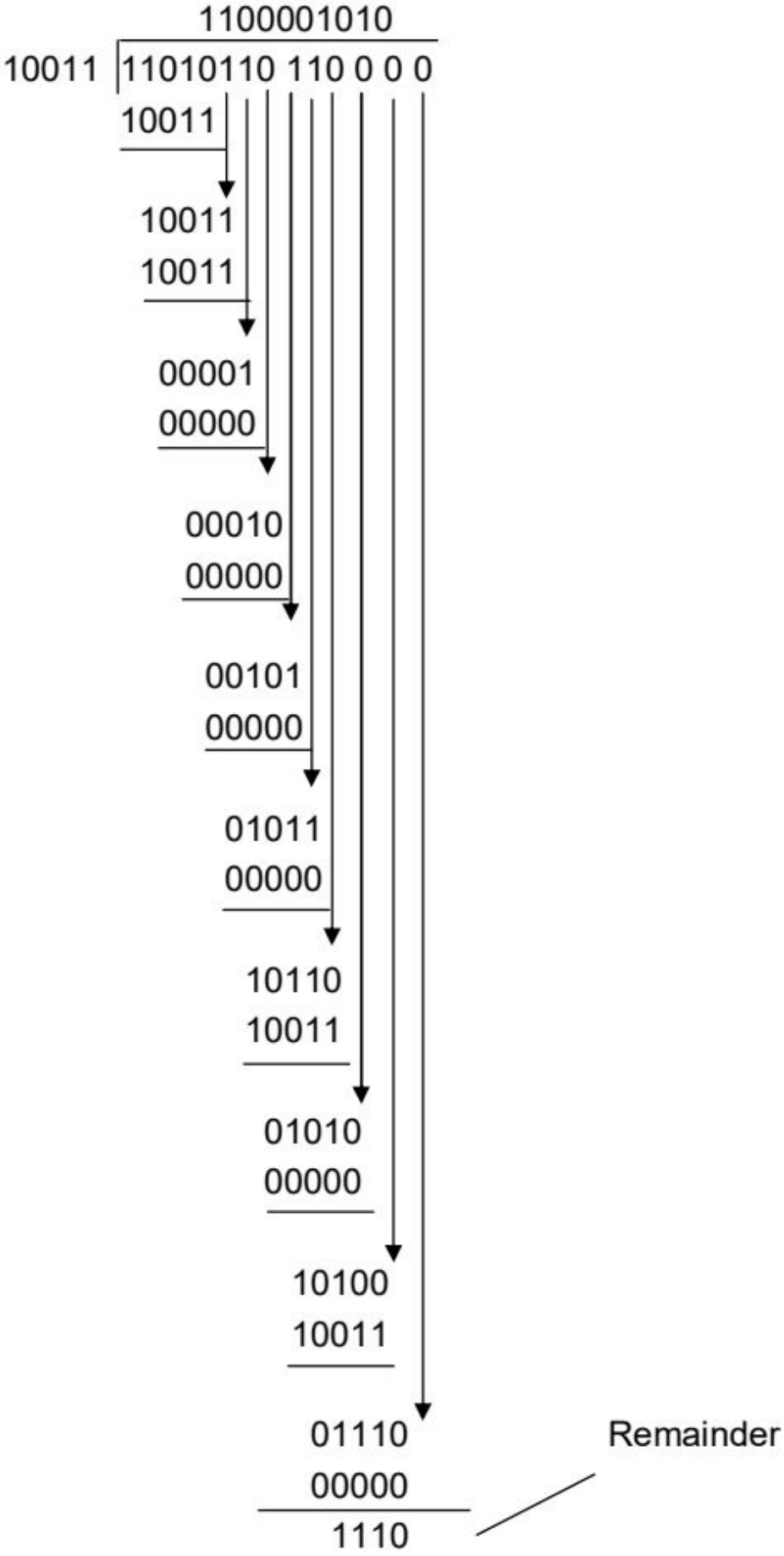
4. Divide the bit string by G(x) using Mod 2 operation.

5. Transmitted frame [T(x)] = frame + remainder

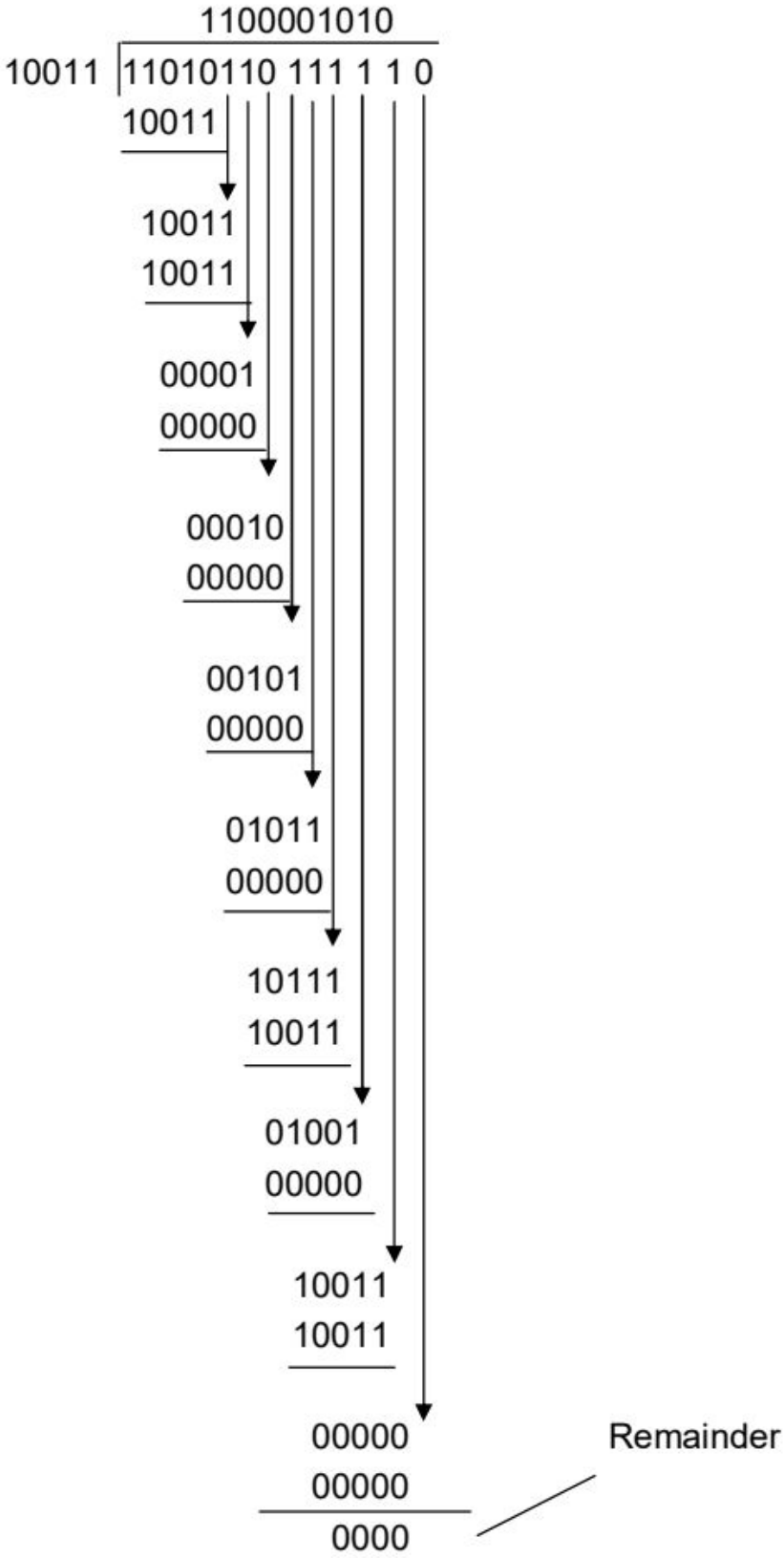
6. Divide T(x) by G(x) at the receiver end. If the result is a zero, then the frame is transmitted correctly. Ex. Frame: 1101011011

Generator: 10011

Message after appending 4 zero bits: 11010110000



Transmitted frame: 11010110111110



Since the remainder is zero there is no error in the transmitted frame.

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

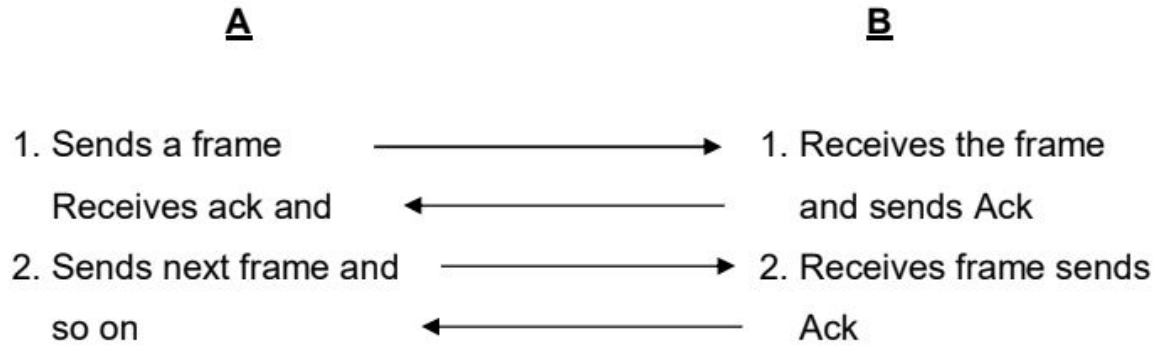
a. Data transmission is simplex i.e. transmitted in one direction only.
b. Both transmitting and receiving network layers are ready.
c. Processing time is ignored.
d. Infinite buffer space is available.
e. An error free channel.

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

2. A simplex stop and wait protocol:

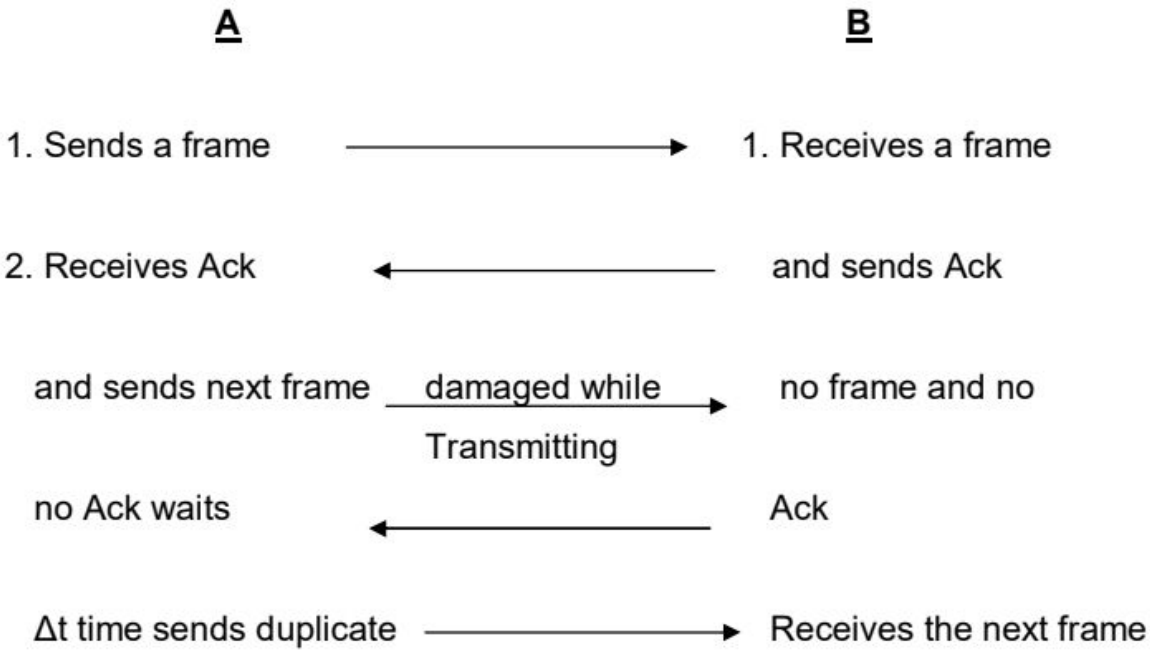
The following assumptions are made

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

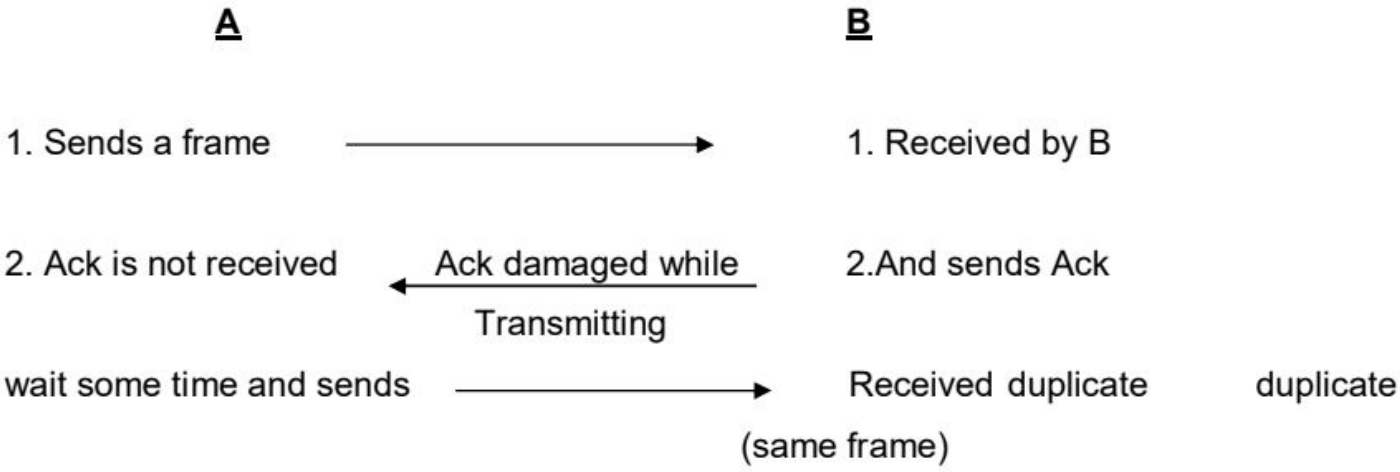


Since the transmitter waits for Δt time for an Ack this protocol is called stop and wait protocol.

3. A simplex protocol for a noisy channel



When this protocol fails?

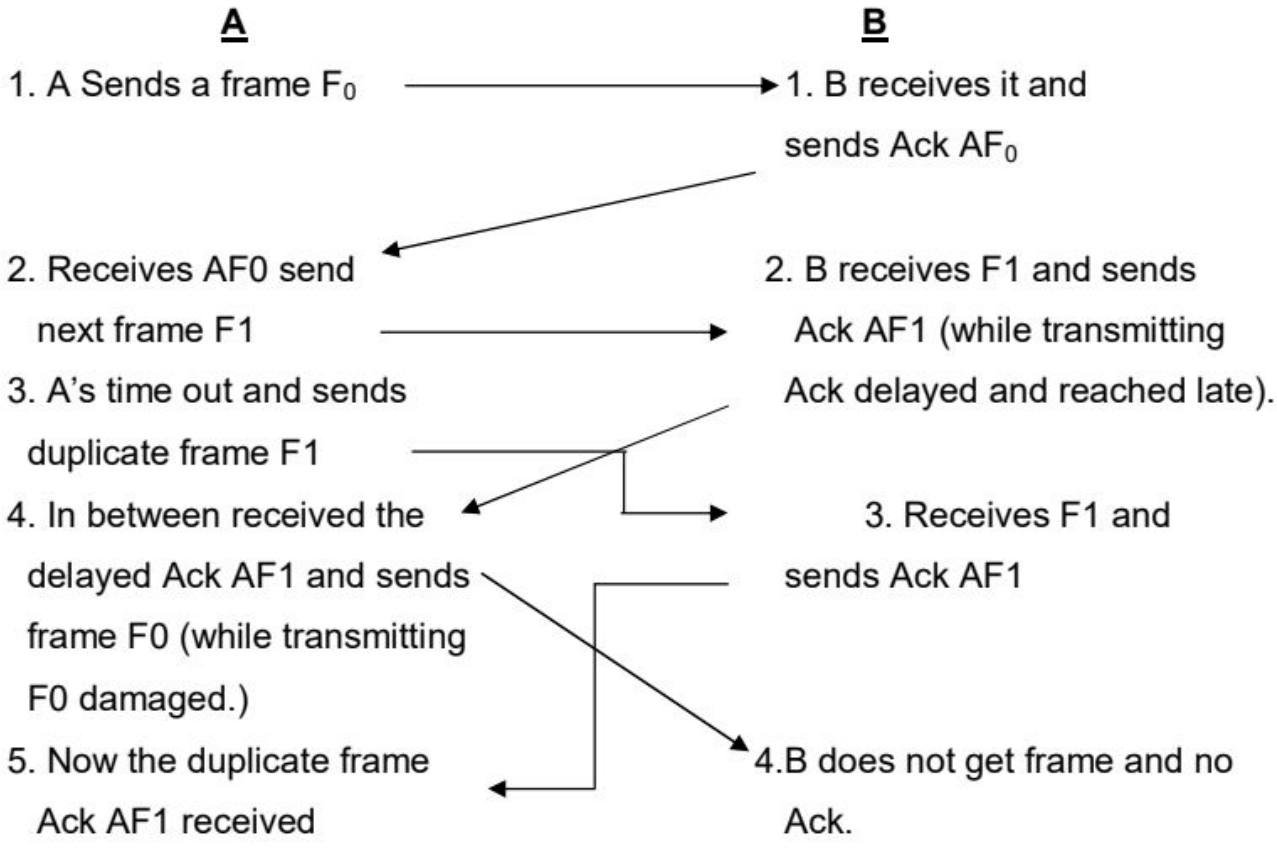


At this situation protocol fails because the receiver receives a duplicate frame and there is no way to find out whether the receiver frame is original or duplicate. So the protocol fails at this situation.

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.

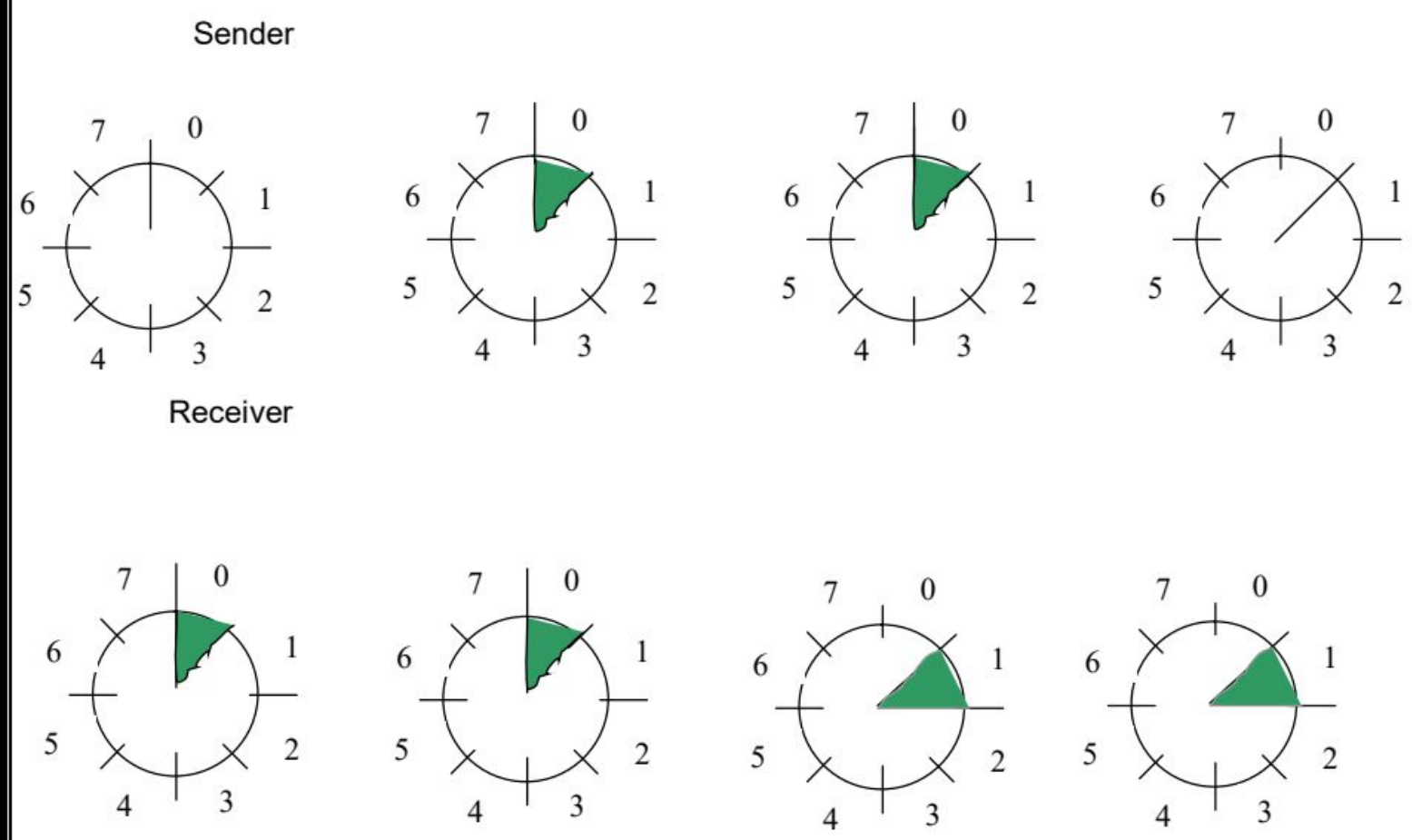
SLIDING WINDOW PROTOCOLS

In all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually $2^n - 1$ so the sequence number fits nicely in an n-bit field. The stop-and-wait sliding window protocol uses n=1, restricting the sequence numbers to 0 and 1, but more sophisticated versions can use arbitrary n.

The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall with in the sending window. Similarly the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept. The sender's window and the receiver's window need not have the same lower and upper limits, or even have the same size. In some protocols they are fixed in size, but in others they can grow or shrink as frames are sent and received.

The sequence numbers with in the sender's window represent frames sent but as yet not acknowledged. Whenever a new packet arrives from the network layer, it is given the next

highest sequence number, and the upper edge of the window is advanced by one. When an acknowledgement comes in, the lower edge is advanced by one. In this way the continuously maintains a list of unacknowledged frames.



(a) Initially (b) After the first frame has been sent c) After the first frame has been received. d) After the first acknowledgement has been received.

PIPELINING

1. Upto now we made the assumption that the transmission time required for a frame to arrive at the receiver plus the transmission time for the ack to come back is negligible.

2. Sometimes this is not true, when there is a long round trip propagation time is there.

3. In these cases round trip propagation time can have important implications for the efficiency of the bandwidth utilization.

Consider the below example.

Let the channel capacity b = 50Kbps.

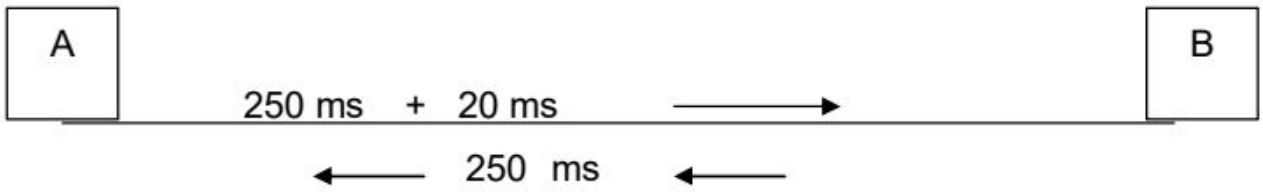
round trip propagation delay = 500ms

Frame size = 1000bits

Without considering the round trip propagation delay

For one frame the time taken will be = 1000/500 ms
= 20 ms

Considering the round trip propagation delay



For one frame the time taken will be = 500 ms +20 ms
= 270 ms

The channel utilization = (20/520)*100 = 4%

i.e. We are wasting 96% of channel time. To overcome this problem we will go for a technique called **PIPELIING.**

In this technique, the sender is allowed to transmit upto 'w ' frames before blocking, instead of just 1.With an appropriate choice of w the sender will be able to continuously

transmit frames for a time equal to the round trip transmit time without filling up the window.

In the above example w would be at least 26 frames. (520/20 = 26 frames)

By the time it has finished sending 26 frames, at t=520 ms, the ack for frame 0 will have just arrived. Thereafter ack will arrive every 20 ms, so the sender always gets permission to continue just when it needs it.

Hence, we can say the sender window size is 26.

Derivation:

Let the channel capacity = b Bps

Let the frame size = l bits

Let the round trip delay = R secs

To send one frame the time will be l/b secs

Due to round trip delay the time taken will be (l/b + R) Sec = l+Rb/b Sec

The channel utilization is l/b (l/b + R) Sec = (l / l + Rb) Sec

If l > bR the efficiency will be greater than 50%.
If l < bR the efficiency will be less than 50%.

If l = bR the efficiency will be 50%.

MEDIUM ACCESS CONTROL SUBLAYER (MAC)

Networks can be categories in to two ways

a) Point to point b) Broad cast channel

- In broadcast network, the key issue is how to share the channel among several users.

- Ex *a conference call with five people*

-Broadcast channels are also called as multi-access channels or random access channels.

-Multi-access channel belong to a sublayer at the DL layer called the MAC sublayer.

The Channel Allocation problem:

a) **Static channel allocation** in LANs & MANs

i) **FDM** ii) **TDM**

Drawbacks: -1) Channel is wasted if one or more stations do not send data.

2) If users increases this will not support.

b) Dynamic channel allocation

i) Pure **ALOHA** & Slotted **ALOHA**

ii) **CSMA** — CSMA/CD, CSMA/CA

Pure ALOHA

-1970's Norman Abramson end his colleagues devised this method, used ground –based radio broad costing. This is called the **ALOHA** system.

-The basic idea, many users are competing for the use of a single shared channel.

-There are two versions of ALOHA: **Pure and Slotted**.

-Pure ALOHA does not require global time synchronization, where as in slotted ALOHA the time is divided into discrete slots into which all frames must fit.

-Let users transmit whenever they have data to be sent.

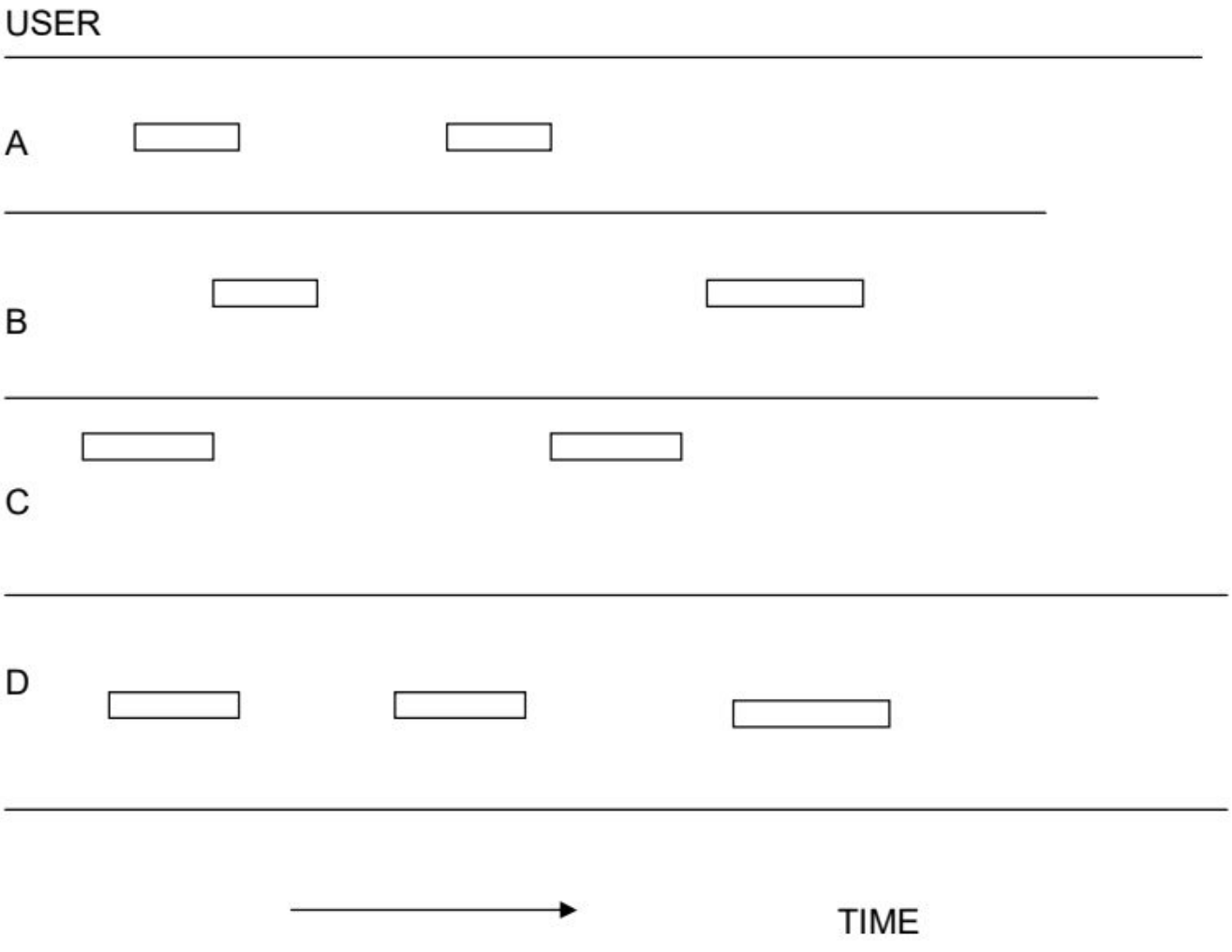
-There will be collisions and all collided frames will be damaged.

-Senders will know through feedback property whether the frame is destroyed or not by listening channel.

[-With a LAN it is immediate, with a satellite, it will take 270m sec.]

-If the frame was destroyed, the sender waits random amount of time and again sends the frame.

-The waiting time must be random otherwise the same frame will collide over and over.



Frames are transmitted at completely arbitrary times

-Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be destroyed.

-We have to find out what is the efficiency of an ALOHA channel?

-Let us consider an infinite collection of interactive users sitting at their systems (stations).

-A user will always in two states **typing or waiting**.

-Let the 'Frame time' denotes the time required to transmit one fixed length frame.

-Assume that infinite populations of users are generating new frames according to possion distribution with mean N frames per frame time.

-If N>1 users are generating frames at a higher rate than the channel can handle.

-For reasonable throughput 0<N<1.

-In addition to new frames, the station also generates retransmission of frames.

-Old and new frames are G per frame time.

-$G \geq N$

-At low load there will be few collisions, so G ~ N

-Under all loads, the throughput $S = GP_o$, where P_o is the probability that a frame does not suffer a collision.

-A frame will not suffer a collision if no other frames are sent with one frame time of its start.

-Let 't' be the time required to send a frame.

-If any other user has generated a frame between time t_o and t_o+t, the end of that frame will collide with the beginning of the shaded frame.

-Similarly, any other frame started b/w t_o+t and t_o+2t will bump into the end of the shaded frame.

-The probability that 'k' frames are generated during a given frame time is given by the possion distribution:

$$P_r[k] = \frac{G^k e^{-G}}{k!}$$

-The probability of zero frames is just e^{-G}

-In an interval two frame times long, the mean number at frames generated is 2G.

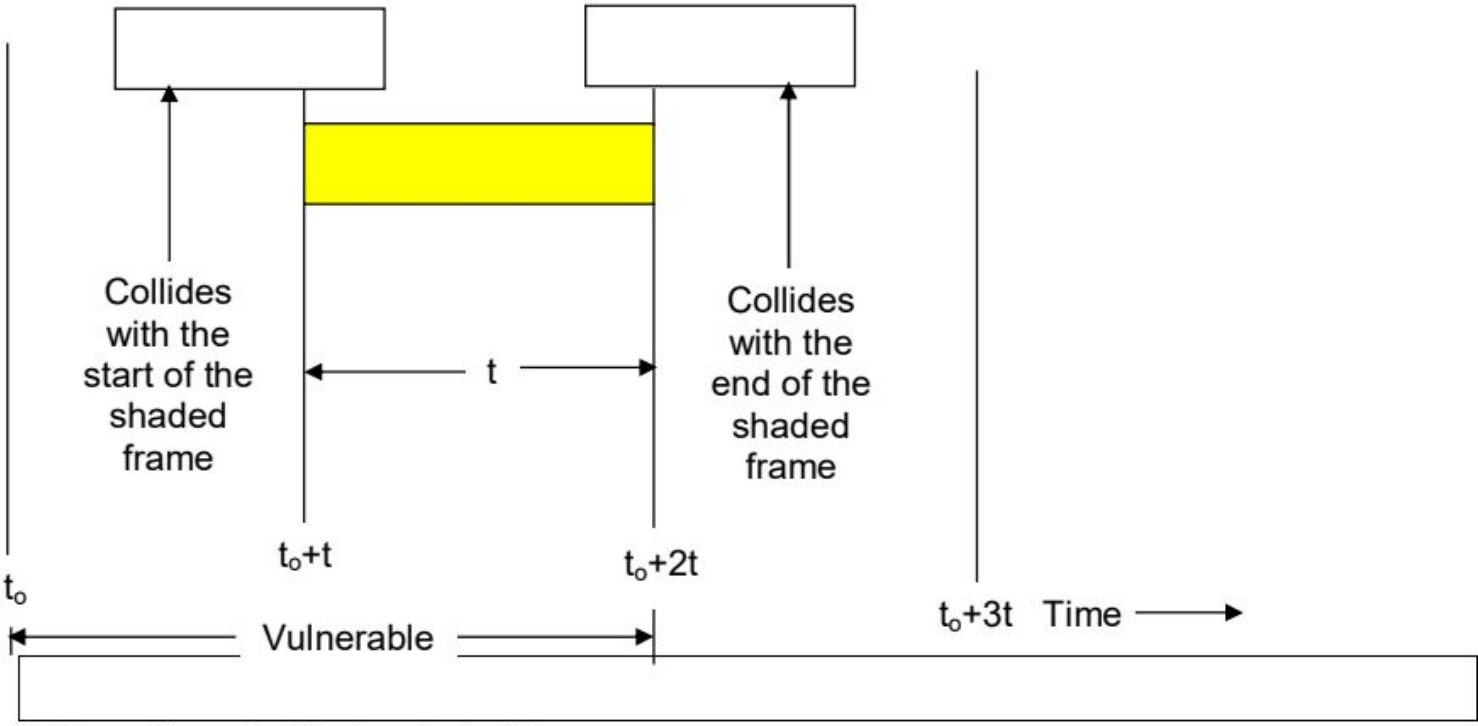
-The probability at no other traffic being initiated during the entire vulnerable period is given by

$$P_o = e^{-2G}$$

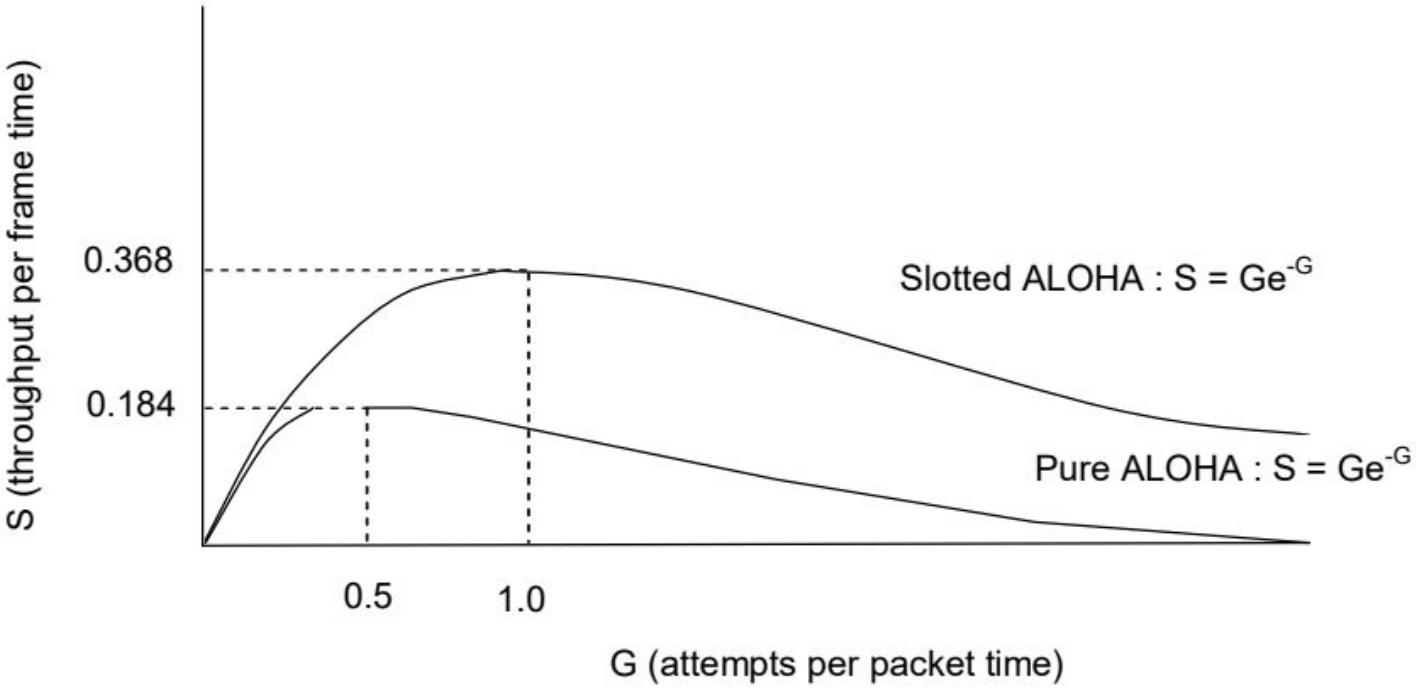
$$S = Ge^{-2G} \qquad [S=GP_o]$$

The Maximum through put occurs at G=0.5 with S=1/2e = 0.184

The channel utilization at pure ALOHA =18%.



Vulnerable period for the shaded frame



Throughput versus offered traffic for ALOHA systems

Slotted ALOHA

-In 1972, Roberts' devised a method for doubling the capacity of ALOHA system.

-In this system the time is divided into discrete intervals, each interval corresponding to one frame.

-One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

-In Roberts' method, which has come to be known as slotted ALOHA, in contrast to Abramson's pure ALOHA; a computer is not permitted to send whenever a carriage return is typed.

-Instead, it is required to wait for the beginning of the next slot.

-Thus the continuous pure ALOHA is turned into a discrete one.

-Since the vulnerable period is now halved, the of no other traffic during the same slot as our test frame is e^{-G} which leads to

$S = Ge^{-G}$

- At G=1, slotted ALOHA will have maximum throughput.
- So S=1/e or about 0.368, twice that of pure ALOHA.
- The channel utilization is 37% in slotted ALOHA.

Carrier Sense Multiple Access Protocols

Protocols in which stations listen for a carrier (transmission) and act accordingly are called carries sense protocols.

Persistent CSMA

When a station has data to send, it first listens to the channel to see if any one else is transmitting at that moment. If the channel is busy, the station waits until it become idle. When the station detects an idle channel, it transmits a frame. If a collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1-persistent also because the station transmits with a probability of 1 when it finds the channel idle.

The propagation delay has an important effect on the performance of the protocol. The longer the propagation delay the worse the performance of the protocol.

Even if the propagation delay is zero, there will be collisions. If two stations listen the channel, that is idle at the same, both will send frame and there will be collision.

Non persistent CSMA

In this, before sending, a station sense the channel. If no one else is sending, the station begins doing so it self. However, if the channel is busy, the station does not continually sense it but it waits a random amount of time and repeats the process.
This algorithms leads to better channel utilization but longer delays then 1-persistent CSMA.

With persistent CSMA, what happens if two stations become active when a third station is busy? Both wait for the active station to finish, then simultaneously launch a packet, resulting a collision. There are two ways to handle this problem.
a) P-persistent CSMA b) exponential backoff.

P-persistent CSMA

The first technique is for a waiting station not to launch a packet immediately when the channel becomes idle, but first toss a coin, and send a packet only if the coin comes up heads. If the coin comes up tails, the station waits for some time (one slot for slotted CSMA), then repeats the process. The idea is that if two stations are both waiting for the medium, this reduces the chance of a collision from 100% to 25%. A simple generalization of the scheme is to use a biased coin, so that the probability of sending a packet when the medium becomes idle is not 0.5, but p, where $0< p < 1$. We call such a scheme **P-persistent CSMA**. The original scheme, where p=1, is thus called 1-persitent CSMA.

Exponential backoff

The key idea is that each station, after transmitting a packet, checks whether the packet transmission was successful. Successful transmission is indicated either by an explicit acknowledgement from the receiver or the absence of a signal from a collision detection circuit. If the transmission is successful, the station is done. Otherwise, the station retransmits the packet, simultaneously realizing that at least one other station is also contending for the medium. To prevent its retransmission from colliding with the other station's retransmission, each station backs off (that is, idles) for a random time chosen from the interval

[0,2*max-propagation_delay] before retransmitting its packet. If the retransmission also fails, then the station backs off for a random time in the interval [0,4* max_propagation_delay], and tries again. Each subsequent collision doubles the backoff interval length, until the retransmission finally succeeds. On a successful transmission, the backoff interval is reset to the initial value. We call this type of backoff exponential backoff.

CSMA/CA

In many wireless LANS, unlike wired LANS, the station has no idea whether the packet collided with another packet or not until it receives an acknowledgement from receiver. In this situation, collisions have a greater effect on performance than with CSMA/CD, where colliding packets can be quickly detected and aborted. Thus, it makes sense to try to avoid collisions, if possible. CSMA/CA is basically p-persistence, with the twist that when the medium becomes idle, a station must wait for a time called the interframe spacing or IFS before contending for a slot. A station gets a higher priority if it is allocated smaller inter frame spacing.

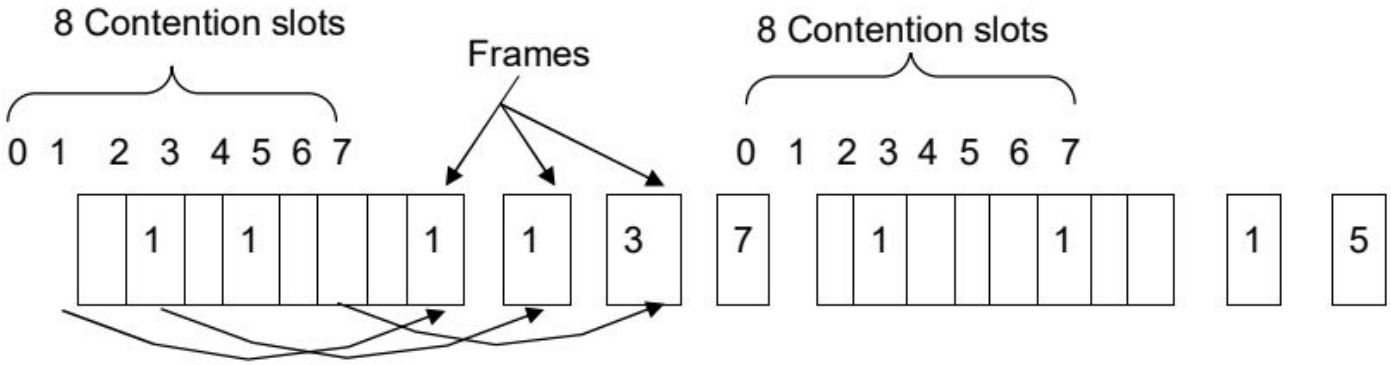
When a station wants to transmit data, it first checks if the medium is busy. If it is, it continuously senses the medium, waiting for it to become idle. When the medium becomes idle, the station first waits for an interframe spacing corresponding to its priority level, then sets a contention timer to a time interval randomly selected in the range [0,CW], where CW is a predefined contention window length. When this timer expires, it transmits a packet and waits for the receiver to send an ack. If no ack is received, the packet is assumed lost to collision, and the source tries again, choosing a contention timer at random from an interval twice as long as the one before(binary exponential backoff). If the station senses that another station has begun transmission while it was waiting for the expiration of the contention timer, it does not reset its timer, but merely freezer it, and restarts the countdown when the packet completes transmission. In this way, stations that happen to choose a longer timer value get higher priority in the next round of contention.

Collision-Free Protocols

A Bit-Map Protocol

In the basic bit-map method, each contention period consists of exactly N slots. If station 0 has a frame to send, it transmits a 1 bit during the zeroth slot. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the

opportunity to transmit a 1during slot 1, but only if it has a frame queued. In general, station j may announce the fact that it has a frame to send by inserting a 1 bit into slot j. after all N slots have passed by, each station has complete knowledge of which stations with to transmit.



The basic bit-map protocol

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another N bit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again. Protocols like this in which the desire to transmit is broadcast before the actual transmission are called reservation protocols.

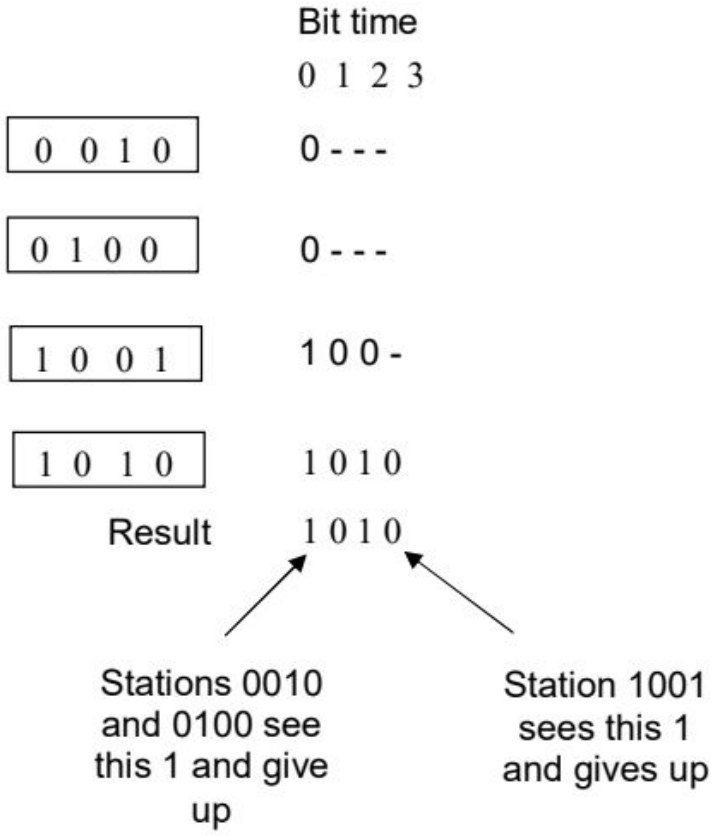
Binary Countdown

A problem with the basic bit-map protocol is that the overhead is 1 bit per station. A station wanting to use the channel now 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 address position from different stations are BOOLEAN ORed together. We will call this protocol binary countdown. It is used in Datakit.

As soon as a station sees that a high-order bit position that is 0 in its address has been overwritten with a 1, it gives up. For example, if station 0010,0100,1001, and 1010 are all trying to get the channel, in the first bit time the stations transmit 0,0,1, and 1, respectively. Stations 0010 and 0100 see the 1 and know that a higher-numbered station is competing for the channel, so they give up for the current round. Stations 1001 and 1010 continue.

The next bit is 0, and both stations continue. The next bit is 1, so station 1001 gives up. The winner is station 1010, because it has the highest address. After winning the bidding, it may now transmit a frame, after which another bidding cycle starts.

The binary countdown protocol. A dash indicates silence



IEEE Standard 802 for LANS and MANS

The IEEE 802.3 is for a 1-persistent CSMA/CD LAN. Xerox built a 2.94 Mbps CSMA/CD system to connect over 100 personal workstations on 1-Km cable. This system was called Ethernet through which electromagnetic radiation was once thought to propagate. Xerox DEC and Intel came with another standard for 100 Mbps Ethernet. This differs from old one that it runs at speeds from 1 to 10 Mbps on various media. The second difference between these two is in one header (802.3 length field is used for packet type in Ethernet).