Data Link Layer

Data link layer can be characterized by two types of layers:

1. Medium Access Layer (MAL)
2. Logical Link Layer

Aloha Protocols

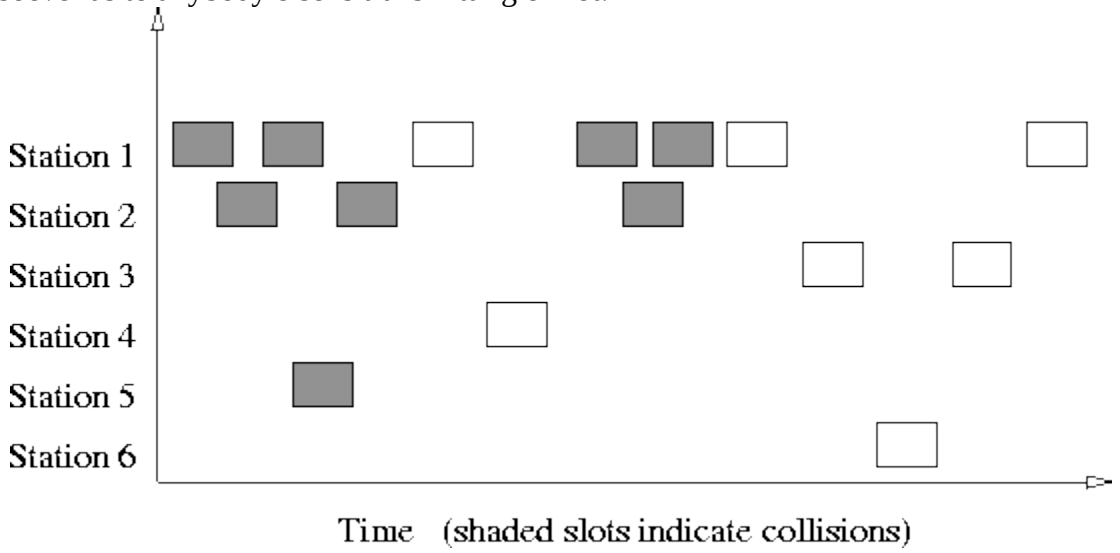
History

The Aloha protocol was designed as part of a project at the University of Hawaii. It provided data transmission between computers on several of the Hawaiian Islands using radio transmissions.

- Communications was typically between remote stations and a central sited named Menehune or vice versa.
- All messages to the Menehune were sent using the same frequency.
- When it received a message intact, the Menehune would broadcast an ack on a distinct outgoing frequency.
- The outgoing frequency was also used for messages from the central site to remote computers.
- All stations listened for message on this second frequency.

Pure Aloha

Pure Aloha is an unslotted, fully-decentralized protocol. It is extremely simple and trivial to implement. The ground rule is - "when you want to talk, just talk!" So, a node which wants to transmit, will go ahead and send the packet on its broadcast channel, with no consideration whatsoever as to anybody else is transmitting or not.



One serious drawback here is that, you don’t know whether what you are sending has been received properly or not (so as to say, "whether you've been heard and understood?"). To resolve this, in Pure Aloha, when one node finishes speaking, it expects an acknowledgement in a finite amount of time - otherwise it simply retransmits the data. This scheme works well in small networks where the load is not high. But in large, load intensive networks where many nodes may want to transmit at the same time, this scheme fails miserably. This led to the development of Slotted Aloha.

Slotted Aloha

This is quite similar to Pure Aloha, differing only in the way transmissions take place. Instead of transmitting right at demand time, the sender waits for some time. This delay is specified as follows - the timeline is divided into equal slots and then it is required that transmission should take place only at slot boundaries. To be more precise, the slotted-Aloha makes the following assumptions:

- All frames consist of exactly L bits.
- Time is divided into slots of size L/R seconds (i.e., a slot equals the time to transmit one frame).
- Nodes start to transmit frames only at the beginnings of slots.
- The nodes are synchronized so that each node knows when the slots begin.
- If two or more frames collide in a slot, then all the nodes detect the collision event before the slot ends.

In this way, the number of collisions that can possibly take place is reduced by a huge margin. And hence, the performance become much better compared to Pure Aloha. Collisions may only take place with nodes that are ready to speak at the same time. But nevertheless, this is a substantial reduction.

Carrier Sense Multiple Access Protocols

In both slotted and pure ALOHA, a node's decision to transmit is made independently of the activity of the other nodes attached to the broadcast channel. In particular, a node neither pays attention to whether another node happens to be transmitting when it begins to transmit, nor stops transmitting if another node begins to interfere with its transmission. As humans, we have human protocols that allow allows us to not only behave with more civility, but also to decrease the amount of time spent "colliding" with each other in conversation and consequently increasing the amount of data we exchange in our conversations. Specifically, there are two important rules for polite human conversation:

1. **Listen before speaking:** If someone else is speaking, wait until they are done. In the networking world, this is termed carrier sensing - a node listens to the channel before transmitting. If a frame from another node is currently being transmitted into the channel, a node then waits ("backs off") a random amount of time and then again senses the channel. If the channel is sensed to be idle, the node then begins frame transmission. Otherwise, the node waits another random amount of time and repeats this process.
2. **If someone else begins talking at the same time, stop talking.** In the networking world, this is termed collision detection - a transmitting node listens to the channel while it is transmitting. If it detects that another node is transmitting an interfering frame, it stops transmitting and uses some protocol to determine when it should next attempt to transmit.

It is evident that the end-to-end channel propagation delay of a broadcast channel - the time it takes for a signal to propagate from one of the the channel to another - will play a crucial role in determining its performance. The longer this propagation delay, the larger the chance that a carrier-sensing node is not yet able to sense a transmission that has already begun at another node in the network.

CSMA- Carrier Sense Multiple Access

This is the simplest version CSMA protocol as described above. It does not specify any collision detection or handling. So collisions might and WILL occur and clearly then, this is not a very good protocol for large, load intensive networks.

So, we need an improvement over CSMA - this led to the development of CSMA/CD.

CSMA/CD- CSMA with Collision Detection

In this protocol, while transmitting the data, the sender simultaneously tries to receive it. So, as soon as it detects a collission (it doesn't receive its own data) it stops transmitting. Thereafter, the node waits for some time interval before attempting to transmit again. Simply put, **"listen while you talk"**. But, how long should one wait for the carrier to be freed? There are three schemes to handle this:

1. **1-Persistent:** In this scheme, transmission proceeds immediately if the carrier is idle. However, if the carrier is busy, then sender continues to sense the carrier until it becomes

idle. The main problem here is that, if more than one transmitters are ready to send, a collision is GUARANTEED!!

2. **Non-Persistent:** In this scheme, the broadcast channel is not monitored continuously. The sender polls it at random time intervals and transmits whenever the carrier is idle. This decreases the probability of collisions. But, it is not efficient in a low load situation, where number of collisions are anyway small. The problems it entails are:
    - If back-off time is too long, the idle time of carrier is wasted in some sense
    - It may result in long access delays
3. **p-Persistent:** Even if a sender finds the carrier to be idle, it uses a probabilistic distribution to determine whether to transmit or not. Put simply, "toss a coin to decide". If the carrier is idle, then transmission takes place with a probability p and the sender waits with a probability 1-p. This scheme is a good trade off between the Non-persistent and 1-persistent schemes. So, for low load situations, p is high (example: 1-persistent); and for high load situations, p may be lower. Clearly, the value of p plays an important role in determining the performance of this protocol. Also the same p is likely to provide different performance at different loads.

CSMA/CD doesn't work in some wireless scenarios called **"hidden node"** problems. Consider a situation, where there are 3 nodes - A, B and C communicating with each other using a wireless protocol. Moreover, B can communicate with both A and C, but A and C lie outside each other's range and hence can't communicate directly with each other. Now, suppose both A and C want to communicate with B simultaneously. They both will sense the carrier to be idle and hence will begin transmission, and even if there is a collision, neither A nor C will ever detect it. B on the other hand will receive 2 packets at the same time and might not be able to understand either of them. To get around this problem, a better version called CSMA/CA was developed, especially for wireless applications.

CSMA with Collision Avoidance

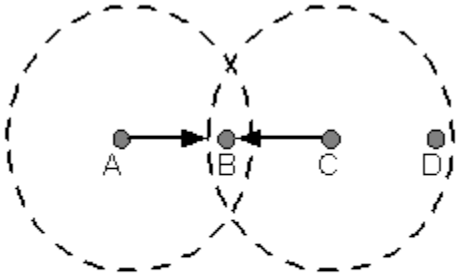
We have observed that CSMA/CD would break down in wireless networks because of hidden node and exposed nodes problems. We will have a quick recap of these two problems through examples.

Hidden Node Problem

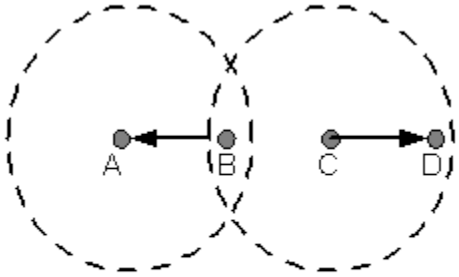
In the case of wireless network it is possible that A is sending a message to B, but C is out of its range and hence while "listening" on the network it will find the network to be free and might try to send packets to B at the same time as A. So, there will be a collision at B. The problem can be looked upon as if A and C are hidden from each other. Hence it is called the "hidden node problem".

Exposed Node Problem

If C is transmitting a message to D and B wants to transmit a message to A, B will find the network to be busy as B hears C transmitting. Even if B would have transmitted to A, it would not have been a problem at A or D. CSMA/CD would not allow it to transmit message to A, while the two transmissions could have gone in parallel.



Hidden node problem



Exposed node problem

Addressing hidden node problem (CSMA/CA)

Consider the figure above. Suppose A wants to send a packet to B. Then it will first send a small packet to B called **"Request to Send" (RTS)**. In response, B sends a small packet to A called **"Clear to Send" (CTS)**. Only after A receives a CTS, it transmits the actual data. Now, any of the nodes which can hear either CTS or RTS assume the network to be busy. Hence even if some other node which is out of range of both A and B sends an RTS to C (which can hear at least one of the RTS or CTS between A and B), C would not send a CTS to it and hence the communication would not be established between C and D.

One issue that needs to be addressed is how long the rest of the nodes should wait before they can transmit data over the network. The answer is that the RTS and CTS would carry some information about the size of the data that B intends to transfer. So, they can calculate time that would be required for the transmission to be over and assume the network to be free after that.Another interesting issue is what a node should do if it hears RTS but not a corresponding CTS. One possibility is that it assumes the recipient node has not responded and hence no transmission is going on, but there is a catch in this. It is possible that the node hearing RTS is just on the boundary of the node sending CTS. Hence, it does hear CTS but the signal is so deteriorated that it fails to recognize it as a CTS. Hence to be on the safer side, a node will not start transmission if it hears either of an RTS or a CTS.

The assumption made in this whole discussion is that if a node X can send packets to a node Y, it can also receive a packet from Y, which is a fair enough assumption given the fact that we are talking of a local network where standard instruments would be used. If that is not the case additional complexities would get introduced in the system.

Does CSMA/CD work universally in the wired networks?

The problem of range is there in wired networks as well in the form of deterioration of signals. Normally to counter this, we use repeaters, which can regenerate the original signal from a

deteriorated one. But does that mean that we can build as long networks as we want with repeaters. The answer, unfortunately, is NO! The reason is the beyond a certain length CSMA/CD will break down.

The mechanism of collision detection which CSMA/CD follows is through listening while talking. What this means is so long as a node is transmitting the packet, it is listening on the cable. If the data it listens to is different from the data it is transmitting it assumes a collision. Once it has stopped transmitting the packet, and has not detected collision while transmission was going on, it assumes that the transmission was successful. The problem arises when the distance between the two nodes is too large. Suppose A wants to transmit some packet to B which is at a very large distance from B. Data can travel on cable only at a finite speed (usually 2/3c, c being the speed of light). So, it is possible that the packet has been transmitted by A onto the cable but the first bit of the packet has not yet reached B. In that case, if a collision occurs, A would be unaware of it occurring. Therefore there is problem in too long a network.

Let us try to parameterize the above problem. Suppose "t" is the time taken for the node A to transmit the packet on the cable and "T" is the time , the packet takes to reach from A to B. Suppose transmission at A starts at time t0. In the worst case the collision takes place just when the first packet is to reach B. Say it is at t0+T-e (e being very small). Then the collision information will take T-e time to propagate back to A. So, at t0+2(T-e) A should still be transmitting. Hence, for the correct detection of collision (ignoring e)

$$t > 2T$$

t increases with the number of bits to be transferred and decreases with the rate of transfer (bits per second). T increases with the distance between the nodes and decreases with the speed of the signal (usually 2/3c). We need to either keep t large enough or T as small. We do not want to live with lower rate of bit transfer and hence slow networks. We can not do anything about the speed of the signal. So what we can rely on is the minimum size of the packet and the distance between the two nodes. Therefore, we fix some minimum size of the packet and if the size is smaller than that, we put in some extra bits to make it reach the minimum size. Accordingly we fix the maximum distance between the nodes. Here too, there is a tradeoff to be made. We do not want the minimum size of the packets to be too large since that wastes lots of resources on cable. At the same time we do not want the distance between the nodes to be too small. Typical minimum packet size is 64 bytes and the corresponding distance is 2-5 kilometers.

Collision Free Protocols

Although collisions do not occur with CSMA/CD once a station has unambiguously seized the channel, they can still occur during the contention period. These collisions adversely affect the efficiency of transmission. Hence some protocols have been developed which are contention free.

Bit-Map Method

In this method, there N slots. If node 0 has a frame to send, it transmits a 1 bit during the first slot. No other node is allowed to transmit during this period. Next node 1 gets a chance to transmit 1 bit if it has something to send, regardless of what node 0 had transmitted. This is done for all the nodes. In general node j may declare the fact that it has a frame to send by inserting a 1 into slot j. Hence after all nodes have passed, each node has complete knowledge of who wants to send a frame. Now they begin transmitting in numerical order. Since everyone knows who is transmitting and when, there could never be any collision.

The basic problem with this protocol is its inefficiency during low load. If a node has to transmit and no other node needs to do so, even then it has to wait for the bitmap to finish. Hence the bitmap will be repeated over and over again if very few nodes want to send wasting valuable bandwidth.

Binary Countdown

In this protocol, a node which wants to signal that it has a frame to send does so by writing its address into the header as a binary number. The arbitration is such that as soon as a node sees that a higher bit position that is 0 in its address has been overwritten with a 1, it gives up. The final result is the address of the node which is allowed to send. After the node has transmitted the whole process is repeated all over again. Given below is an example situation.

Nodes	Addresses
A	0010
B	0101
C	1010
D	1001

	1010

Node C having higher priority gets to transmit. The problem with this protocol is that the nodes with higher address always wins. Hence this creates a priority which is highly unfair and hence undesirable.

Limited Contention Protocols

Both the type of protocols described above - Contention based and Contention - free has their own problems. Under conditions of light load, contention is preferable due to its low delay. As the load increases, contention becomes increasingly less attractive, because the overload associated with channel arbitration becomes greater. Just the reverse is true for contention - free protocols. At low load, they have high delay, but as the load increases , the channel efficiency improves rather than getting worse as it does for contention protocols.

Obviously it would be better if one could combine the best properties of the contention and contention - free protocols, that is, protocol which used contention at low loads to provide low delay, but used a cotention-free technique at high load to provide good channel efficiency. Such protocols do exist and are called Limited contention protocols.

It is obvious that the probability of some station acquiring the channel could only be increased by decreasing the amount of competition. The limited contention protocols do exactly that. They first divide the stations up into (not necessarily disjoint) groups. Only the members of group 0 are permitted to compete for slot 0. The competition for acquiring the slot within a group is contention based. If one of the members of that group succeeds, it acquires the channel and transmits a frame. If there is collision or no node of a particular group wants to send then the members of the next group compete for the next slot. The probability of a particular node is set to a particular value (optimum).

Adaptive Tree Walk Protocol

The following is the method of adaptive tree protocol. Initially all the nodes are allowed to try to acquire the channel. If it is able to acquire the channel, it sends its frame. If there is collision then the nodes are divided into two equal groups and only one of these groups competes for slot 1. If one of its members acquires the channel then the next slot is reserved for the other group. On the other hand, if there is a collision then that group is again subdivided and the same process is followed. This can be better understood if the nodes are thought of as being organised in a binary tree as shown in the following figure.

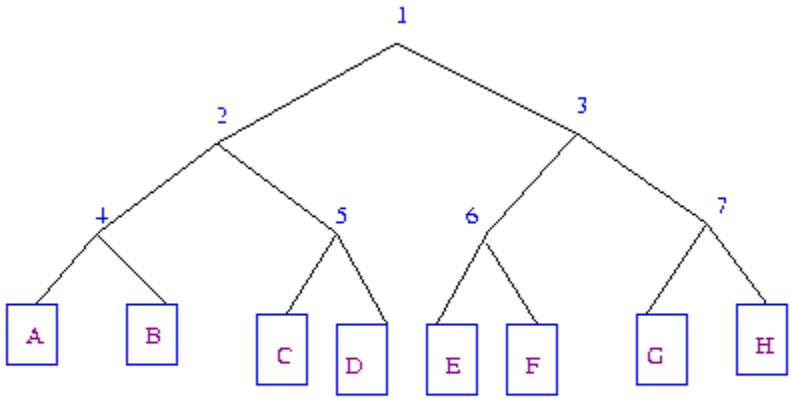


Fig. Adaptive Tree Walk abstraction of nodes in binary tree.

Many improvements could be made to the algorithm. For example, consider the case of nodes G and H being the only ones wanting to transmit. At slot 1 a collision will be detected and so 2 will be tried and it will be found to be idle. Hence it is pointless to probe 3 and one should directly go to 6,7.

IEEE 802.3 and Ethernet

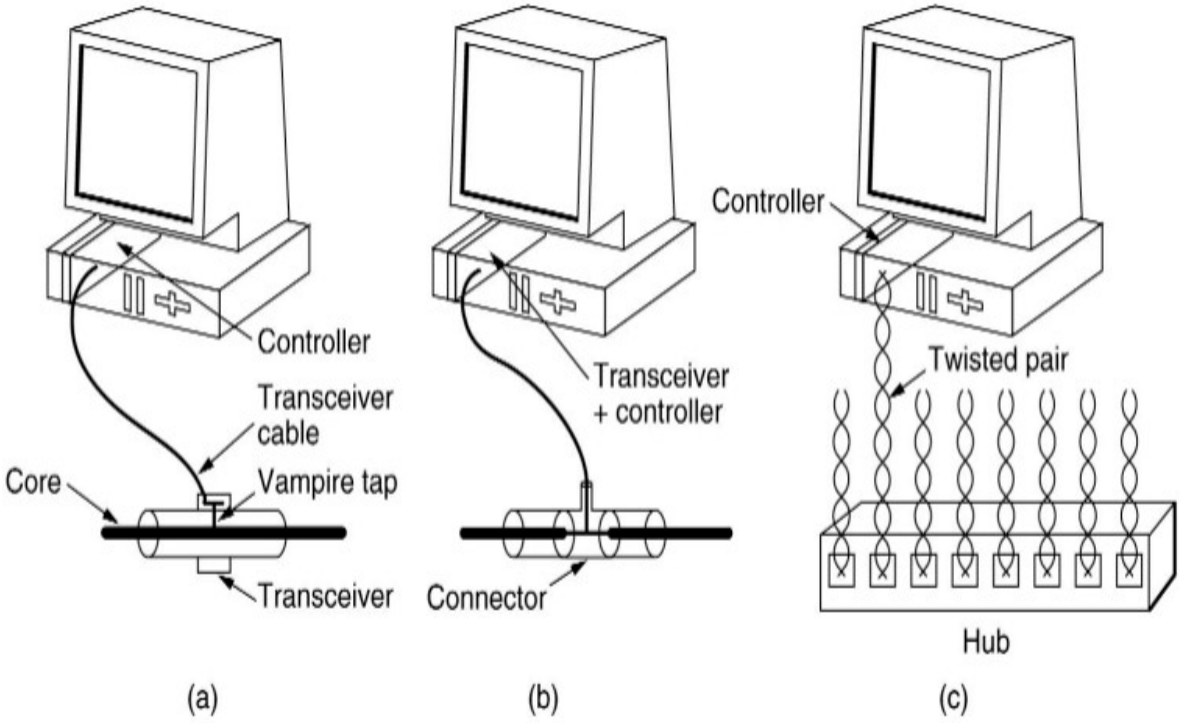
- Very popular LAN standard.
- Ethernet and IEEE 802.3 are distinct standards but as they are very similar to one another these words are used interchangeably.
- A standard for a 1-persistent CSMA/CD LAN.
- It covers the physical layer and MAC sublayer protocol.

Ethernet Physical Layer

A Comparison of Various Ethernet and IEEE 802.3 Physical-Layer Specifications

Characteristic	**Ethernet Value**	**IEEE 802.3 Values**					
		10Base5	**10Base2**	**10BaseT**	**10BaseF**	**10 Base - TX**	**100BaseT4**
Data rate (Mbps)	10	10	10	10	10	100	100
Signaling method	Baseband	Baseband	Baseband	Baseband	Baseband	Baseband	Baseband
Maximum segment length (m)	500	500	185	100	2,000	100	100
Media	50-ohm coax (thick)	50-ohm coax (thick)	50-ohm coax (thin)	Unshielded twisted-pair cable	Fiber-optic	Cat 5 UTP	Unshielded twisted-pair cable
Nodes/segment	100	100	30	1024	1024		
Topology	Bus	Bus	Bus	Star	Point-to-point	Bus	Bus

10Base5 means it operates at 10 Mbps, uses baseband signaling and can support segments of up to 500 meters. The 10Base5 cabling is popularly called the Thick Ethernet. Vampire taps are used for their connections where a pin is carefully forced halfway into the co-axial cable's core as shown in the figure below. The 10Base2 or Thin Ethernet bends easily and is connected using standard BNC connectors to form T junctions (shown in the figure below). In the 10Base-T scheme a different kind of wiring pattern is followed in which all stations have a twisted-pair cable running to a central hub (see below). The difference between the different physical connections is shown below:



(a) 10Base5 (b) 10Base2 (c) 10Base-T

All 802.3 baseband systems use Manchester encoding, which is a way for receivers to unambiguously determine the start, end or middle of each bit without reference to an external clock. There is a restriction on the minimum node spacing (segment length between two nodes) in 10Base5 and 10Base2 and that is 2.5 meter and 0.5 meter respectively. The reason is that if two nodes are closer than the specified limit then there will be very high current which may cause trouble in detection of signal at the receiver end. Connections from station to cable of 10Base5 (i.e. Thick Ethernet) are generally made using vampire taps and to 10Base2 (i.e. Thin Ethernet) are made using industry standard BNC connectors to form T junctions. To allow larger networks, multiple segments can be connected by repeaters as shown. A repeater is a physical layer device. It receives, amplifies and retransmits signals in either direction.

Note: To connect multiple segments, amplifier is not used because amplifier also amplifies the noise in the signal, whereas repeater regenerates signal after removing the noise.

IEEE 802.3 Frame Structure

Preamble (7 bytes)	Start of Frame Delimiter (1 byte)	Dest. Address (2/6 bytes)	Source Address (2/6 bytes)	Length (2 bytes)	802.2 Header+Data (46-1500 bytes)	Frame Checksum (4 bytes)

A brief description of each of the fields

- **Preamble:** Each frame starts with a preamble of 7 bytes, each byte containing the bit pattern 10101010. Manchester encoding is employed here and this enables the receiver's clock to synchronize with the sender's and initialize itself.
- **Start of Frame Delimiter:** This field containing a byte sequence 10101011 denotes the start of the frame itself.
- **Dest. Address: The** standard allows 2-byte and 6-byte addresses. Note that the 2-byte addresses are always local addresses while the 6-byte ones can be local or global.

  2-Byte Address - Manually assigned address

Individual(0)/Group(1) (1 bit)	Address of the machine (15 bits)

- ***6-Byte Address - Every Ethernet card with globally unique address***

Individual(0)/Group(1) (1 bit)	Universal(0)/Local(1) (1 bit)	Address of the machine (46 bits)

- **Multicast:** Sending to group of stations. This is ensured by setting the first bit in either 2-byte/6-byte addresses to 1.
  Broadcast: Sending to all stations. This can be done by setting all bits in the address field to 1.All Ethernet cards (Nodes) are a member of this group.
- **Source Address: Refer** to Dest. Address. Same holds true over here.
- **Length:** The Length field tells how many bytes are present in the data field, from a minimum of 0 to a maximum of 1500. The Data and padding together can be from 46bytes to 1500 bytes as the valid frames must be at least 64 bytes long, thus if data is less than 46 bytes the amount of padding can be found out by length field.
- **Data: Actually** this field can be split up into two parts - Data(0-1500 bytes) and Padding(0-46 bytes).
  Reasons for having a minimum length frame :
  1. To prevent a station from completing the transmission of a short frame before the first bit has even reached the far end of the cable, where it may collide with another frame. Note that the transmission time ought to be greater than twice the propagation time between two farthest nodes.

     transmission time for frame > 2*propagation time between two farthest nodes

  2. When a transceiver detects a collision, it truncates the current frame, which implies that stray bits and pieces of frames appear on the cable all the time. Hence to distinguish between valid frames from garbage, 802.3 states that the minimum

length of valid frames ought to be 64 bytes (from Dest. Address to Frame Checksum).

- **Frame Checksum:** It is a 32-bit hash code of the data. If some bits are erroneously received by the destination (due to noise on the cable), the checksum computed by the destination wouldn't match with the checksum sent and therefore the error will be detected. The checksum algorithm is a cyclic redundancy checksum (CRC) kind. The checksum includes the packet from Dest. Address to Data field.

Ethernet Frame Structure

Preamble (8 bytes)	Dest. Address (2/6 bytes)	Source Address (2/6 bytes)	Type (2 bytes)	Data (46-1500 bytes)	Frame Checksum (4 bytes)

A brief description of the fields which differ from IEEE 802.3

- **Preamble: The** *Preamble* and *Start of Frame Delimiter* are merged into one in Ethernet standard. However, the contents of the first 8 bytes remains the same in both.
- **Type: The** length field of IEEE 802.3 is replaced by Type field, which denotes the type of packet being sent viz. IP, ARP, RARP, etc. If the field indicates a value less than 1500 bytes then it is length field of 802.3 else it is the type field of Ethernet packet.

Truncated Binary Exponential Back off

In case of collision the node transmitting backs off by a random number of slots , each slot time being equal to transmission time of 512 bits (64 Byte- minimum size of a packet) in the following fashion:

No of Collision	Random No of slots
1st	0-1
2nd	0-3
3rd	0-7
|	|
|	|
10th	0-1023
---	---
11th	0-1023

12th	0-1023
|	|
16th	0-1023

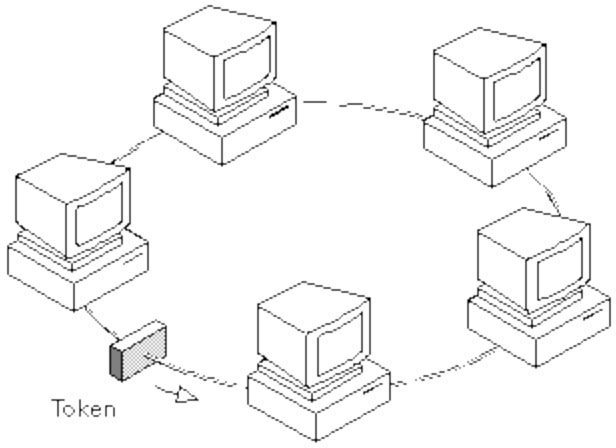
In general after i collisions a random number between 0-2^i-1 is chosen , and that number of slots is skipped. However, after 10 collisions have been reached the randomization interval is frozen at maximum of 1023 slots. After 16 collisions the controller reports failure back to the computer.

5-4-3 Rule

Each version of 802.3 has a maximum cable length per segment because long propagation time leads to difficulty in collision detection. To compensate for this the transmission time has to be increased which can be achieved by slowing down the transmission rate or increasing the packet size, neither of which is desirable. Hence to allow for large networks, multiple cables are connected via **repeaters**. Between any two nodes on an Ethernet network, there can be at most five segments, four repeaters and three populated segments (non-populated segments are those which do not have any machine connected between the two repeaters). This is known as the **5-4-3 Rule**.

IEEE 802.5: Token Ring Network

- Token Ring is formed by the nodes connected in ring format as shown in the diagram below. The principle used in the token ring network is that a token is circulating in the ring and whichever node grabs that token will have right to transmit the data.
- Whenever a station wants to transmit a frame it inverts a single bit of the 3-byte token which instantaneously changes it into a normal data packet. Because there is only one token, there can at most be one transmission at a time.
- Since the token rotates in the ring it is guaranteed that every node gets the token with in some specified time. So there is an upper bound on the time of waiting to grab the token so that starvation is avoided.
- There is also an upper limit of 250 on the number of nodes in the network.
- To distinguish the normal data packets from token (control packet) a special sequence is assigned to the token packet. When any node gets the token it first sends the data it wants to send, then recalculates the token.



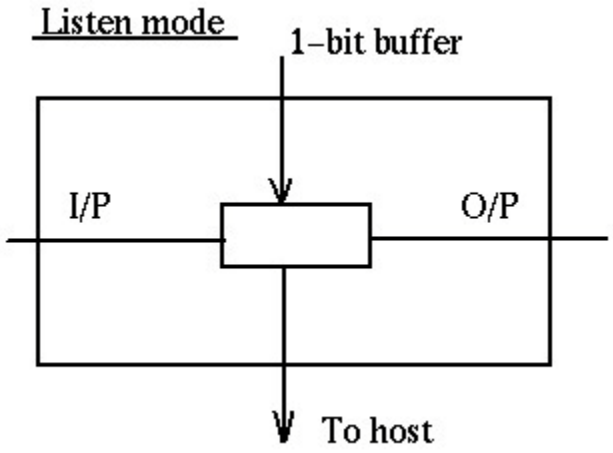
If a node transmits the token and nobody wants to send the data the token comes back to the sender. If the first bit of the token reaches the sender before the transmission of the last bit, then error situation arises. So to avoid this we should have:

propagation delay + transmission of n-bits (1-bit delay in each node) > transmission of the token time

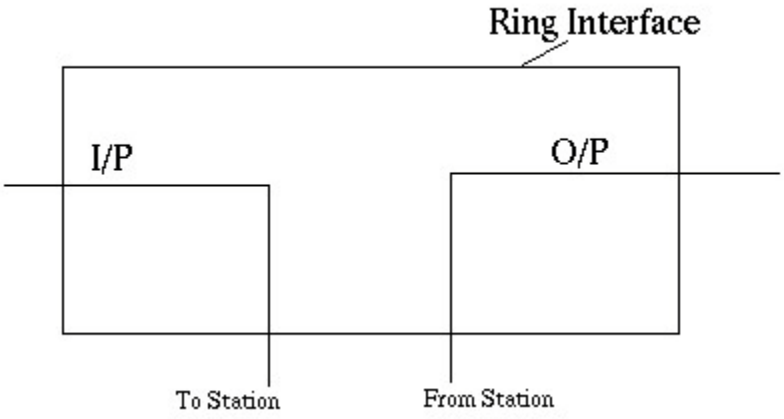
A station may hold the token for the token-holding time. which is 10 ms unless the installation sets a different value. If there is enough time left after the first frame has been transmitted to send more frames, then these frames may be sent as well. After all pending frames have been transmitted or the transmission frame would exceed the token-holding time, the station regenerates the 3-byte token frame and puts it back on the ring.

Modes of Operation

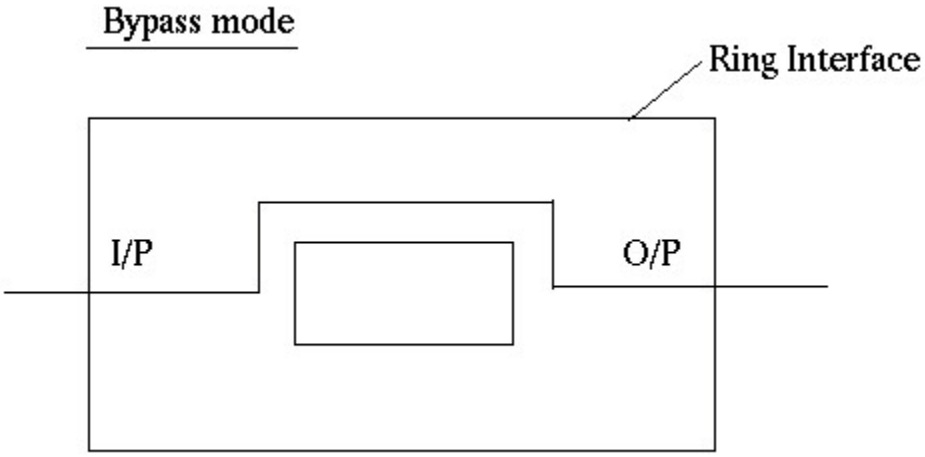
1. **Listen Mode:** In this mode the node listens to the data and transmits the data to the next node. In this mode there is a one-bit delay associated with the transmission.



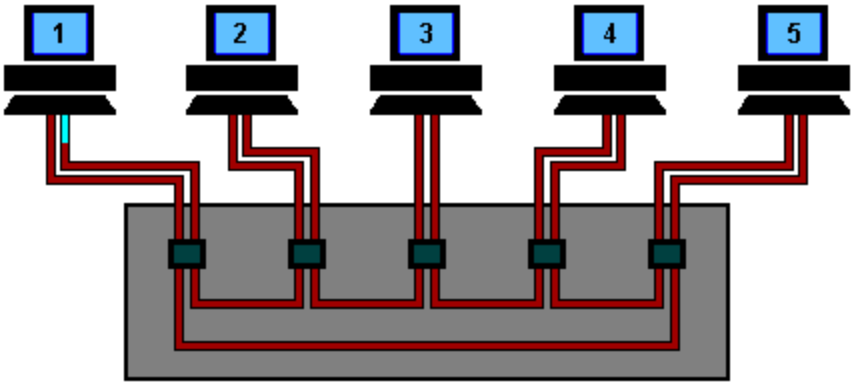
2. **Transmit Mode:** In this mode the node just discards the any data and puts the data onto the network.



3. **By-pass Mode:** In this mode reached when the node is down. Any data is just bypassed. There is no one-bit delay in this mode.



Token Ring Using Ring Concentrator



One problem with a ring network is that if the cable breaks somewhere, the ring dies. This problem is elegantly addressed by using a ring concentrator. A Token Ring concentrator simply changes the topology from a physical ring to a star wired ring. But the network still remains a ring logically. Physically, each station is connected to the ring concentrator (wire center) by a cable containing at least two twisted pairs, one for data to the station and one for data from the station. The Token still circulates around the network and is still controlled in the same manner, however, using a hub or a switch greatly improves reliability because the hub can automatically bypass any ports that are disconnected or have a cabling fault. This is done by having bypass relays inside the concentrator that are energized by current from the stations. If the ring breaks or station goes down, loss of the drive current will release the relay and bypass the station. The ring can then continue operation with the bad segment bypassed.

Who should remove the packet from the ring ?

There are 3 possibilities-

1. **The source itself removes the packet after one full round in the ring**.
2. **The destination removes it after accepting it**: This has two potential problems. Firstly, the solution won't work for broadcast or multicast, and secondly, there would be no way to acknowledge the sender about the receipt of the packet.
3. **Have a specialized node only to discard packets**: This is a bad solution as the specialized node would know that the packet has been received by the destination only when it receives the packet the second time and by that time the packet may have actually made about one and half (or almost two in the worst case) rounds in the ring.

Thus the first solution is adopted with the source itself removing the packet from the ring after a full one round. With this scheme, broadcasting and multicasting can be handled as well as the destination can acknowledge the source about the receipt of the packet (or can tell the source about some error).

Token Format

The token is the shortest frame transmitted (24 bit)
MSB (Most Significant Bit) is always transmitted first - as opposed to Ethernet

SD	AC	ED

SD = Starting Delimiter (1 Octet)
AC = Access Control (1 Octet)
ED = Ending Delimiter (1 Octet)

Starting Delimiter Format:

J	K	O	J	K	O	O	O

J = Code Violation
K = Code Violation

Access Control Format:

P	P	P	T	M	R	R	R

T=Token
T = 0 for Token
T = 1 for Frame
When a station with a Frame to transmit detects a token which has a priority equal to or less than the Frame to be transmitted, it may change the token to a start-of-frame sequence and transmit the Frame

P = Priority
Priority Bits indicate tokens priority, and therefore, which stations are allowed to use it. Station can transmit if its priority as at least as high as that of the token.

M = Monitor
The monitor bit is used to prevent a token whose priority is greater than 0 or any frame from continuously circulating on the ring. If an active monitor detects a frame or a high priority token with the monitor bit equal to 1, the frame or token is aborted. This bit shall be transmitted as 0 in all frame and tokens. The active monitor inspects and modifies this bit. All other stations shall repeat this bit as received.

R = Reserved bits
The reserved bits allow station with high priority Frames to request that the next token be issued at the requested priority.

Ending Delimiter Format:

J	K	1	J	K	1	1	E

J = Code Violation
K = Code Violation
I = Intermediate Frame Bit
E = Error Detected Bit

Frame Format:

MSB (Most Significant Bit) is always transmitted first - as opposed to Ethernet

SD	AC	FC	DA	SA	DATA	CRC	ED	FS

SD=Starting Delimiter(1 octet)
AC=Access Control(1 octet)
FC = Frame Control (1 Octet)
DA = Destination Address (2 or 6 Octets)
SA = Source Address (2 or 6 Octets)
DATA = Information 0 or more octets up to 4027
CRC = Checksum(4 Octets)
ED = Ending Delimiter (1 Octet)
FS=Frame Status

Starting Delimiter Format:

J	K	0	J	K	0	0	0

J = Code Violation
K = Code Violation

Access Control Format:

P	P	P	T	M	R	R	R

T=Token

T = “0” for Token,
T = “1” for Frame.

When a station with a Frame to transmit detects a token which has a priority equal to or less than the Frame to be transmitted, it may change the token to a start-of-frame sequence and transmit the Frame.

P = Priority
Bits Priority Bits indicate tokens priority, and therefore, which stations are allowed to use it. Station can transmit if its priority as at least as high as that of the token.

M = Monitor
The monitor bit is used to prevent a token whose priority is greater than 0 or any frame from continuously circulating on the ring. if an active monitor detects a frame or a high priority token with the monitor bit equal to 1, the frame or token is aborted. This bit shall be transmitted as 0 in all frame and tokens. The active monitor inspects and modifies this bit. All other stations shall repeat this bit as received.

R = Reserved bits the reserved bits allow station with high priority Frames to request that the next token be issued at the requested priority

Frame Control Format:

F	F	CONTROL BITS (6 BITS)

FF= Type of Packet-Regular data packet or MAC layer packet
Control Bits= Used if the packet is for MAC layer protocol itself

Source and Destination Address Format:

The addresses can be of 2 bytes (local address) or 6 bytes (global address).

local address format:

I/G (1 BIT)	NODE ADDRESS (15 BITS)

alternatively

I/G (1 BIT)	RING ADDRESS (7 BITS)	NODE ADDRESS (8 BITS)

The first bit specifies individual or group address.

universal (global) address format:

I/G (1 BIT)	L/U (1 BIT)	RING ADDRESS (14 BITS)	NODE ADDRESS (32 BITS)

The first bit specifies individual or group address.
The second bit specifies local or global (universal) address.

local group addresses (16 bits):

I/G (1 BIT)	T/B(1 BIT)	GROUP ADDRESS (14 BITS)

The first bit specifies an individual or group address.
The second bit specifies traditional or bit signature group address.

Traditional Group Address: 2Exp14 groups can be defined.
Bit Signature Group Address: 14 grtoups are defined. A host can be a member of none or any number of them. For multicasting, those group bits are set to which the packet should go. For broadcasting, all 14 bits are set. A host receives a packet only if it is a member of a group whose corresponding bit is set to 1.

universal group addresses (16 bits):

I/G (1 BIT)	RING NUMBER	T/B (1 BIT)	GROUP ADDRESS (14 BITS)

The description is similar to as above.

Data Format:

No upper limit on amount of data as such, but it is limited by the token holding time.

Checksum:

The source computes and sets this value. Destination too calculates this value. If the two are different, it indicates an error, otherwise the data may be correct.

Frame Status:

It contains the A and C bits.

A bit set to 1: destination recognized the packet.
C bit set to 1: destination accepted the packet.

This arrangement provides an automatic acknowledgement for each frame. The A and C bits are present twice in the Frame Status to increase reliability in as much as they are not covered by the checksum.

Ending Delimiter Format:

J	K	1	J	K	1	I	E

J = Code Violation
K = Code Violation
I = Intermediate Frame Bit

If this bit is set to 1, it indicates that this packet is an intermediate part of a bigger packet, the last packet would have this bit set to 0.
E = Error Detected Bit
This bit is set if any interface detects an error.

This concludes our description of the token ring frame format.

Phase Jitter Compensation:

In a token ring the source starts discarding all it's previously transmitted bits as soon as they circumnavigate the ring and reach the source. Hence, it's not desirable that while a token is being sent some bits of the token which have already been sent become available at the incoming end of the source. This behavior though is desirable in case of data packets which ought to be drained from the ring once they have gone around the ring. To achieve the aforesaid behavior with respect to tokens, we would like the ring to hold at least 24 bits at a time. How do we ensure this?

Each node in a ring introduces a 1 bit delay. So, one approach might be to set the minimum limit on the number of nodes in a ring as 24. But, this is not a viable option. The actual solution is as follows. We have one node in the ring designated as **"monitor"**. The monitor maintains a 24 bits buffer with help of which it introduces a 24 bit delay. The catch here is what if the clocks of nodes following the source are faster than the source? In this case the 24 bit delay of the monitor would be less than the 24 bit delay desired by the host. To avoid this situation the monitor maintains 3 extra bits to compensate for the faster bits. The 3 extra bits suffice even if bits are 10 % faster. This compensation is called Phase Jitter Compensation.

Handling multiple priority frames

Each node or packet has a priority level. We don't concern ourselves with how this priority is decided. The first 3 bits of the Access Control byte in the token are for priority and the last 3 are for reservation.

P	P	P	T	M	R	R	R

Initially the reservation bits are set to 000. When a node wants to transmit a priority n frame, it must wait until it can capture a token whose priority is less than or equal to n. Furthermore, when a data frame goes by, a station can try to reserve the next token by writing the priority of the frame it wants to send into the frame's Reservation bits. However, if a higher priority has already been reserved there, the station cannot make a reservation. When the current frame is finished, the next token is generated at the priority that has been reserved.

A slight problem with the above reservation procedure is that the reservation priority keeps on increasing. To solve this problem, the station raising the priority remembers the reservation priority that it replaces and when it is done it reduces the priority to the previous priority.

Note that in a token ring, low priority frames may starve.

Ring Maintenance

Each token ring has a monitor that oversees the ring. Among the monitor's responsibilities are seeing that the token is not lost, taking action when the ring breaks, cleaning the ring when garbled frames appear and watching out for orphan frames. An orphan frame occurs when a station transmits a short frame in it's entirety onto a long ring and then crashes or is powered down before the frame can be removed. If nothing is done, the frame circulates indefinitely.

- **Detection of orphan frames:** The monitor detects orphan frames by setting the monitor bit in the Access Control byte whenever it passes through. If an incoming frame has this bit set, something is wrong since the same frame has passed the monitor twice. Evidently it was not removed by the source, so the monitor drains it.
- **Lost Tokens:** The monitor has a timer that is set to the longest possible tokenless interval : when each node transmits for the full token holding time. If this timer goes off, the monitor drains the ring and issues a fresh token.
- **Garbled frames:** The monitor can detect such frames by their invalid format or checksum, drain the ring and issue a fresh token.

The token ring control frames for maintenance are:

Control field	*Name*	*Meaning*
00000000	Duplicate address test	Test if two stations have the same address
00000010	Beacon	Used to locate breaks in the ring
00000011	Claim token	Attempt to become monitor
00000100	Purge	Reinitialize the ring
00000101	Active monitor present	Issued periodically by the monitor
00000110	Standby monitor present	Announces the presence of potential monitors

The monitor periodically issues a message "Active Monitor Present" informing all nodes of its presence. When this message is not received for a specific time interval, the nodes detect a monitor failure. Each node that believes it can function as a monitor broadcasts a "Standby Monitor Present" message at regular intervals, indicating that it is ready to take on the monitor's job. Any node that detects failure of a monitor issues a "Claim" token. There are 3 possible outcomes :

1. If the issuing node gets back its own claim token, then it becomes the monitor.
2. If a packet different from a claim token is received, apparently a wrong guess of monitor failure was made. In this case on receipt of our own claim token, we discard it. Note that our claim token may have been removed by some other node which has detected this error.
3. If some other node has also issued a claim token, then the node with the larger address becomes the monitor.

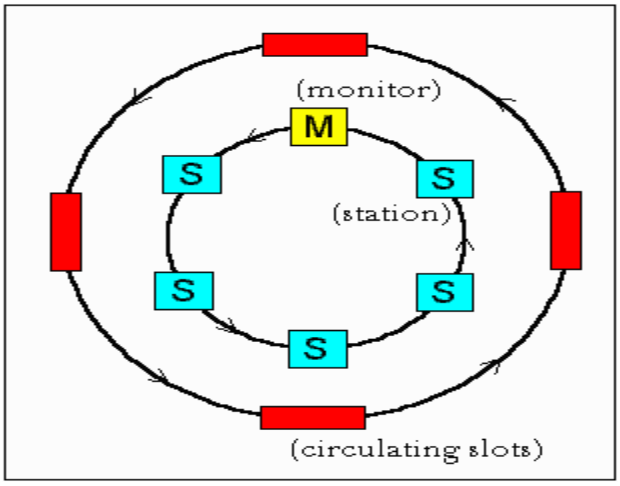
In order to resolve errors of duplicate addresses, whenever a node comes up it sends a **"Duplicate Address Detection"** message (with the destination = source) across the network. If the address recognize bit has been set on receipt of the message, the issuing node realizes a duplicate address and goes to standby mode. A node informs other nodes of removal of a packet from the ring through a **"Purge"** message. One maintenance function that the monitor cannot handle is locating breaks in the ring. If there is no activity detected in the ring (e.g. Failure of monitor to issue the **Active Monitor Present** token...) , the usual procedures of sending a claim token are followed. If the claim token itself is not received besides packets of any other kind, the node then sends **"Beacons"** at regular intervals until a message is received indicating that the broken ring has been repaired.

Other Ring Networks

The problem with the token ring system is that large rings cause large delays. It must be made possible for multiple packets to be in the ring simultaneously. The following ring networks resolve this problem to some extent:-

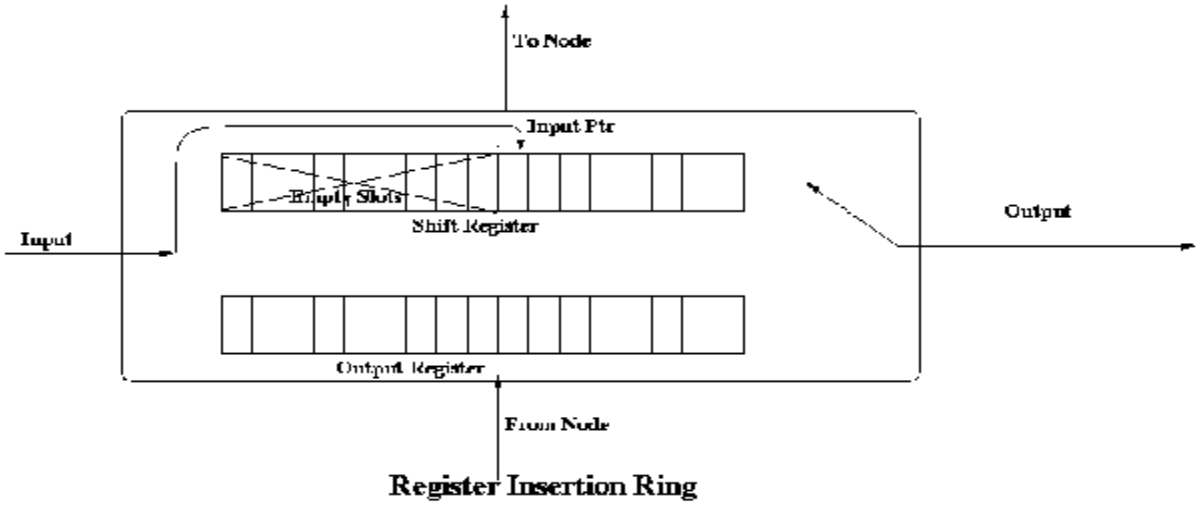
Slotted Ring:

In this system, the ring is slotted into a number of fixed size frames which are continuously moving around the ring. This makes it necessary that there be enough number of nodes (large ring size) to ensure that all the bits can stay on the ring at the same time. The frame header contains information as to whether the slots are empty or full. The usual disadvantages of overhead/wastage associated with fixed size frames are present.



Register Insertion Rings:

This is an improvement over slotted ring architecture. The network interface consists of two registers: a shift register and an output buffer. At startup, the input pointer points to the rightmost bit position in the input shift register .When a bit arrives it is in the rightmost empty position (the one indicated by the input pointer). After the node has detected that the frame is not addressed to it, the bits are transmitted one at time (by shifting). As new bits come in, they are inserted at the position indicated by the pointer and then the contents are shifted. Thus the pointer is not moved. Once the shift register has pushed out the last bit of a frame, it checks to see if it has an output frame waiting. In case yes, then it checks that if the number of empty slots in the shift register is at least equal to the number of bits in the output frame. After this the output connection is switched to this second register and after the register has emptied its contents, the output line is switched back to the shift register. Thus, no single node can hog the bandwidth. In a loaded system, a node can transmit a k-bit frame only if it has saved up a k-bits of inter frame gaps.



Register Insertion Ring

Two major disadvantages of this topology are complicated hardware and difficulty in the detection of start/end of packets.

Contention Ring

The token ring has primarily two problems:

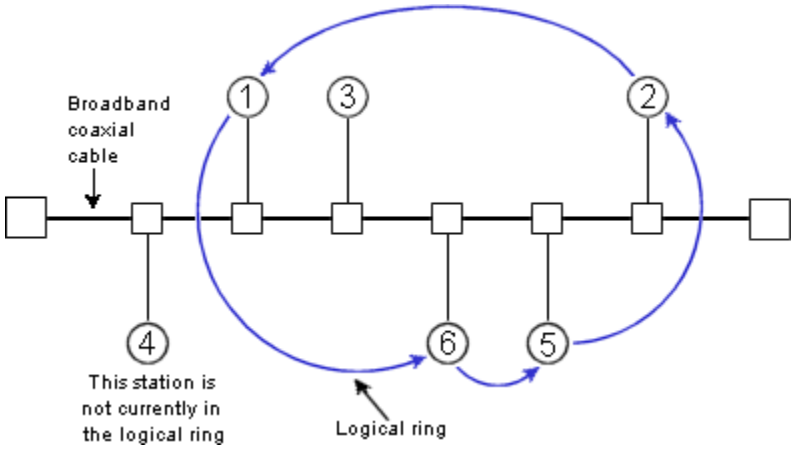
- On light loads, huge overhead is incurred for token passing.
- Nodes with low priority data may starve if there is always a node with high priority data.

A contention ring attempts to address these problems. In a contention ring, if there is no communication in the ring for a while, a sender node will send its data immediately, followed by a token. If the token comes back to the sender without any data packet in between, the sender removes it from the ring. However under heavy load the behavior is that of a normal token ring.

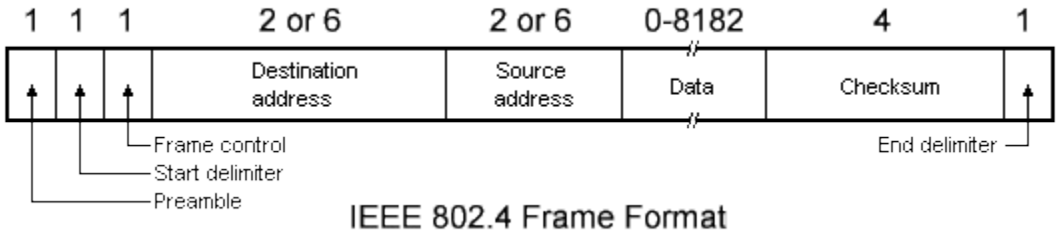
In case a collision, each of the sending nodes will remove the others' data packet from the ring, back off for a random period of time and then resend their data.

IEEE 802.4: Token Bus Network

In this system, the nodes are physically connected as a bus, but logically form a ring with tokens passed around to determine the turns for sending. It has the robustness of the 802.3 broadcast cable and the known worst case behavior of a ring. The structure of a token bus network is as follows:



Frame Structure



IEEE 802.4 Frame Format

A 802.4 frame has the following fields:

- Preamble: The Preamble is used to synchronize the receiver's clock.
- Starting Delimiter (SD) and End Delimiter (ED): The Starting Delimiter and Ending Delimiter fields are used to mark frame boundaries. Both of them contain analog encoding of symbols other than 1 or 0 so that they cannot occur accidentally in the user data. Hence no length field is needed.
- Frame Control (FC): This field is used to distinguish data frames from control frames. For data frames, it carries the frame's priority as well as a bit which the destination can set as an acknowledgement. For control frames, the Frame Control field is used to specify

the frame type. The allowed types include token passing and various ring maintenance frames.

- Destination and Source Address: The Destination and Source address fields may be 2 bytes (for a local address) or 6 bytes (for a global address).
- Data: The Data field carries the actual data and it may be 8182 bytes when 2 byte addresses are used and 8174 bytes for 6 byte addresses.
- Checksum: A 4-byte checksum calculated for the data. Used in error detection.

Ring Maintenance:

Mechanism:

When the first node on the token bus comes up, it sends a **Claim_token** packet to initialize the ring. If more than one station send this packet at the same time, there is a collision. Collision is resolved by a contention mechanism, in which the contending nodes send random data for 1, 2, 3 and 4 units of time depending on the first two bits of their address. The node sending data for the longest time wins. If two nodes have the same first two bits in their addresses, then contention is done again based on the next two bits of their address and so on.

After the ring is set up, new nodes which are powered up may wish to join the ring. For this a node sends **Solicit_successor_1** packets from time to time, inviting bids from new nodes to join the ring. This packet contains the address of the current node and its current successor, and asks for nodes in between these two addresses to reply. If more than one nodes respond, there will be collision. The node then sends a **Resolve_contention** packet, and the contention is resolved using a similar mechanism as described previously. Thus at a time only one node gets to enter the ring. The last node in the ring will send a **Solicit_successor_2** packet containing the addresses of it and its successor. This packet asks nodes not having addresses in between these two addresses to respond.

A question arises that how frequently should a node send a Solicit_successor packet? If it is sent too frequently, then overhead will be too high. Again if it is sent too rarely, nodes will have to wait for a long time before joining the ring. If the channel is not busy, a node will send a Solicit_successor packet after a fixed number of token rotations. This number can be configured by the network administrator. However if there is heavy traffic in the network, then a node would defer the sending of bids for successors to join in.

There may be problems in the logical ring due to sudden failure of a node. What happens when a node goes down along with the token? After passing the token, a node, say node A, listens to the channel to see if its successor either transmits the token or passes a frame. If neither happens, it resends a token. Still if nothing happens, A sends a **Who_follows** packet, containing the address of the down node. The successor of the down node, say node C, will now respond with a **Set_successor** packet, containing its own address. This causes A to set its successor node to C, and the logical ring is restored. However, if two successive nodes go down suddenly, the ring will be dead and will have to be built afresh, starting from a **Claim_token** packet.

When a node wants to shutdown normally, it sends a **Set_successor** packet to its predecessor, naming its own successor. The ring then continues unbroken, and the node goes out of the ring.

The various control frames used for ring maintenance are shown below:

Frame Control Field	**Name**	**Meaning**
00000000	Claim_token	Claim token during ring maintenance
00000001	Solicit_successor_1	Allow stations to enter the ring
00000010	Solicit_successor_2	Allow stations to enter the ring
00000011	Who_follows	Recover from lost token
00000100	Resolve_contention	Used when multiple stations want to enter
00001000	Token	Pass the token
00001100	Set_successor	Allow the stations leave the ring

Priority Scheme:

Token bus supports four distinct priority levels: 0, 2, 4 and 6.

0 is the lowest priority level and 6 the highest. The following times are defined by the token bus:

- THT: Token Holding Time. A node holding the token can send priority 6 data for a maximum of this amount of time.
- TRT_4: Token Rotation Time for class 4 data. This is the maximum time a token can take to circulate and still allow transmission of class 4 data.
- TRT_2 and TRT_0: Similar to TRT_4.

When a station receives data, it proceeds in the following manner:

- It transmits priority 6 data for at most THT time, or as long as it has data.
- Now if the time for the token to come back to it is less than TRT_4, it will transmit priority 4 data, and for the amount of time allowed by TRT_4. Therefore the maximum time for which it can send priority 4 data is= Actual TRT - THT - TRT_4
- Similarly for priority 2 and priority 0 data.

This mechanism ensures that priority 6 data is always sent, making the system suitable for real time data transmission. In fact this was one of the primary aims in the design of token bus.

Data Link Layer

What is DLL (Data Link Layer)

The Data Link Layer is the second layer in the OSI model, above the Physical Layer, which ensures that the error free data is transferred between the adjacent nodes in the network. It breaks the datagrams passed down by above layers and convert them into frames ready for transfer. This is called Framing. It provides two main functionalities

- Reliable data transfer service between two peer network layers
- Flow Control mechanism which regulates the flow of frames such that data congestion is not there at slow receivers due to fast senders.

What is Framing?

Since the physical layer merely accepts and transmits a stream of bits without any regard to meaning or structure, it is upto the data link layer to create and recognize frame boundaries. This can be accomplished by attaching special bit patterns to the beginning and end of the frame. If these bit patterns can accidentally occur in data, special care must be taken to make sure these patterns are not incorrectly interpreted as frame delimiters. The four framing methods that are widely used are

- Character count
- Starting and ending characters, with character stuffing
- Starting and ending flags, with bit stuffing
- Physical layer coding violations

Character Count

This method uses a field in the header to specify the number of characters in the frame. When the 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 disadvantage is that if the count is garbled by a transmission error, the destination will lose synchronization and will be unable to locate the start of the next frame. So, this method is rarely used.

Character stuffing

In the second method, each frame starts with the ASCII character sequence DLE STX and ends with the sequence DLE ETX.(where DLE is Data Link Escape, STX is Start of TeXt and ETX is End of TeXt.) This method overcomes the drawbacks of the character count method. If the destination ever loses synchronization, it only has to look for DLE STX and DLE ETX characters. If however, binary data is being transmitted then there exists a possibility of the characters DLE STX and DLE ETX occurring in the data. Since this can interfere with the framing, a technique called character stuffing is used. The sender's data link layer inserts an ASCII DLE character just before the DLE character in the data. The receiver's data link layer removes this DLE before this data is given to the network layer. However character stuffing is

closely associated with 8-bit characters and this is a major hurdle in transmitting arbitrary sized characters.

Bit stuffing

The third method allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. At the start and end of each frame is a flag byte consisting of the special bit pattern 01111110 . Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a zero bit into the outgoing bit stream. This technique is called bit stuffing. When the receiver sees five consecutive 1s in the incoming data stream, followed by a zero bit, it automatically destuffs the 0 bit. The boundary between two frames can be determined by locating the flag pattern.

Physical layer coding violations

The final framing method is physical layer coding violations and is applicable to networks in which the encoding on the physical medium contains some redundancy. In such cases normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair. The combinations of low-low and high-high which are not used for data may be used for marking frame boundaries.

Error Control

The bit stream transmitted by the physical layer is not guaranteed to be error free. The data link layer is responsible for error detection and correction. The most common error control method is to compute and append some form of a checksum to each outgoing frame at the sender's data link layer and to recompute the checksum and verify it with the received checksum at the receiver's side. If both of them match, then the frame is correctly received; else it is erroneous. The checksums may be of two types:
Error detecting : Receiver can only detect the error in the frame and inform the sender about it.
Error detecting and correcting : The receiver can not only detect the error but also correct it.
Examples of Error Detecting methods:

- **Parity bit:**
  Simple example of error detection technique is parity bit. The parity bit is chosen that the number of 1 bits in the code word is either even(for even parity) or odd (for odd parity). For example when 10110101 is transmitted then for even parity an 1 will be appended to the data and for odd parity a 0 will be appended. This scheme can detect only single bits. So if two or more bits are changed then that can not be detected.
- **Longitudinal Redundancy Checksum:**
  Longitudinal Redundancy Checksum is an error detecting scheme which overcomes the problem of two erroneous bits. In this conceptof parity bit is used but with slightly more intelligence. With each byte we send one parity bit then send one additional byte which have the parity corresponding to the each bit position of the sent bytes. So the parity bit is set in both horizontal and vertical direction. If one bit get flipped we can tell which row and column have error then we find the intersection of the two and determine the erroneous bit. If 2 bits are in error and they are in the different column and row then they

can be detected. If the error are in the same column then the row will differentiate and vice versa. Parity can detect the only odd number of errors. If they are even and distributed in a fashion that in all direction then LRC may not be able to find the error.

- **Cyclic Redundancy Checksum (CRC):**
  We have an n-bit message. The sender adds a k-bit Frame Check Sequence (FCS) to this message before sending. The resulting (n+k) bit message is divisible by some (k+1) bit number. The receiver divides the message ((n+k)-bit) by the same (k+1)-bit number and if there is no remainder, assumes that there was no error. How do we choose this number? For example, if k=12 then 1000000000000 (13-bit number) can be chosen, but this is a pretty crappy choice. Because it will result in a zero remainder for all (n+k) bit messages with the last 12 bits zero. Thus, any bits flipping beyond the last 12 go undetected. If k=12, and we take 1110001000110 as the 13-bit number (incidentally, in decimal representation this turns out to be 7238). This will be unable to detect errors only if the corrupt message and original message have a difference of a multiple of 7238. The probability of this is low, much lower than the probability that anything beyond the last 12-bits flips. In practice, this number is chosen after analyzing common network transmission errors and then selecting a number which is likely to detect these common errors.

How to detect source errors?

In order ensure that the frames are delivered correctly, the receiver should inform the sender about incoming frames using positive or negative acknowledgements. On the sender's side the receipt of a positive acknowledgement implies that the frame has arrived at the destination safely while the receipt of a negative acknowledgement means that an error has occurred in the frame and it needs to be retransmitted. However, this scheme is too simplistic because if a noise burst causes the frame to vanish completely, the receiver will not respond at all and the sender would hang forever waiting for an acknowledgement. To overcome this drawback, timers are introduced into the data link layer. When the sender transmits a frame it also simultaneously starts a timer. The timer is set to go off after a interval long enough for the frame to reach the destination, be processed there, and have the acknowledgement propogate back to the sender. If the frame is received correctly the positive acknowledgment arrives before the timer runs out and so the timer is canceled. If however either the frame or the acknowledgement is lost the timer will go off and the sender may retransmit the frame. Since multiple transmission of frames can cause the receiver to accept the same frame and pass it to the network layer more than once, sequence numbers are generally assigned to the outgoing frames.
The types of acknowledgements that are sent can be classified as follows:

- Cumulative acknowledgements: A single acknowledgement informing the sender that all the frames upto a certain number has been received.
- Selective acknowledgements: Acknowledgement for a particular frame.

They may be also classified as:

- Individual acknowledgements: Individual acknowledgement for each frame.

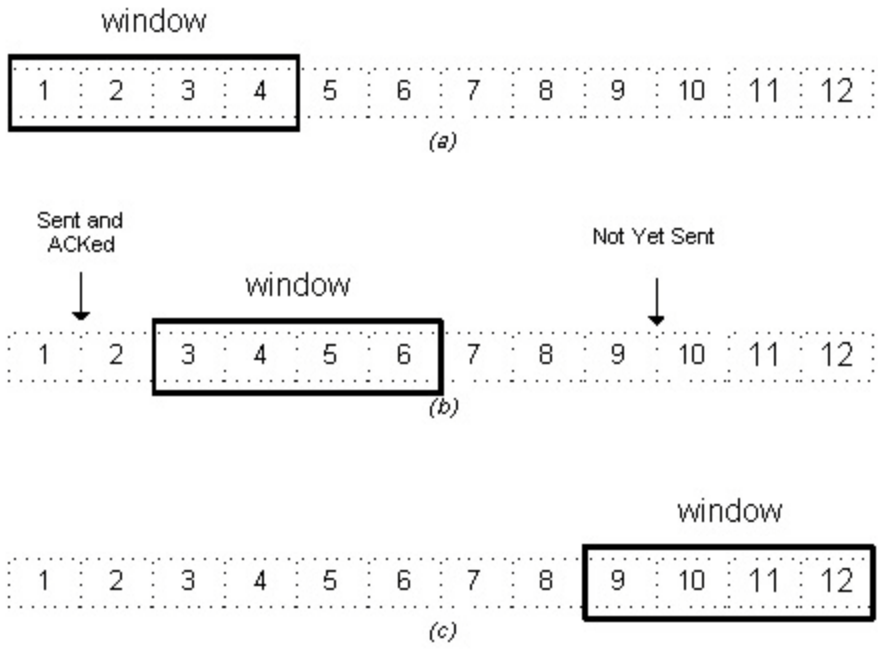
- Group acknowledgements: A bit-map that specifies the acknowledgements of a range of frame numbers.

Flow Control

Consider a situation in which the sender transmits frames faster than the receiver can accept them. If the sender keeps pumping out frames at high rate, at some point the receiver will be completely swamped and will start losing some frames. This problem may be solved by introducing flow control. Most flow control protocols contain a feedback mechanism to inform the sender when it should transmit the next frame.

Mechanisms for Flow Control:

- **Stop and Wait Protocol:** This is the simplest file control protocol in which the sender transmits a frame and then waits for an acknowledgement, either positive or negative, from the receiver before proceeding. If a positive acknowledgement is received, the sender transmits the next packet; else it retransmits the same frame. However, this protocol has one major flaw in it. If a packet or an acknowledgement is completely destroyed in transit due to a noise burst, a deadlock will occur because the sender cannot proceed until it receives an acknowledgement. This problem may be solved using timers on the sender's side. When the frame is transmitted, the timer is set. If there is no response from the receiver within a certain time interval, the timer goes off and the frame may be retransmitted.
- **Sliding Window Protocols:** In spite of the use of timers, the stop and wait protocol still suffers from a few drawbacks. Firstly, if the receiver had the capacity to accept more than one frame, its resources are being underutilized. Secondly, if the receiver was busy and did not wish to receive any more packets, it may delay the acknowledgement. However, the timer on the sender's side may go off and cause an unnecessary retransmission. These drawbacks are overcome by the sliding window protocols.
  In sliding window protocols the sender's data link layer maintains a 'sending window' which consists of a set of sequence numbers corresponding to the frames it is permitted to send. Similarly, the receiver maintains a 'receiving window' corresponding to the set of frames it is permitted to accept. The window size is dependent on the retransmission policy and it may differ in values for the receiver's and the sender's window. The sequence numbers within the sender's window represent the frames sent but as yet not acknowledged. Whenever a new packet arrives from the network layer, the upper edge of the window is advanced by one. When an acknowledgement arrives from the receiver the lower edge is advanced by one. The receiver's window corresponds to the frames that the receiver's data link layer may accept. When a frame with sequence number equal to the lower edge of the window is received, it is passed to the network layer, an acknowledgement is generated and the window is rotated by one. If however, a frame falling outside the window is received, the receiver's data link layer has two options. It may either discard this frame and all subsequent frames until the desired frame is received or it may accept these frames or buffer them until the appropriate frame is received and then pass the frames to the network layer in sequence.



In this simple example, there is a 4-byte sliding window. Moving from left to right, the window "slides" as bytes in the stream are sent and acknowledged.
Most sliding window protocols also employ ARQ (Automatic Repeat reQuest) mechanism. In ARQ, the sender waits for a positive acknowledgement before proceeding to the next frame. If no acknowledgement is received within a certain time interval it retransmits the frame. ARQ is of two types:

1. **Go Back 'n':** If a frame is lost or received in error, the receiver may simply discard all subsequent frames, sending no acknowledgments for the discarded frames. In this case the receive window is of size 1. Since no acknowledgements are being received the sender's window will fill up, the sender will eventually time out and retransmit all the unacknowledged frames in order starting from the damaged or lost frame. The maximum window size for this protocol can be obtained as follows. Assume that the window size of the sender is n. So the window will initially contain the frames with sequence numbers from 0 to (w-1). Consider that the sender transmits all these frames and the receiver's data link layer receives all of them correctly. However, the sender's data link layer does not receive any acknowledgements as all of them are lost. So the sender will retransmit all the frames after its timer goes off. However the receiver window has already advanced to w. Hence to avoid overlap, the sum of the two windows should be less than the sequence number space.

w-1 + 1 < Sequence Number Space
i.e., w < Sequence Number Space
Maximum Window Size = Sequence Number Space - 1

2. **Selective Repeat:** In this protocol rather than discard all the subsequent frames following a damaged or lost frame, the receiver's data link layer simply stores them in buffers. When the sender does not receive an acknowledgement for the first frame it's timer goes off after a certain time interval and it retransmits only the lost frame. Assuming error - free transmission this time, the sender's data link layer will have a sequence of a many correct frames which it can hand over to the network layer. Thus there is less overhead in retransmission than in the case of Go Back n protocol.
   In case of selective repeat protocol the window size may be calculated as follows. Assume that the size of both the sender's and the receiver's window is w. So initially both of them contain the values 0 to (w-1). Consider that sender's data link layer transmits all the w frames; the receiver's data link layer receives them correctly and sends acknowledgements for each of them. However, all the acknowledgements are lost and the sender does not advance it's window. The receiver window at this point contains the values w to (2w-1). To avoid overlap when the sender's data link layer retransmits, we must have the sum of these two windows less than sequence number space. Hence, we get the condition

Maximum Window Size = Sequence Number Space / 2