Computer Networks (CS425)

Instructor: Dr. Dheeraj Sanghi

Prev| Next| Index

Token Ring Network (Contd...)

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.

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.

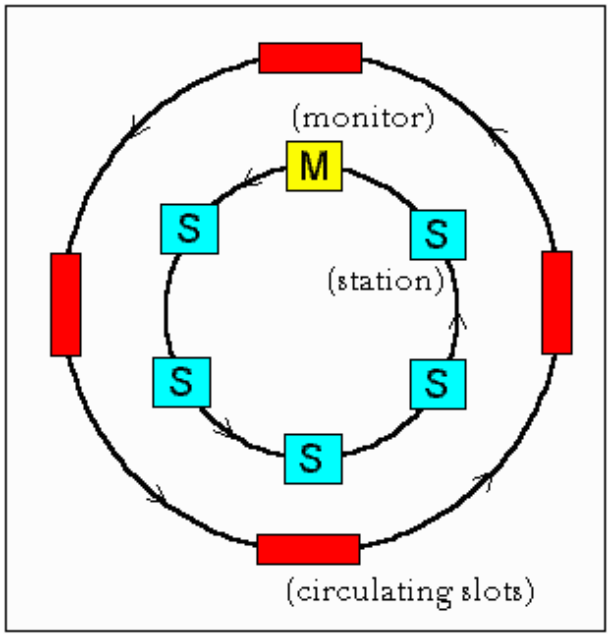
In order to resolve errors of duplicate addresses, whenever a node comes up it sends a **"Duplicate Address Detection"** message (with the destination = source) across the network. If the address recognize bit has been set on receipt of the message, the issuing node realizes a duplicate address and goes 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 :-

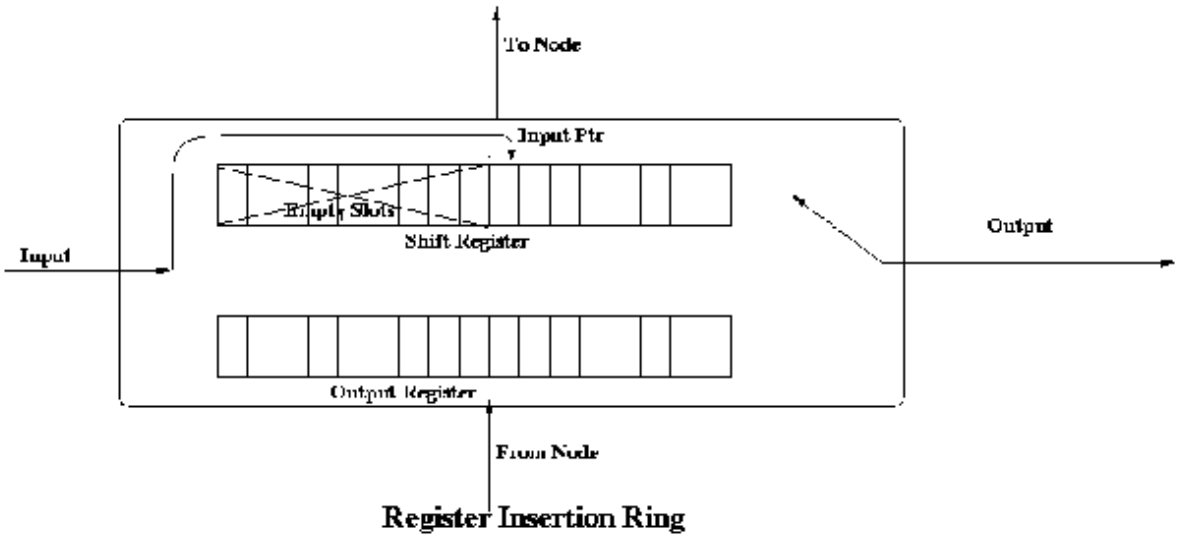
Slotted Ring :

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.



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

Contention Ring

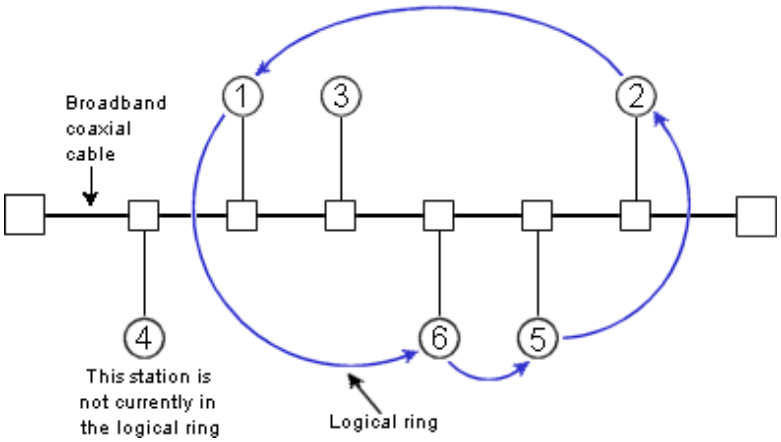
The token ring has primarily two problems:

- 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.

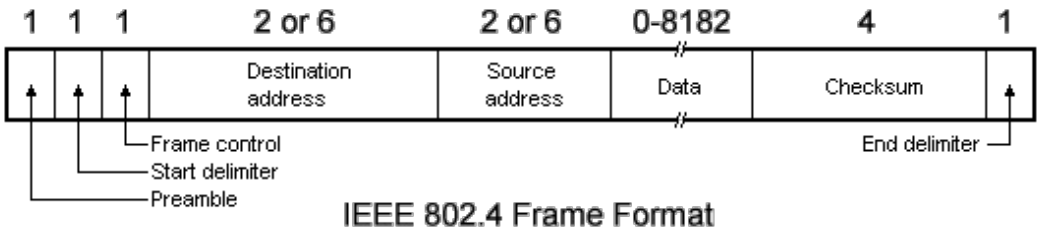
A contention ring attempts to address these problems. In a contention ring, if there is no communication in the ring for a while, a sender node will send its data immediately, followed by a token. If the token comes back to the sender without any data packet in between, the sender removes it from the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

Image References:

- https://www.cis.strath.ac.uk/teaching/ug/classes/52.354/Images/SlottedRing.gif
- http://www.lkn.ei.tum.de/lehre/mmprog/mac/protocols/token-bus/pics/bus1.gif
- http://www.acerimmeronline.com/networks/images/token_bus_frame.gif

---

back to top
Prev| Next | Index