The Token Ring network was originally developed by IBM in the 1970s.

IEEE 802.5 was modeled after IBM Token Ring, and it continues to shadow IBM’s Token Ring development.

The IEEE 802.5 specification is almost identical to and completely compatible with IBM’s Token Ring network.

So, the term Token Ring generally is used to refer to both IBM’s Token Ring network and IEEE 802.5 networks.

Classical Token Rings operate at 4 Mbps and 16 Mbps.

The original 4 Mbps version ran on STP cable, support for UTP telephone wire was added later.

A 100 Mbps version called **High Speed Token Ring (HSTR)** was debut in 1998, it useful to customers who wish to improve the performance of their existing Token Ring networks, however, the interfaces are expensive.
### IEEE 802.5 Specification

<table>
<thead>
<tr>
<th>Data rate (Mbps)</th>
<th>4, 16, 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>UTP / STP / Fiber</td>
</tr>
<tr>
<td>Signaling Technique</td>
<td>Differential Manchester</td>
</tr>
<tr>
<td>Maximum Frame Size (Bytes)</td>
<td>4550, 18200, 18200</td>
</tr>
<tr>
<td>Max Repeater</td>
<td>250</td>
</tr>
</tbody>
</table>

### Basic Ring Operation

- A special protocol data unit called a *token* circulates from station to station around the ring.
- A token is a simple placeholder frame that is passed from station to station around the ring.
- The mechanism that coordinates this rotation is called *token passing*.
- A station may send data only when it has possession of the token.
- On completion, it forwards a token to the next station.
Token Passing
MAC Protocol

- Small frame (token) circulates when all stations are idle
- Station waits for token
- Seizes the token by changing one bit in token to make it SOF (start-of-frame) for data frame
- Append rest of data frame
- Frame makes round trip, being regenerated by each station

- The intended recipient recognizes this frame has its own address, copies the message, checks for errors and changes four bits in the last byte of the frame to indicate address recognized and frame copied
- The full packet then continues around the ring until return to the station that sent it
- The sender receives the frame and recognizes itself in the source address field, then examines the address-recognized bits, if they are set, it knows the frame was received
- The sender then discards the used data frame and releases the token back to the ring
- Under light loads, some inefficiency
- Under heavy loads, round robin
## Token Ring MAC Frame Format

### (a) General Frame Format

<table>
<thead>
<tr>
<th>Octets</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>6</th>
<th>6</th>
<th>DA</th>
<th>SA</th>
<th>Data units</th>
<th>4</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SD</td>
<td>AC</td>
<td>FC</td>
<td>DA</td>
<td>SA</td>
<td>FCS</td>
<td>ED</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SD**: starting delimiter
- **DA**: destination address
- **AC**: access control
- **SA**: source address
- **FC**: frame control
- **FCS**: frame check sequence
- **ED**: ending delimiter
- **FS**: frame status

### (b) Token Frame Format

<table>
<thead>
<tr>
<th>P</th>
<th>P</th>
<th>P</th>
<th>T</th>
<th>M</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPP: priority bits</td>
<td>M: monitor bit</td>
<td>T: token bit</td>
<td>RRR: reservation bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### (c) Access Control Field

<table>
<thead>
<tr>
<th>P</th>
<th>F</th>
<th>Z</th>
<th>Z</th>
<th>Z</th>
<th>Z</th>
<th>Z</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT: frame-type bits</td>
<td>ZZZZZZ: control bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### (d) Frame Control Field

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>I</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>J, K: non-data bits</td>
<td>I: intermediate frame bit</td>
<td>E: error-detected bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### (e) Ending Delimiter Field

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>r</th>
<th>r</th>
<th>A</th>
<th>C</th>
<th>r</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: Address recognizable bit</td>
<td>C: Frame copied bit</td>
<td>( r ): reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### (f) Frame Status Field
**Token Ring MAC Frame Format**

**General Frame Format**

<table>
<thead>
<tr>
<th>Bytes</th>
<th>1</th>
<th>1</th>
<th>2 or 6</th>
<th>2 or 6</th>
<th>&gt;0</th>
<th>4</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SD</td>
<td>AC</td>
<td>FC</td>
<td>DA</td>
<td>SA</td>
<td>Info.</td>
<td>FCS</td>
<td>ED</td>
</tr>
</tbody>
</table>

- **Starting delimiter (SD):** Indicates start of frame
- **Access Control (AC):** Has the format PPPTMRRR, where PPP and RRR are 3-bit priority and reservation variables, and M is monitor bit, T indicates whether this is a token or data frame. In the case of a token frame, the only remaining field is ED
- **Frame control (FC)**
- **Destination address (DA)**
- **Source address (SA)**
- **Data unit (Info.):** Contains LLC data unit
- **Frame check sequence (FCS)**
- **End delimiter (ED):** Contains the error detected bit (E), which is set if any repeater detects an error, and intermediate bit (I), which is used to indicate that this is a frame other than the final one of a multiple frame transmission
- **Frame status (FS):** Contains the address recognized (A) and frame copied (C) bits
One-Bit Delay / Ring Delay

- Check Address
- Copy
- Alter Bit

Minimum Ring Delay

- Ring Delay < 24 bits
- Ring Delay ≥ 24 bits
In this case, the priority and reservation bits are set to zero.

A station wish to transmit wait until a token goes by, as indicated (token free) by a token bit of 0 in the AC field.

The station seizes the token by setting the token bit to 1.

The SD and AC fields of the received token now function as the first two fields of the outgoing frame.

The station transmits one or more frames, continuing until either it supply of frames is exhausted or a token-holding timer expires.

When the AC field of last transmitted frame returns, the station sets the token bit to 0 and appends an ED field, resulting in the insertion of a new token on the ring.
Station in the receive mode listen to the ring
Each station can check passing frames for errors and set the E bit to 1 if an error is detected
If station detects its own MAC address, it sets the A bit to 1; it may also copy the frame, setting the C bit to 1
This allows the originating station to differentiate three results of a frame transmission:
- Destination station nonexistent or not active (A = 0, C = 0)
- Destination station exists but frame not copied (A = 1, C = 0)
- Frame received (A = 1, C = 1)
The 802.5 standard includes a specification for an optional priority mechanism.

Eight levels of priority are supported by providing two 3-bit fields in each data frame and token.

Two 3-bit fields are a priority field and a reservation field.

The priority parameters are defined as follows:

\[ P = \text{priority the data / token frame} \]
\[ R = \text{reservation the data / token frame} \]

- Seize the token if \( P_f \geq P \)
- Reserve for next token \((R \leftarrow P)\) if \( P_f > R \)

\( P_f \) : priority of the data frame waiting to be transmitted.
**Priority Parameters**

\[ P_r = \text{priority of the last received data / token frame} \]
\[ R_r = \text{reservation of the last received data / token frame} \]
\[ P_s = \text{priority of the send data / token frame} \]
\[ R_s = \text{reservation of the send data / token frame} \]

**Priority Scheme**

1. A station wishing to transmit must wait for a token with \( P_r \leq P_f \).
2. While waiting, a station may reserve a future token at its priority level \( P_f \).
3. If a data frame goes by, and if the reservation field is less than its priority \( R_r < P_f \), then the station may set the reservation field of the frame to its priority \( R_s \rightarrow P_f \).
4. If a token frame goes by, and if \( R_r < P_f \) AND \( P_f < P_s \), then the station sets the reservation field of the frame to its priority \( R_s \rightarrow P_f \).
5. When a station seizes a token, it sets the token bit to 1 to start a data frame, sets the reservation field of the data frame to 0, and leaves the priority field unchanged (the same as that of the incoming token frame).
6. Following transmission of one or more data frames, a station issues a new token with the priority and reservation fields set appropriately.
The effect of the steps 1 through 3 is to sort the competing claims and allow the waiting transmission of highest priority to seize the token as soon as possible. To avoid driving token to the highest used level and keeping it there, a station that raises the priority (issues a token that has a higher priority than the token that it received) has the responsibility of later lowering the priority to its previous level. A station that raises priority must remember both the old and the new priorities and downgrade the priority of the token at the appropriate time. To implement the downgrading mechanism, two stacks are maintained by each station, one for reservations and one for priorities:

- $S_x$ = stack used to store new values of token priority
- $S_r$ = stack used to store old values of token priority

Operation Modes:

Priority System-1

State Diagram of A Station with Priority

- **Seize the token** if $P_f \geq P_r$
- **Sending Frame**
- **Normal Mode**
- **Waiting** if data or token with $P_f < P_r$
- **Reserving** if $R_f < P_r$
- **Token Holder**
- **Return the Token**
  - Increase to $P_r$ if $P_f \geq P_r$
  - Otherwise unchanged
Normal Mode: A Token Arrive

A token arrive

Normal Station

- Reserve for next token
- \( P_f \leftarrow P_r \) unchanged
- \( R_s < P_f \)
- \( R_s \leftarrow R_f \) unchanged
- Release a Token

Normal Mode: A Frame Arrive

- Priority of the frame waiting to be transmitted no \( P_f = 0 \)
- A frame arrive
- \( P_f \leftarrow P_f \) unchanged
- \( R_s < P_f \)
- \( R_s \leftarrow R_s \) unchanged
- Forward the frame

Send a frame
- \( P_s \leftarrow P_f \) (unchanged)
- \( R_s \leftarrow 0 \) (reset)
Normal Mode

Do nothing if data frame not available
If data frame with priority $P_r$ available
• Try to seize the token with $P_r \leq P_f$
• Able to send a data frame if seize a token and become a Token Holder Station
• Try to reserve for next token if it cannot seize the token by $R_s \leftarrow P_f$ if $R_r \leq P_f$

Note:
• A normal station does not change the priority $P_s$ of the incoming data / token frame

Operation Modes: Priority System-2

State Diagram of A Station with Priority

- Waiting if data or token with $P_f < P_r$
- Reserving if $R_r < P_f$
- Increase to $P_r$ if $P_f \geq P_r$
- Issue
  • There must be someone responsible for decreasing (restoring) the priority to $P_r$
  • when no station requesting for token with $P_f$ any more
Operation Modes:
Priority System-3

State Diagram of A Station with Priority

- Waiting
- Reserving
- Seize the token
- Sending Frame
- Normal Mode
- Restore Priority
- Increase Priority
- Stacking Station
- The station which increases the priority of the token, is responsible for restoring the priority
- Monitor the priority of the incoming token

Token Holder : Priority Assignment

- Increase the priority become a stack station
  - How: Increase priority from $P_O$ to $P_H$
  - When: There are stations requesting (reserve) for $P_H$

- Decrease the priority become a normal station
  - How: Decrease (restore) priority from $P_H$ to $P_O$
  - When: No station requesting for token with priority $P_H$ anymore

How to know?
How to know?

- Any station which increase the priority of the token, is responsible for restoring the priority.
- **Answer** token returns with priority $P_H$.

Stacking Station Concept

- When increase priority station store:
  - $P_o$: to restore
  - $P_H$: to check whether to restore $P_o$ or not
- When token arrive check
  Token Priority = $P_H$
  - Yes: restore $P_o$
  - No: unchanged
State Diagram of A Station with Priority

- **Normal Mode**
  - Waiting
  - Reserving
  - Seize the token
  - Sending Frame

- **Stacking Station**
  - Waiting
  - Reserving
  - Seize the token
  - Unchanged
  - Increase Priority
  - Sending Frame

**Stacking Station Concept**

- When increase priority station store
  - $P_o(i)$: to restore
  - $P_H(i)$: to check whether to restore $P_o(i)$ or not

- When token arrive check
  - **Token Priority** = $P_H(i)$
  - Yes: restore $P_o(i)$
  - No: unchanged
A Token Holder

- Able to send data as long as data available and THT (Token Hold Time) not expired
- Alters the token priority after finishing sending
  - If $P_r < \text{MAX}(R_r, P_r)$ increase priority to the max. reservation, e.g., $P_s \leftarrow \text{MAX}(R_r, P_r)$
  - Else, unchanged priority ($P_s \leftarrow P_r$)
- Become a Stacking Station if PS is increased
- Become a normal station if PS unchanged

Action of A Token Holder
Stacking Station

When a TH raise the priority for a station it must remember
- the new token priority in stack $S_x$
- the old token priority in stack $S_r$

The SS is responsible for
- restore the old priority when it found that the station, for which it raised the token priority, had used the token

```
the token returns without priority change
```

The SS operations
- If the arriving token has $P_r = S_x$ restore the priority by setting $P_s = S_r$
- Become a normal mode station if restore all value (stack empty)

Normal Stacking Station:
A Token Arrived (1)

[Diagram showing decision tree for A Token Arrive, Frame available, $P_r = S_x$, $P_f \geq P_r$, Release a Token, Modify Stack, Normal Mode, Stacking Station, Stacking Token Holder, Send a frame $P_s \leftarrow P_r$ (unchanged), $R_s \leftarrow 0$ (reset)]
Normal Stacking Station:
A Token Arrived (2)

Stacking Token Holder:
A Frame Returned
Stacking Token Holder: A Frame Returned (2)

- Increase $P_s$
  - $P_s \leftarrow \text{MAX}(R_s, P_f)$
  - $R_s \leftarrow 0$

- Stack
  - If $P_s > S_x$, push $S_x$: $S_x \leftarrow P_s$
  - If $P_s = S_x$, pop $S_x$: $S_x \leftarrow P_f$

- Replace
  - Push $S_x$: $S_x \leftarrow P_f$
  - Release a Token

Priority Scheme

1. A is sending to B; D reserves at higher level
2. A generates higher priority token and remembers lower priority.
3. D uses higher priority token to send data to C.
4. D generates token at higher priority level
5. A sees the high priority token and captures it.
6. A generates token at the preempted lower priority level.
Example of Priority Scheme

- A is transmitting a data frame to B at priority 0. When the frame has completed a circuit of the ring and returns to A, A will issue a token. However, as the data frame passes D, D makes a reservation at priority 3 by setting the reservation field to 3.
- A issues a token with the priority field set to 3.
- If neither B nor C has data of priority 3 or greater to send, they cannot seize the token. The token circulates to D, which seizes the token and issues a data frame.
- After D’s data frame returns to D, D issues a new token at the same priority as the token that it received: priority 3.
- A sees a token at the priority level that it used to last issue a token. It therefore seizes the token even if it has no data to send.
- A issues a token at the previous priority level: priority 0.

Note

- After A has issued a priority 3 token.
- If any station with data of priority 3 or greater may seize the token.
- Suppose that C has priority 4 data to send, C will seize the token, transmit its data frame, and reissue a priority 3 token, which is then seized by D.
- By the time that a priority 3 token arrives at A, all intervening stations with data of priority 3 or greater to send will have had the opportunity.
- It is now appropriate, therefore, for A to downgrade the token.
Example 2

A token ring network with priority 0, 2, 4 and 8 (the highest)

- Station 1: Frame of priority 2
- Station 7: Frame of priority 2
- Station 15: Frame of priority 4
- Station 17: Frame of priority 4
Example 2: Round 3,4

Example 2: Round 5,6
### Example 2: Round 7,8

<table>
<thead>
<tr>
<th>Station 1</th>
<th>Station 7</th>
<th>Station 14</th>
<th>Station 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_f = 2$</td>
<td>$P_f = 2$</td>
<td>$P_f = 4$</td>
<td>$P_f = 4$</td>
</tr>
<tr>
<td>F/T(P,R)</td>
<td>F/T(P,R)</td>
<td>F/T(P,R)</td>
<td>F/T(P,R)</td>
</tr>
<tr>
<td>$T(0,0)$</td>
<td>$T(0,0)$</td>
<td>$T(0,0)$</td>
<td>$T(0,0)$</td>
</tr>
<tr>
<td>$P_r = 0$, $R_r = 0$</td>
<td>$P_r = 0$, $R_r = 0$</td>
<td>$P_r = 0$, $R_r = 0$</td>
<td>$P_r = 0$, $R_r = 0$</td>
</tr>
<tr>
<td>$P_l = 0$</td>
<td>$P_l = 0$</td>
<td>$P_l = 0$</td>
<td>$P_l = 0$</td>
</tr>
<tr>
<td>$P_r \neq S_x$</td>
<td>$P_r \neq S_x$</td>
<td>$P_r \neq S_x$</td>
<td>$P_r \neq S_x$</td>
</tr>
</tbody>
</table>

Cease Stacking