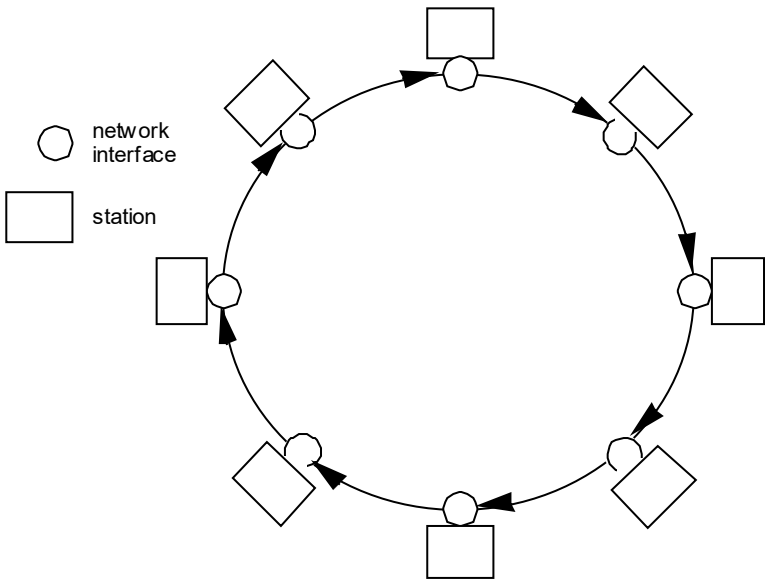
227-PTR TOKEN RING (TOR)

Protocol definition and specification

Network topology

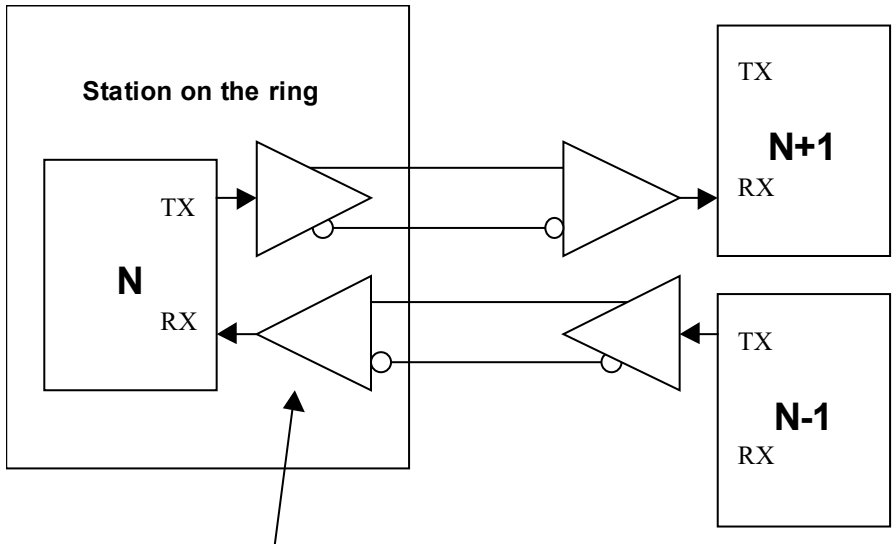
Our Token Ring network is inspired by (but not compatible with) the IEEE802.5 Token Ring. As in the original Token Ring, our network is based on simple point-to-point transmission lines between neighbors stations on the ring:



Network topology

A single token is traveling on the network. Only the station that owns it may send frames. Frames are removed from the ring by the sender station.

The line between stations uses an asynchronous transmission with 8 data bits, 1 stop bit, and no parity. As shown in the figure below, a single asynchronous port is required: the receiver part of the port is connected to the previous station on the ring, whereas the transmitter part is connected to the next station. To avoid ground loops, the transmission is differential. If both directions were connected to the same device, the transmission would be according to the RS422 standard (a differential version of RS232).

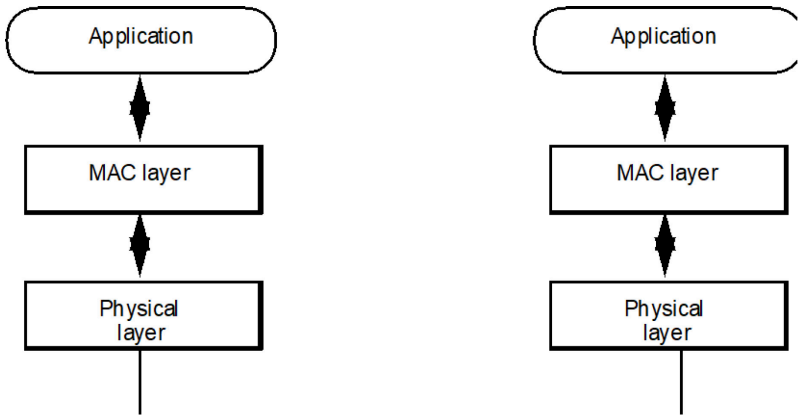


Asynchronous differential transmission on the token ring

Network architecture

The Token Ring network features only two layers:

- the physical layer as described above,
- the MAC (Medium Access Control Layer) freely inspired by the IEEE802.5 protocol.



Protocol layers in the Token Ring network

The MAC layer allows two applications within two stations to exchange information packets.

Physical layer services

The Physical Layer provides a single service with 2 service primitives to the MAC layer:

- PH_DATA_REQUEST(*frame*)

  Request to transmit the frame '*frame'* on the line segment towards the next station on the one-way ring.

- PH_DATA_INDICATION(*frame*)

  Indicate that the frame '*frame'* has been received from the previous station on the ring.

MAC layer services

The MAC layer provides a non-reliable connectionless data transfer service to the applications.

The available service primitives are listed below:

- MAC_DATA_REQUEST(*station*, *SAPI*, *data*)

  Request to transmit '*data'* towards a peer application defined by the station address *station* and the SAPI *SAPI*.

- MAC_DATA_INDICATION(*station*, *SAPI*, *data*)

  Indication that '*data'* have been received from a peer application defined by its station address (*station*) and its *SAPI*.

- MAC_LIST_REQUEST(*SAPI*)

  Request to get the list of stations ready to receive data on the SAPI *SAPI*.

- MAC_LIST_RESPONSE(*SAPI*, *station_i*, ...)

  Response to the request above: the station addresses listed have applications listening on the specified *SAPI*.

- MAC_START_REQUEST(*SAPI*)

  An application informs its MAC layer entity that it is ready to exchange data through the SAP defined by *SAPI*.

- MAC_END_REQUEST(*SAPI*)

  An application informs is MAC layer that is not more willing to exchange data through the SAP defined by SAPI.

- MAC_ERROR_INDICATION()

  Indicate when data could not be delivered as expected (timeout, no response, bad CRC, unknown SAPI, etc.).

- MAC_NEW_TOKEN()

  Allows an application to generate a new token. Required to initialize the network.

Addressing

Two types of addresses are used:

- **Station address:**

  This address (between 1 and 14) must be configured in each station. The address 15 is a broadcast address. Station addresses must be managed by the network manager.

- **Application address:**

  An application is bound to a SAP (*Service Access Point*) identified by a SAPI (*SAP Identifier*) (a 3-bit value between 0 and 7).
  To communicate, applications must use the same SAPI.

  The chat application uses SAPI number 1. The time application uses SAPI number 3.

MAC layer protocol

For simplicity reasons, the protocol does not deal with most of the error cases.

Frame type and frame format.

Two types of frames are used:

- **Data frames:**

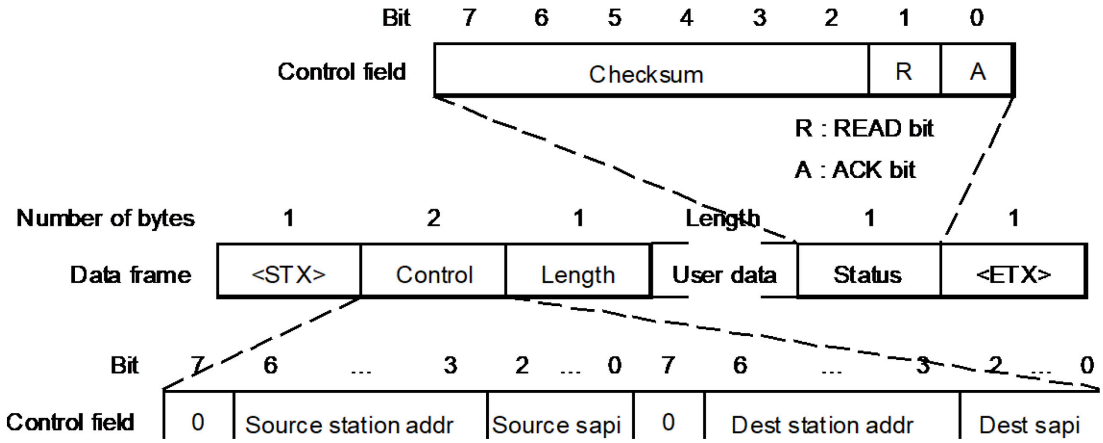
  A data frame contains application data (i.e. data generated by an application and destined to another application).

- **Token:**

  The token is used to:

  - to control the right "to talk" on the network,
  - to broadcast the list of applications (defined by the couple (*station*, *SAPI*)) ready to receive data.

The format of a data frame is defined in the following figure:



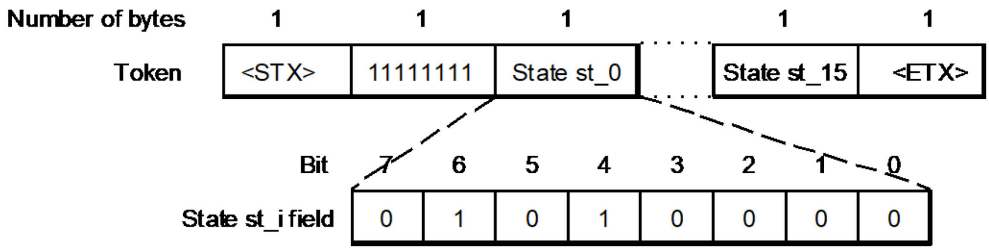
Structure of a data frame

The READ and ACK bits are always generated as logical "0" by the source station. The READ bit is set to 1 by the receiving station to indicate that it has been able to read the frame. The ACK bit is set to 1 by the receiving station if the error control procedure is successful (i.e. no transmission error).

When sending broadcast data, the READ and ACK bits are not used and must be both set to '1' by the source station. Broadcast data are always unconfirmed.

The Checksum field contains the 6 LSB (Least Significant Bits) of the sum of all octets (including Control, Length, and User Data fields).

The format of a token frame is defined in the following figure:



Bit j of field Stat st_ i = 1 -> An application in station i is willing to accept data on SAPI j

Bit j of field Stat st_ i = o -> An application in station i is NOT willing to accept data onSAPI i

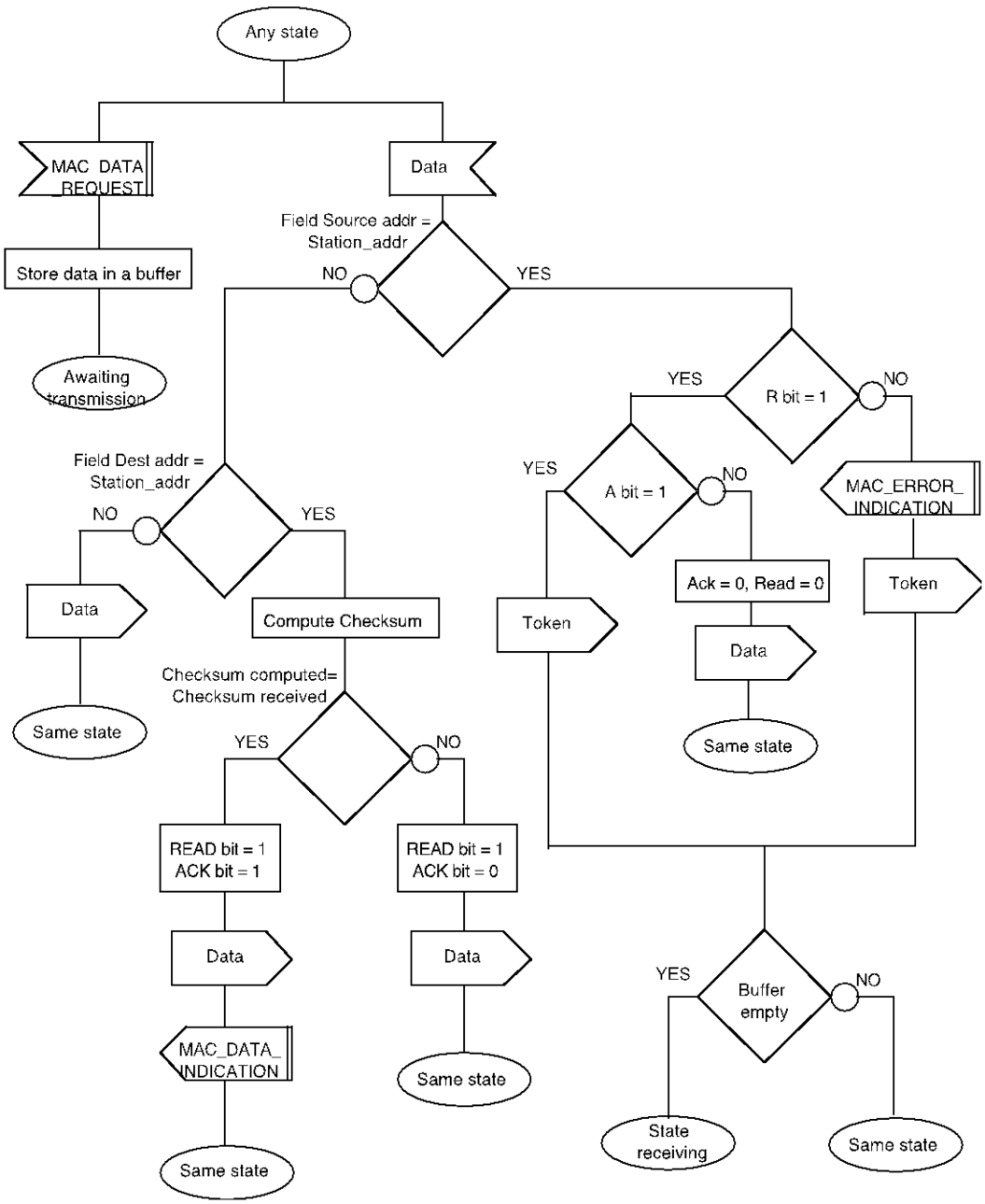
Token frame format

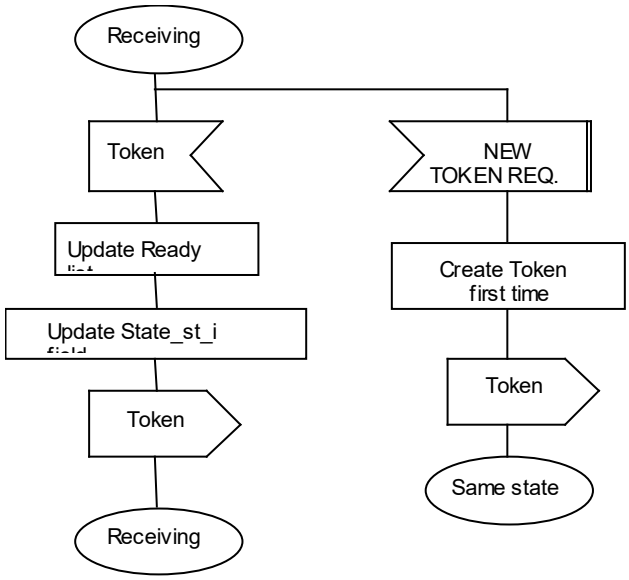
SDL (*Specification and Description Language*) diagrams

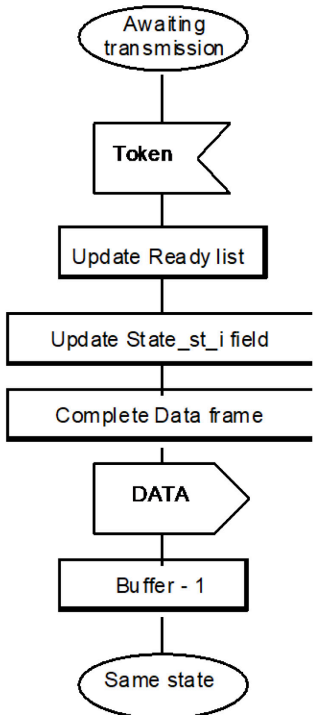
A station can be in one of the following states: receiving or awaiting transmission. A station can:

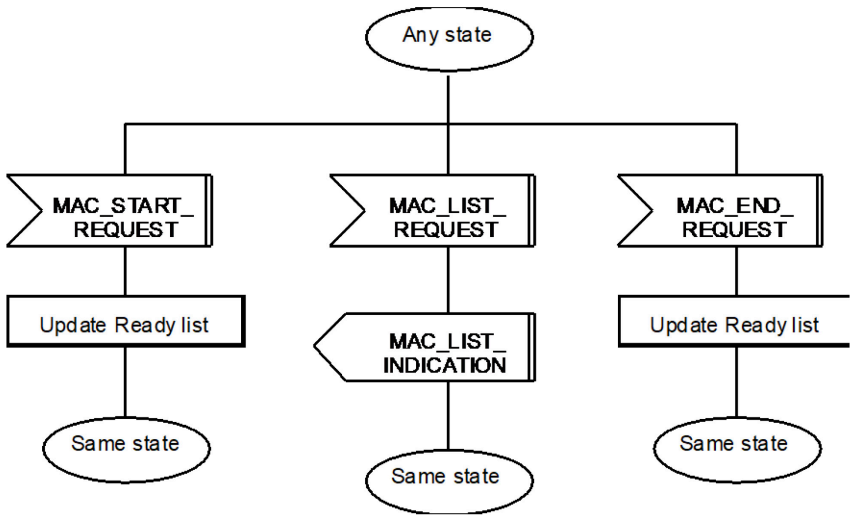
- Receiving data and displaying application information
- Enabling or disabling services
- Awaiting transmission and transmitting when allowed (only once message)

An early SDL




Receiving
Token
NEW
TOKEN REQ.
Update Ready
Create Token
first time
Update State_st_i
Token
Token
Same state
Receiving


Awaiting
transmission
Token
Update Ready list
Update State_st_i field
Complete Data frame
DATA
Buffer - 1
Same state



Those SDL diagrams must be completed and improved:

1. Messages are validated using a CRC and are acknowledged (confirmed). A message should be retransmitted when not received or confirmed by the destination (the Ack and Read bits must be used correctly).

2. When the chat application is disabled, no chat messages should be shown. In that case, a message is not marked as read. If valid, it should be marked as valid.

3. All received messages with a valid CRC should be marked as valid. CRCs are not used for broadcast messages. A broadcast message is always unconfirmed.

4. Messages can be sent to ourselves, to a specific station, or all stations (broadcast).

5. Time messages are generated by one station only. It is sent as a broadcast. All stations should display a time message. This cannot be disabled.

6. A chat message can be sent to ourselves. In that case, the source and destination addresses are the same. This is allowed and the message should be displayed.

7. A message with a bad CRC should not be displayed by the recipient. The sender should send the same message again, with limited retries (e.g. 3). An error should be displayed when the message has not been confirmed.

8. Other data messages than chat or time messages (sent to an unknown SAPI) should be ignored by the applications.