# Token ring
## (TOR)

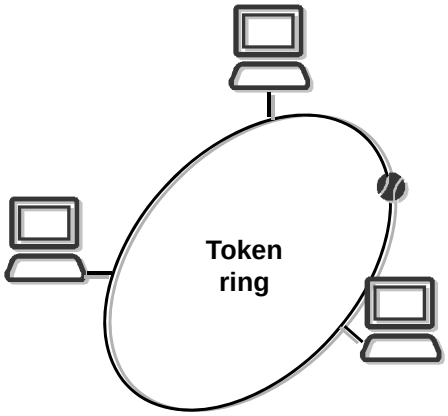
Pierre Pompili



## Token ring goals

- Goals
  - to associate the knowledge acquired in technical data processing and telecommunication
  - to develop an application of teleconference on distributed embedded systems
- Objectives
  - to practise the methods of formal description
  - to implement the layers of protocol
  - to practise the asynchronous programming
  - to implement an multi-task application

## Token ring functionalities




- Token ring topology

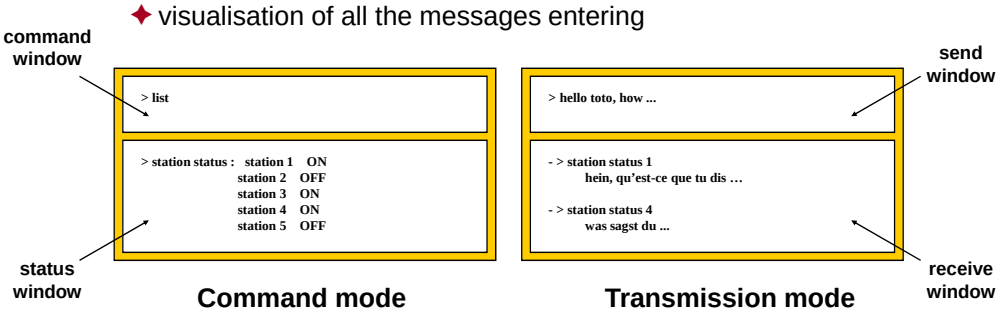

- Chat

- Timer

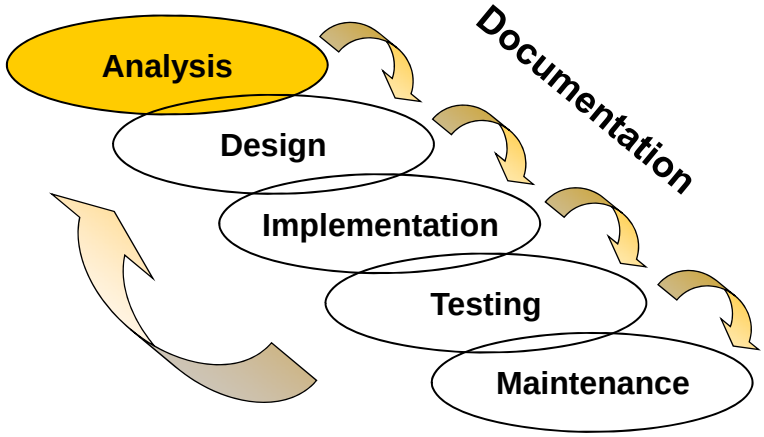


## Teleconference = chat

- Primaries functions
  - connection and disconnection of the network
  - directory of the users connected
  - selection of the correspondent
  - interactive mode with a correspondent
  - visualisation of all the messages entering



## Software life cycle

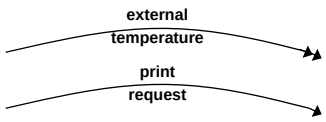





## Functional aspect

The structured analysis models the activity of a system in the form of data flow circulating between processes which treat them

- Data flow diagram (**DFD**)
  - data flow
    - **continue**
    - **discrete**



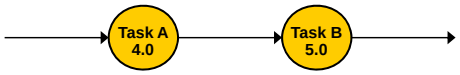

  - process


  - data storage

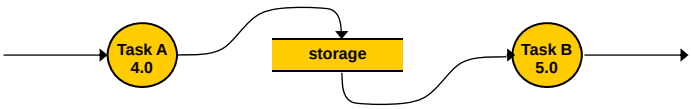

  - border termination

## Functional aspect (cont.)

- Causal relation (synchronous)
  - the arrival of the data causes the activation of the process




- No causal relation (asynchronous)
  - the data storage breaks flow tended

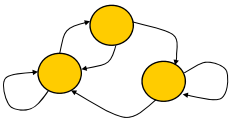





## Event aspect

- Events
  - activate the processes
  - influence the behaviour of the system

- A STR comprises a finished number of states possible
  - at any moment a STR is in only one state
  - the command logic is described by
    - states machine

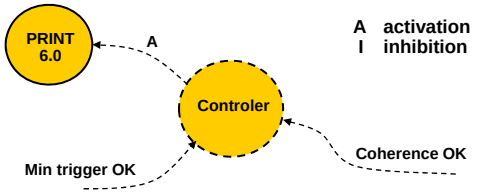


    - combinative system

| Red | Orange | Stop |
|---|---|---|
| off | off | No active |
| off | on | Active |
| on | off | Active |

## Event aspect (cont.)

- Actions on the processes
  - activation
  - inhibition
  - suspension (preserve the intermediate results)
  - resumption
- Representation



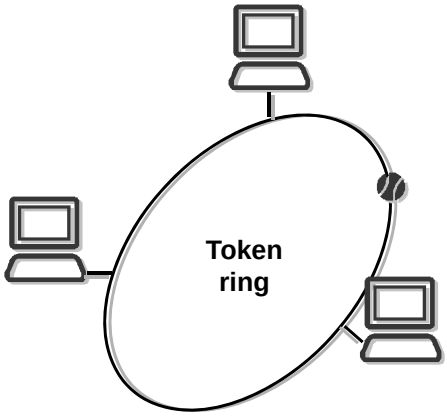



## Information aspect

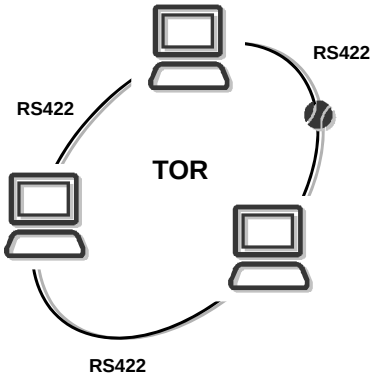
- The data are composed in a dictionary
  - simple data
  - composite data
  - times constraints
  - operating range

| *Input event* | *Indirect event* | *Output action* | *Response time* | *Comment* |
|---|---|---|---|---|
| **Red on** | **feedback** | **Active break** | **20 ms** | **Bypass human** |
| **Orange on** | **feedback** | **Active klaxon** | **40 ms** | **Check HP** |

## Token ring topology



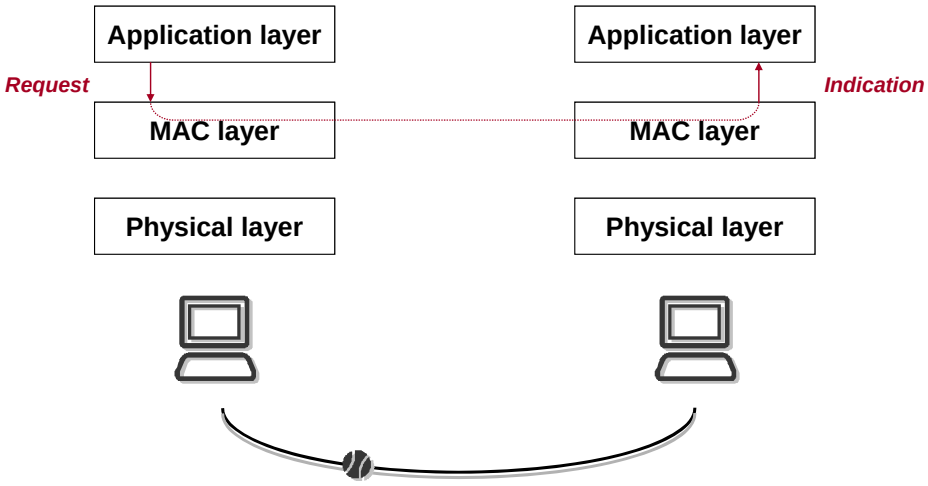

- Real ring topology



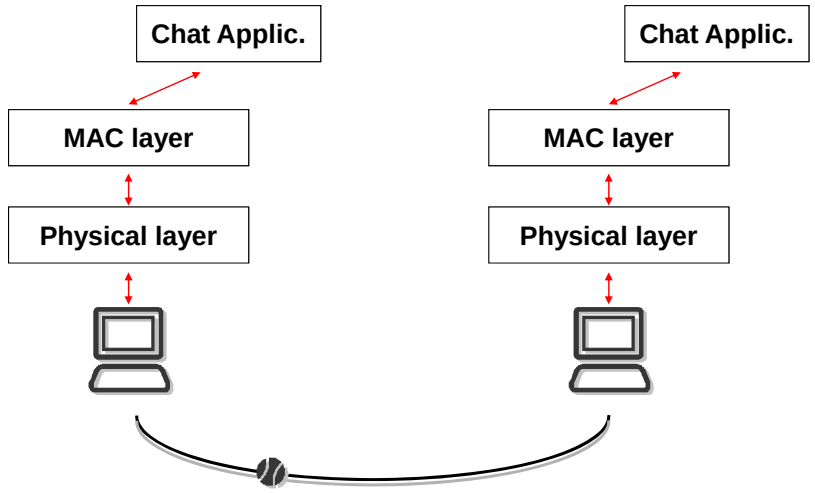

- Point-to-point topology



## TOR OSI model

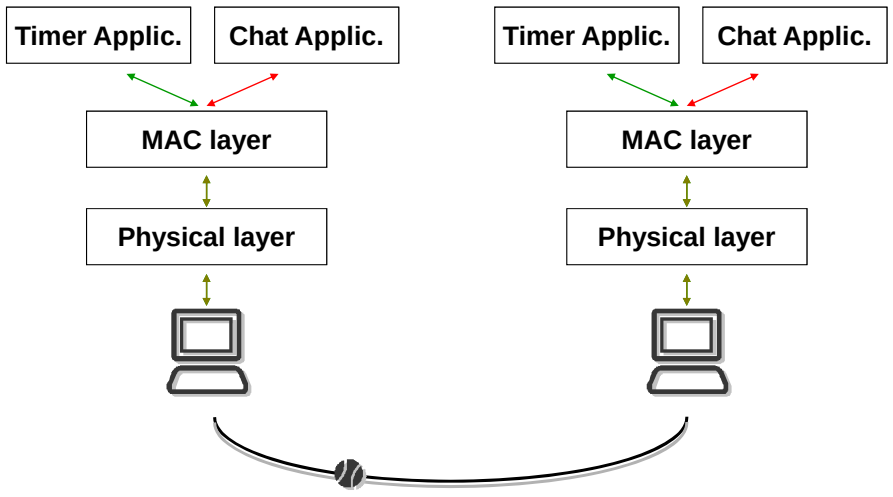


## TOR module model

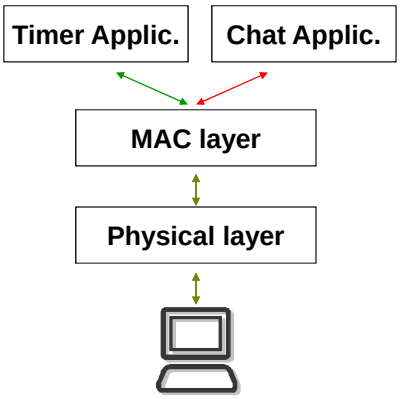





## TOR module model



## TOR Constraint on the model

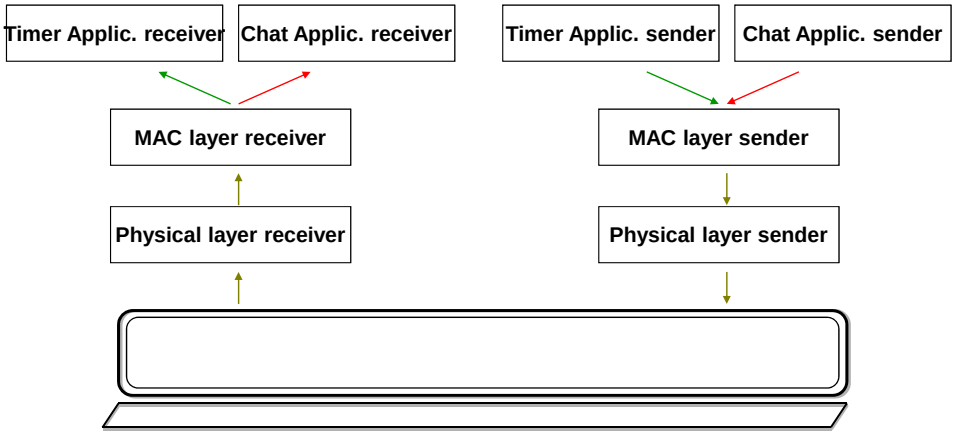



**Data flow diagram (DFD)**

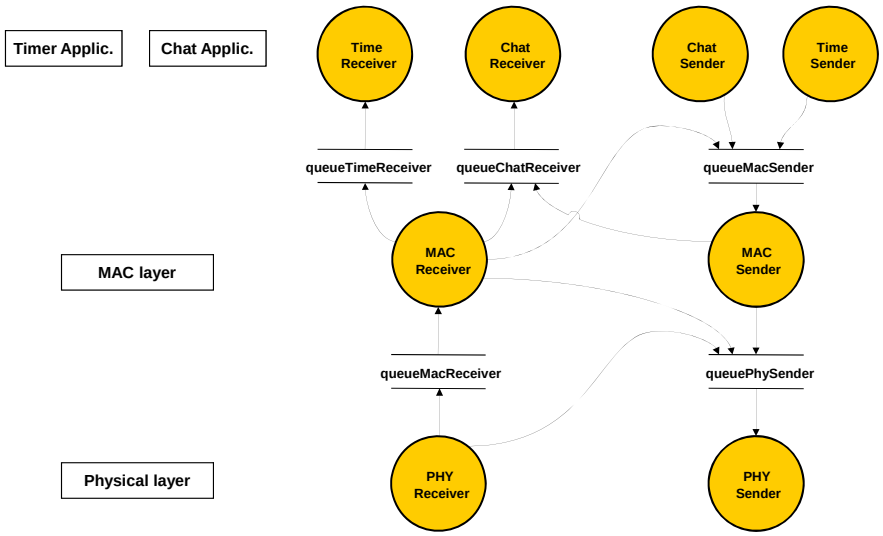

**Discrete Data flow**
**only directional arrow**



## Splitting model

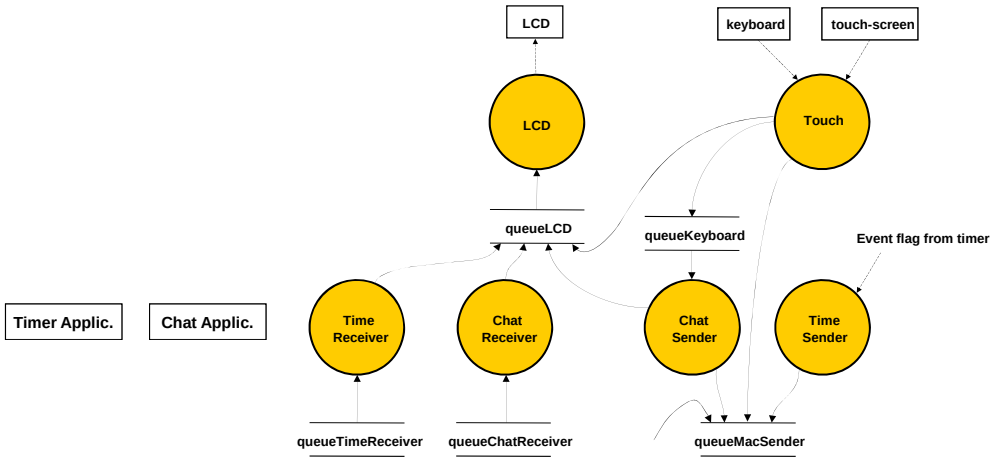


# DFD TOR module 1/2



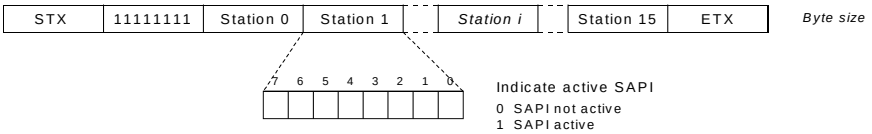



# DFD TOR module 2/2

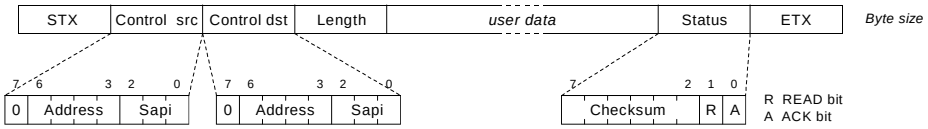


## Frames structure

- Token frame



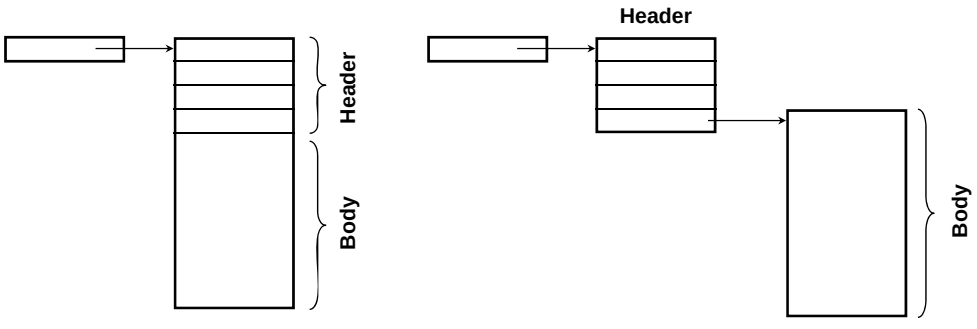

- **Data frame**






## Message structure

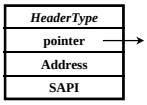
- Message with variable length




- **First method**
- **Second method**

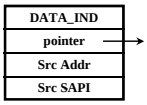
## Message header definitions 1/2

- Header definition




- **Example from MacReceiver**

➡ **to Applications receivers**




➡ **to queueMacSender**




## Intertask communication

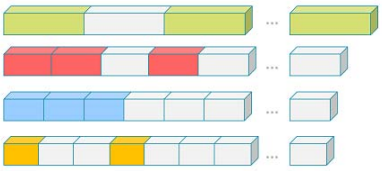
- Message queues
  - All entries are the same size
  - messages are copied between the queue area and application data areas
  - post message at the end of the queue (task pending if full)
  - spend for a message from the queue (task pending if empty)
  - specify an optional time limit

```
queue_macS_id = osMessageQueueNew(2,sizeof(struct queueMsg_t),
                                  &queue_macS_attr);     create queue

osMessageQueuePut(queue_macS_id, &queueMsg,
                  osPriorityNormal, osWaitForever);      post message

osMessageQueueGet(queue_macS_id, &queueMsg,
                  NULL, osWaitForever);                  wait for message
```

# Memory allocation

- memory allocation methods
  - Global Memory Pool
    - uses a single global memory pool for all objects. Memory fragmentation when objects with different sizes are created and destroyed

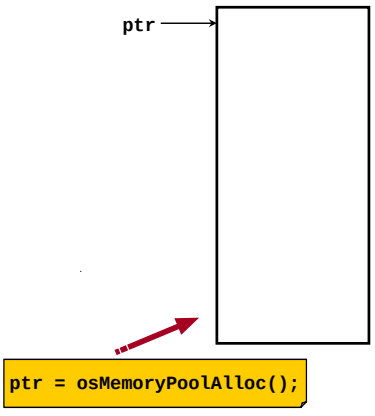
  - Object-specific Memory Pools
    - uses a fixed-size memory pool for each object type. The method is time deterministic and avoids memory fragmentation



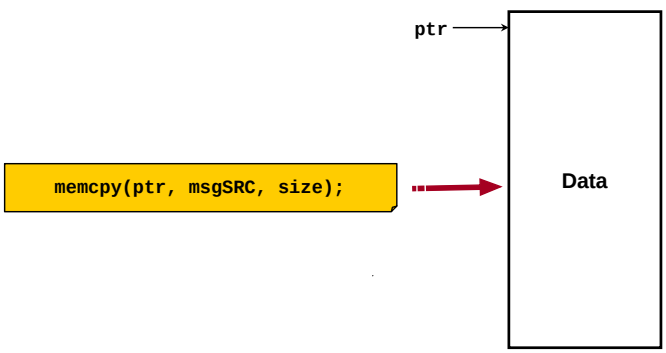
```
ptr = osMemoryPoolAlloc(memPool, osWaitForever);        new

osMemoryPoolFree(memPool, ptr);                         dispose
```



# Message structure implementation 1/6

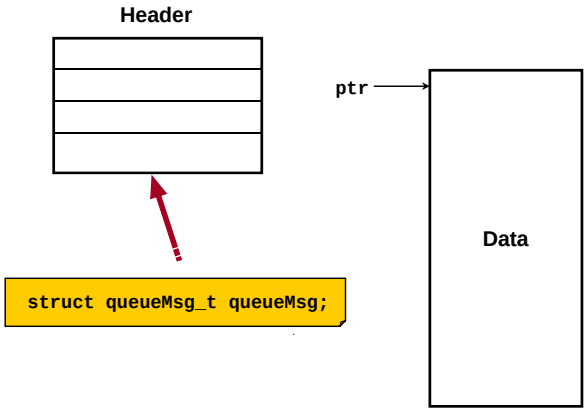


## Message structure implementation 2/6

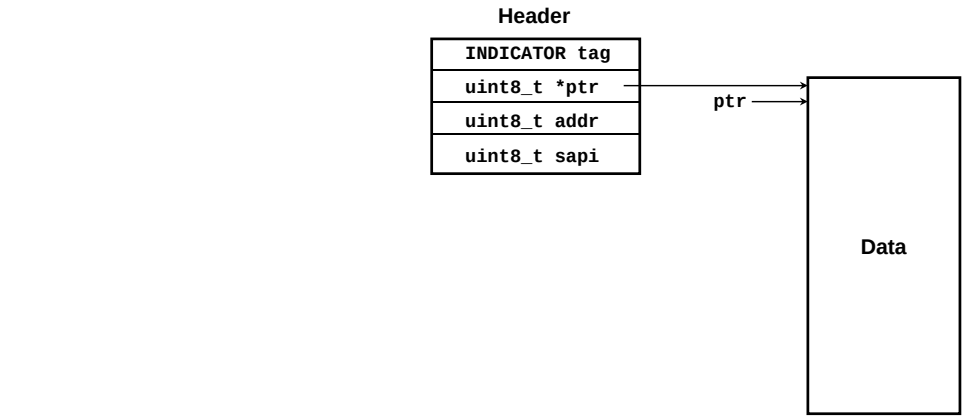





## Message structure implementation 3/6

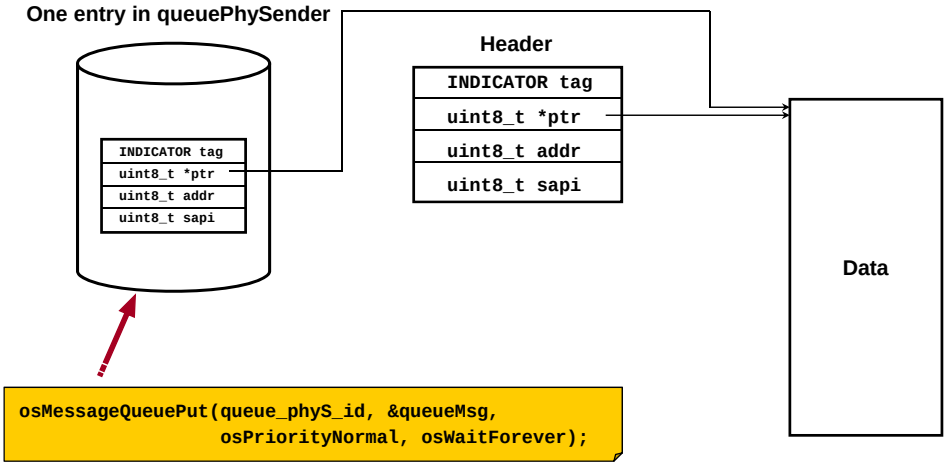


## Message structure implementation 4/6



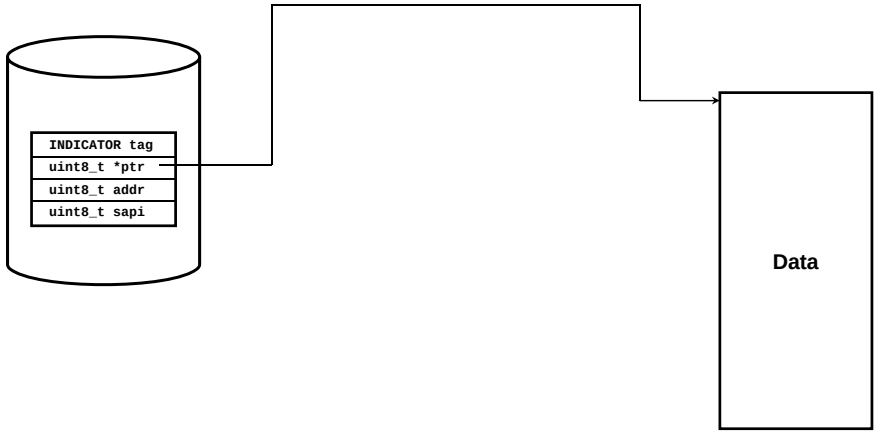



## Message structure implementation 5/6



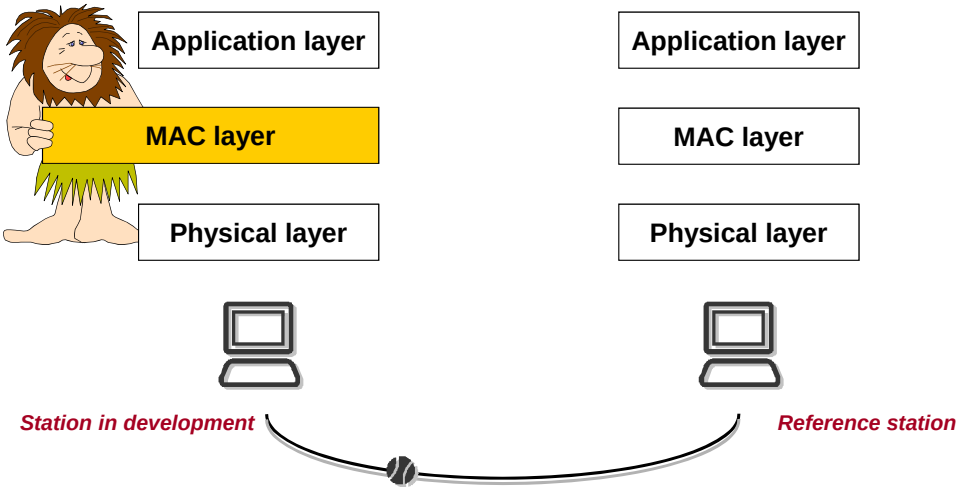

```
osMessageQueuePut(queue_phyS_id, &queueMsg,
                  osPriorityNormal, osWaitForever);
```

## Message structure implementation 6/6

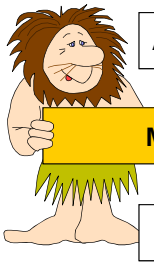





## TOR test with a distant reference station



# TOR test with a virtual reference station



Application layer

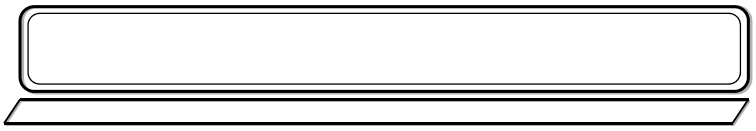
MAC layer

Physical layer

Application layer

MAC layer

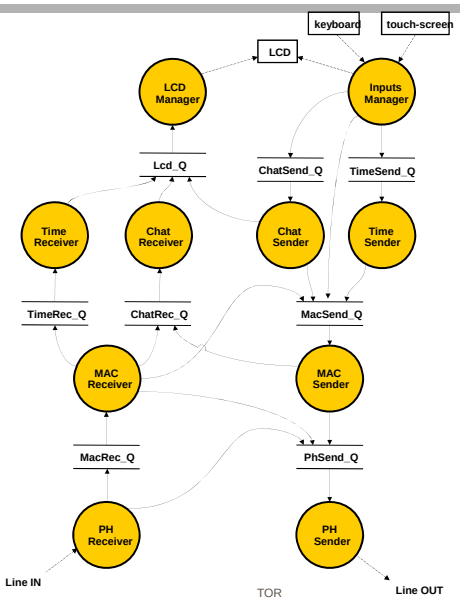
Physical layer



*Station in development with virtual reference station*



# DFD in design application



# DFD with virtual distant station