⇒ Com stack:

↳ what is PDU?

: Packet Data unit

Consist of SDU : service Data unite

+ PCI : protocol Control Info

PDU = SDU + PCI

Consist of : Signals + signal groups

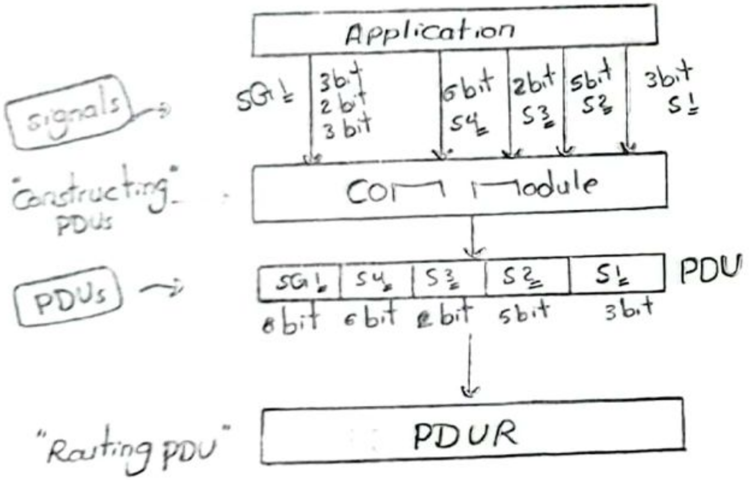
↳ signal : smallest entities of data exchange

↳ signal group : Two or more signals have to be transmitted together

↳ signal - signal group :

← output of Application Layer (SWC)

→ Input to service Layer (Com Module)



↳ Com Module:

↳ Transmission Mode: (PDU Config)

1- Direct: Application Trigger Com to send PDU.

2- periodic: Com send PDUs based on time trigger event.

3- Hyprid: " " " " " Application or time Trigger.

4- None : Triggered by a lower layer

↳ Transfer property: (signal config)

1- Triggered: only with PDU config Direct

↳ the PDU "Direct" will be sent in case of the triggred signal sent from APP

Note: even though the signal value equal the precious or default signal value

2- Triggered on change: only with Direct PDUs

↳ the PDU "Direct" will be sent in case of the triggred signal has a different value from APP rather than the previous value

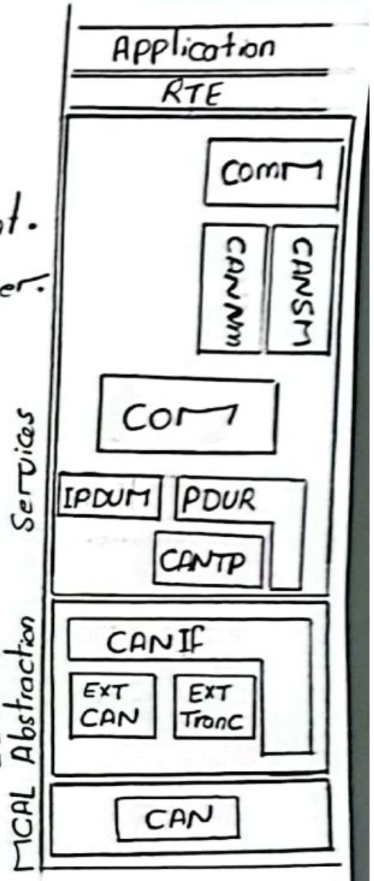
3- pending:

↳ with periodic/Hyprid PDU the OS will trigger PDU

↳ with Direct PDU and has at least one signal -signal group

↳ Triggered - Triggered on change

- Direct PDUs must have { Triggered / Triggered on change } signal - signal group
- periodic PDUs may have pending signal - signal group
- Hyprid PDUs may have all signal types



"Com stack"
(CAN stack)

↳ com filter : "In case of receiving only"

1- No filter :
2- signal values within range
3- " " outside "
4- " " equal to
5- " " Not equal to

↳ Update bit :

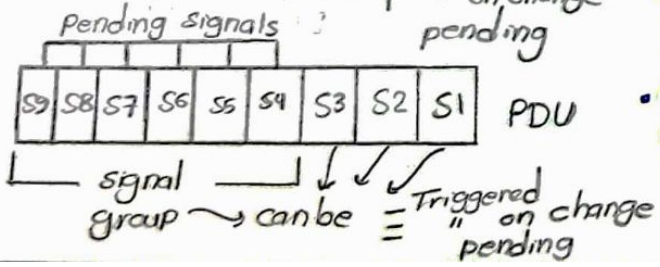
why update bit? "To reduce overhead"
: if we have a Direct PDU, only one signal is updated
- the whole PDU "updated - Not updated" signals will be sent after updating any signal on PDU
- after using update bit, only updated signals will be sent

↳ signal Grouping :

- we can group number of related signals within one group
  - ↳ To make sure all signals arrived at the same time

Note : all signals within group must be pending but the whole group can be { triggered / " on change / pending

Example



↳ Deadline Monitoring :

- Tx Confirmation
- Rx Indication

↳ Timeout vs First Timeout :

- First Timeout : "for Rx Indication" Normal Timeout + com Initialization time at the begining of process
- Timeout : a time must be considered between receiving the two PDU, the first receiving one not one of them

↳ each signal on PDU can has its timeout
- the PDU timeout will be equal to the smallest timeout.

↳ if timeout value reached com module Notify the Application

↳ PDU Grouping :

- we can make a group of PDUs as a unit group of Date

↳ benifits :
- we can stop the whole group from sending - receiving
- we can start the PDUs group again

↳ signal Invalidation :

↳ • Transmission case / Receiving case
if the com module take an Invalid signal value "out of config range"
- com module may has a default value "use it to complete proces"
- com notify the app about the Invalid signal value

↳ PDUR :

- Handling Transmission from upper Modules
  - ↳ config each Msg PDU
    - Msg source
    - Msg destination
- Multicast Transmission
  - ↳ send Msg on one more Network
    - ↳ Using PDU getwaying
- Routing path grouping
  - ↳ Enabling / Disabling bus on getwaying or multicasting

↳ IPDUM :

- PDU Layout Multiplexing
- Multiple PDU to container Handling

↳ CANTP:

- segmentation of long msgs In Transmission case
- Reassembly of segmanted Msgs In receiving case
- PDU padding to be compatable with layers requirements

↳ CANIF:

- Abstract Driver Access
- Transmission Buffering
- Receiving Indecation for upper / lower layers
- software filtering
- DLC check
- Define Msgs : - sent / received
  - Msg ID
  - upper layer
    - •PDU •CANTP
  - Hw object reference

↳ CAN Driver / Tranceiver:

- Define Hw Controller :
  - Baudrate - Filter MASK - Hw object
- Mode of operation : -polling -Interrupt
- for Tranceiver : -num of channels - DIO/SPI

→ Communication Management : "Network Controll path"

↳ ComM : "Com Manager"

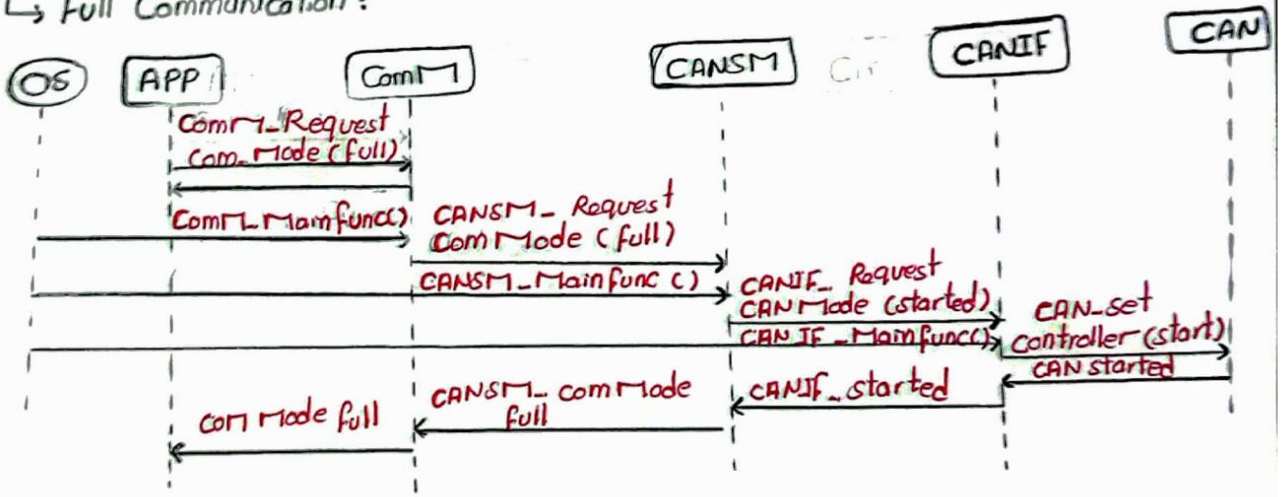
→ sys stack

- Coordinating the availability of the bus
  - No communication]
  - full "] com mode request
  - silent "]
- Coordinating the requests from different swcs on the same channel

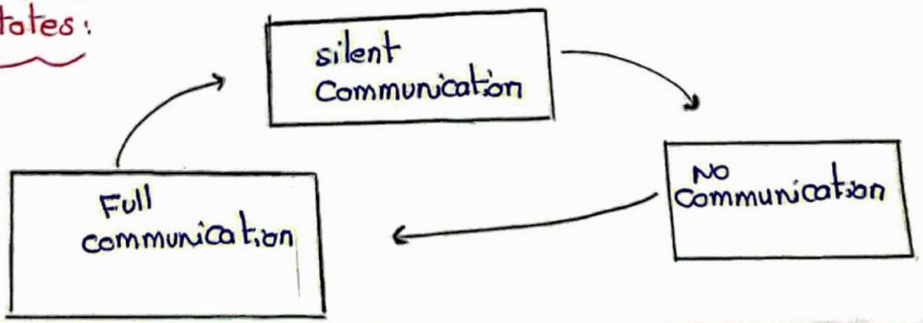
↳ CANSM : "state Manager"

- Manage the state machine for each can controller - Tranceiver according to requested Mode (- full - No - silent)

↳ Full Communication :

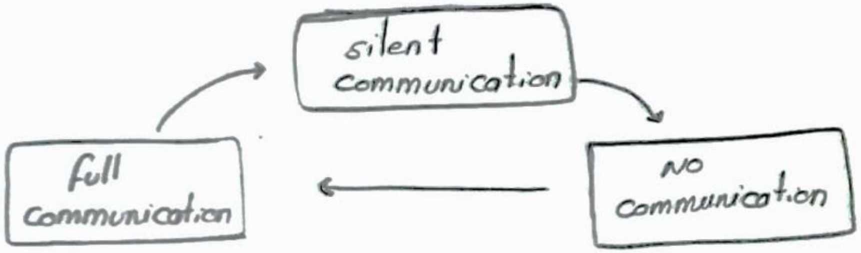


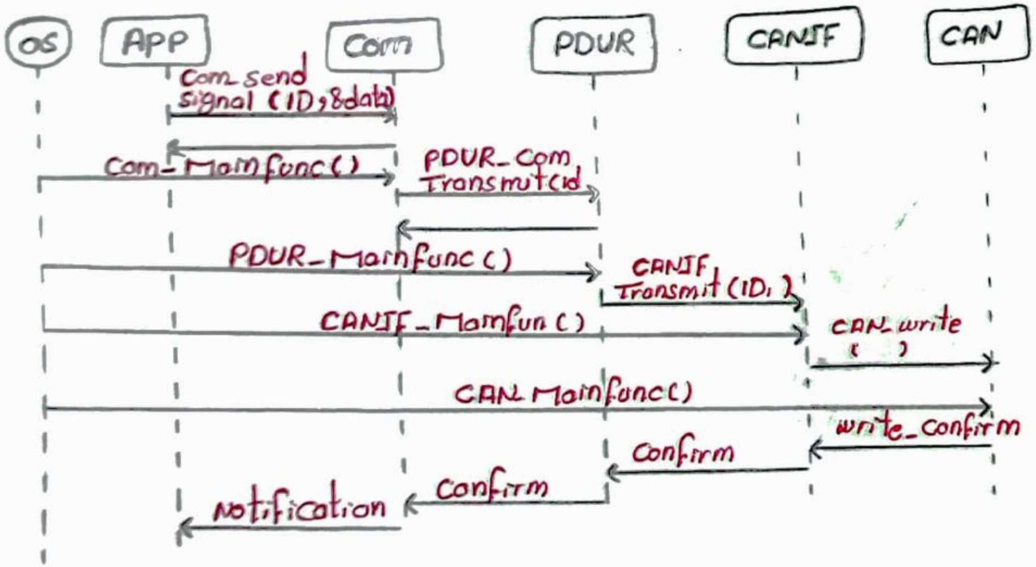
→ Com states:



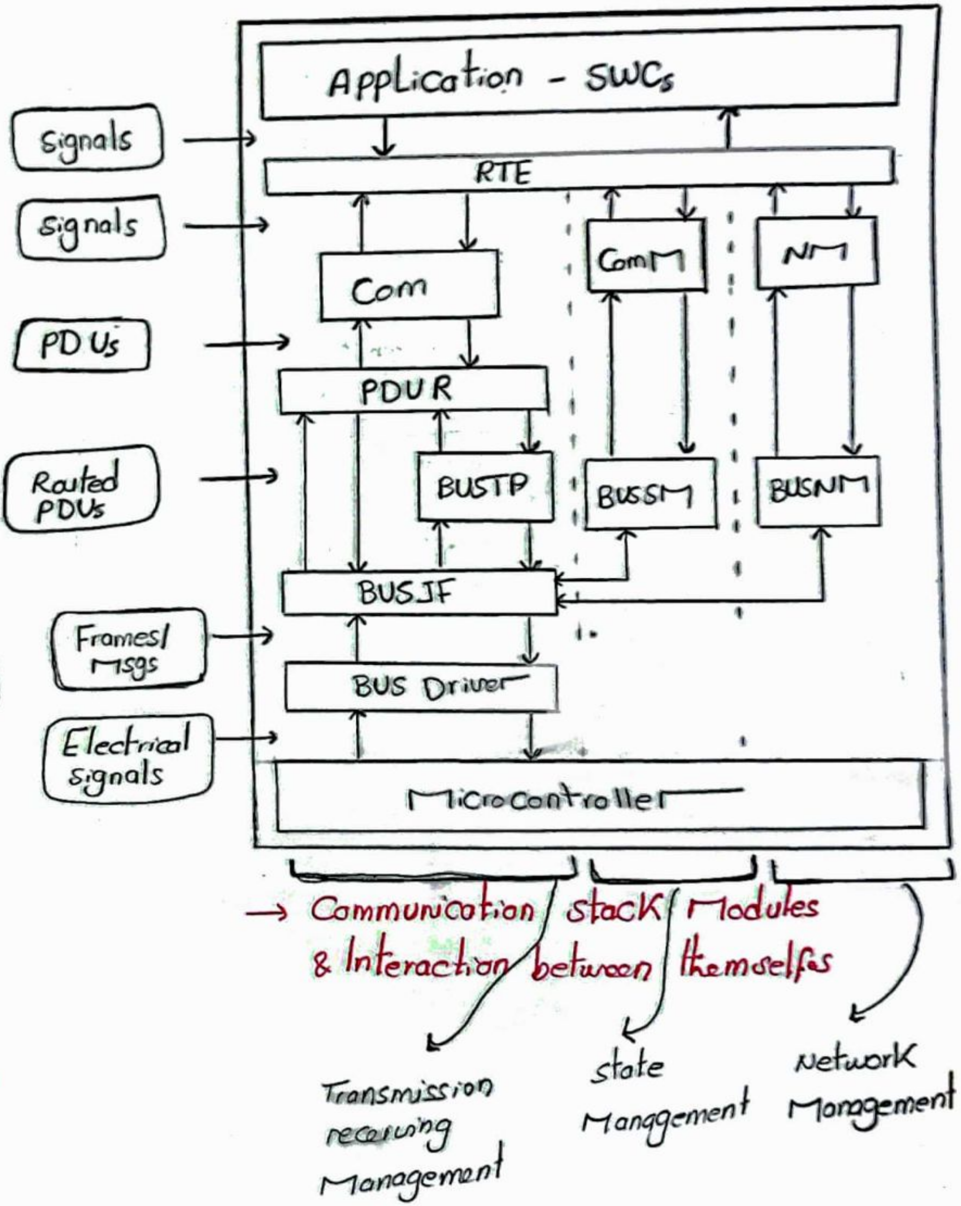
↳ Com states:

- In case of No communication request received (from full → No communication)
  1- CANSM request a silent request
     ↳ Read only case "Tx : offline"
  2- after a period CANSM send No communication request to CANIF





→ Sending Message sequance



→ Communication stack Modules & Interaction between themselfes

→ Network Management : "To Reduce power consumption"

↳ NM Messages :

- Msgs Has a fixed IDs [500 : 5FF]
- purpose : to manage network not exchange data

→ According to the small range of Msgs IDs

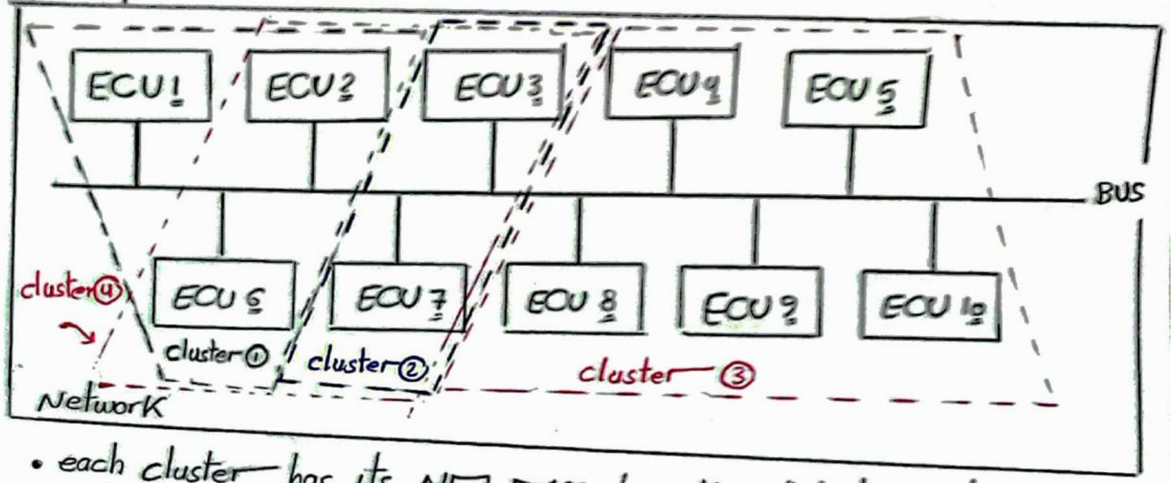
↳ partial Networking ;
using of Data Feild + Msg ID
to define which cluster to wakeup

Msg ID	1byte	7byte (56 bit)

✓ Control Bit vector

✓ to control 56 cluster
each bit refer to a cluster

Smart Trancaiver check
both Msg ID + PNC bit
"partial Network" ↵
cluster

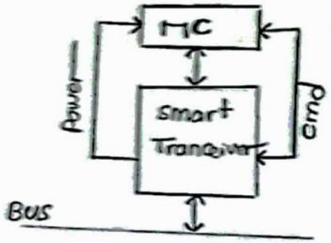


- each cluster has its NM Msg to wakeup "start operate"
- a cluster can be used to wakeup another cluster

? How ECUs sense Msgs In case of No power ?

↳ Using a smart Tranceiver

- smart Tranceiver is the power provider to MC
- Monitoring the Bus
  ↳ if sense a NM Msg to wakeup → provide power to MC
- MC doesn't receive any Msg "send cmd to trancever" to prevent power from MC



↳ using sleep Mode of MC
↳ "low power mode"

↳ The ECU sent an NM msg "Active ECU" → Active wakeup bit → ①

↳ " " Received " " "
and doesn't sent an 'NM msg "passive ECU" → " " " → ⓪

] control Bit vector

except "ACK Msg"

↳ N of Msgs all bits "o"

↳ wake up sequance:

① smart Trancaiver received NM Msg

② " " connect power to ECU to wake up

↳ sequance of wake up

① ECU call ECUM to start the system

② call ComM → BusSM to request a full communication

③ BusNM called by NM to send a passive NM Msg as on ACK of wakeup

④ start receiving the actual Msgs

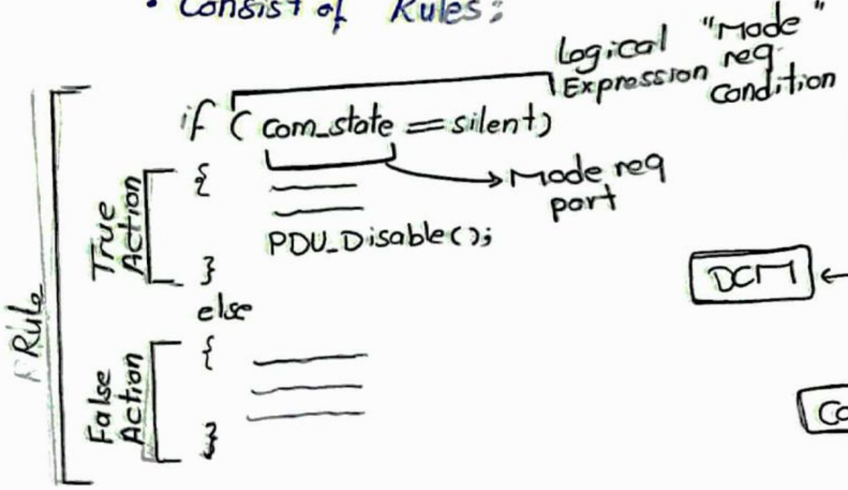
⑤ check for the NM Msg "aperiodic function" to resume receiving

⑥ if there is No NM Msg after a period

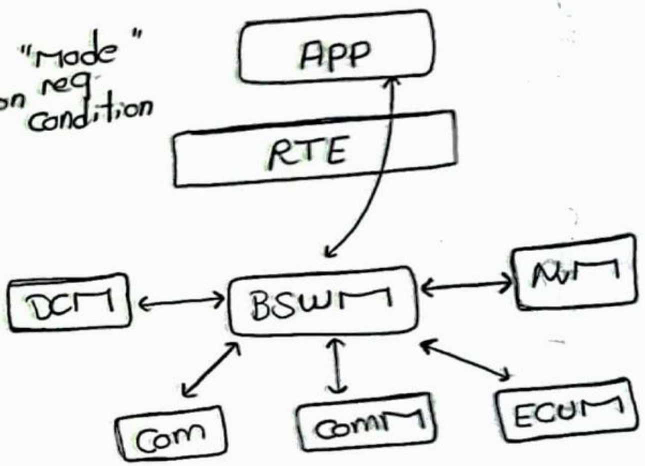
⑦ ECU call Trancaiver to shutdown"

→ BSWM : Basic software Mangger

- centrolized module communicate with other modules and SWCs
- Take Inputs from modules, request services from modules
- Consist of Rules:



```
if ( com_state == silent)
{
    ____
    ____
    PDU_Disable();
}
else
{
    ____
    ____
    ____
}
```



→ Configuring BSWM:

- Create mode request ports
- " " " conditions
- create True / False Actions
- create Rule

```
if ( com_Mode == No_Com)
{ RTE_stop ();
  DEM_shutdown();
  NvM_writeAll();
  ECUM_shutdown( —);
}
```

off/reset

↳ shutdown sequance on BSWM