12.15 – 12.40

Why are RTOS so complex to learn and use?

Gerard Fianen

INDES-IDS BV - Embedded Software Development

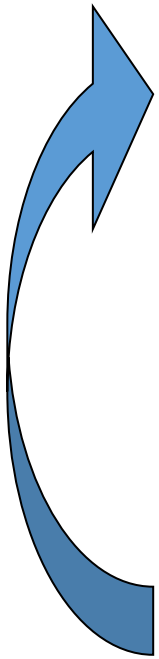
"The choice of professionals"

info@indes.com

www.indes.com/embedded

Tel: 0345 - 545.535

Why an RTOS of platform ?



Main loop:

module 1

module 2

module 3

module 4

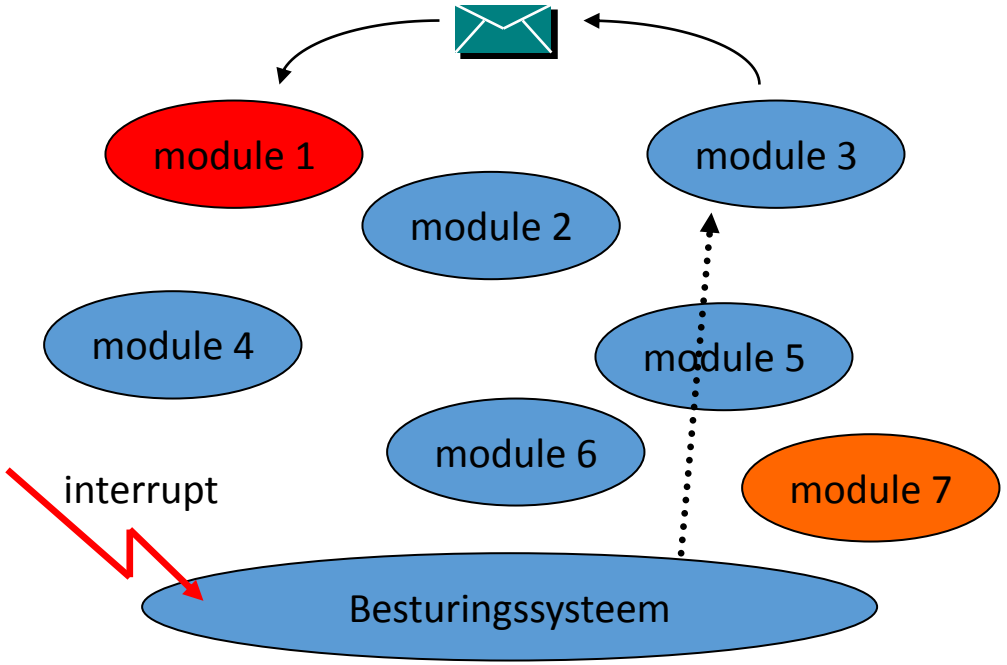
module 5

module 6

module 7

Loop main.

interrupt



The RTOS should fit the application

-Code size / performance / costs

- Real-Time vs non-Real-Time

- Available standard modules

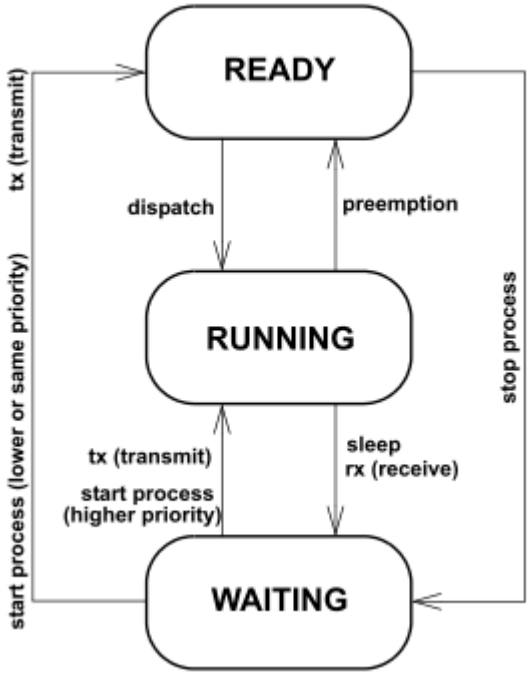
- Single-CPU v.s. distributed system

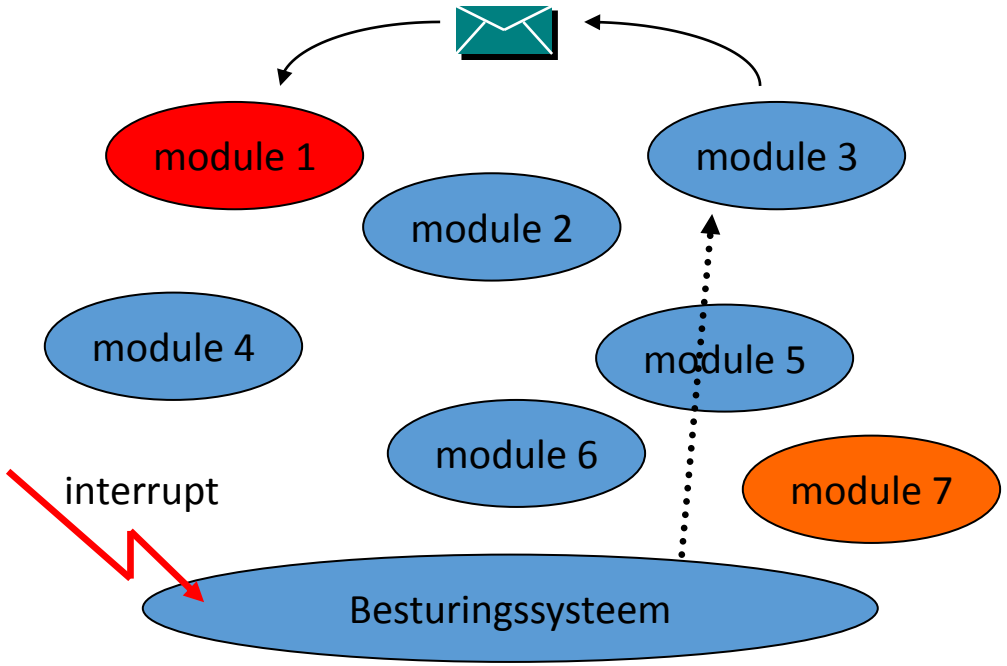
- Non-stop v.s. Non-critical system

- RTOS methodology

What does an RTOS offer?

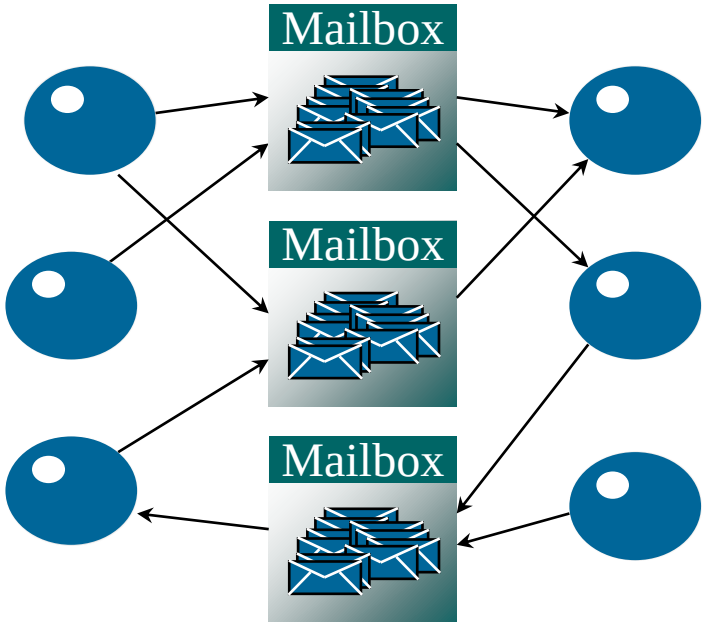
Inter-Process Communication





Scheduling

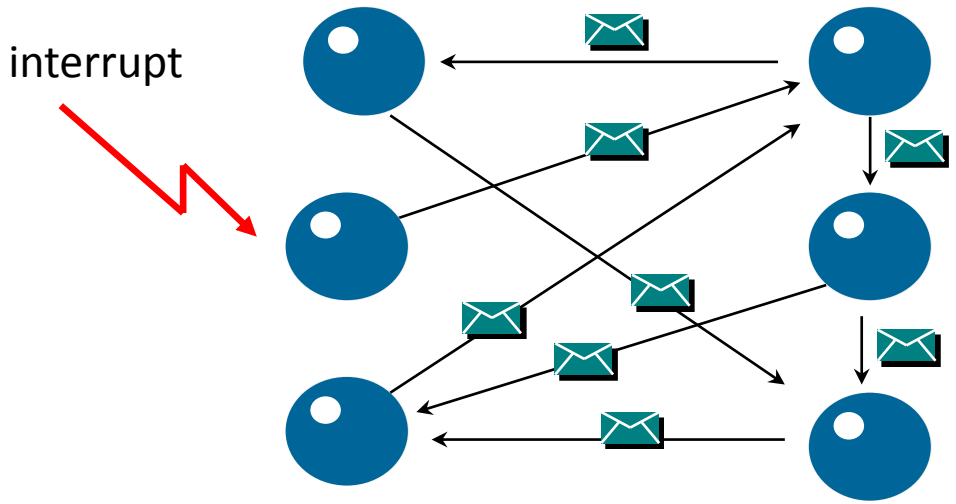
IPC: Conventional RTOS



Indirect IPC
Shared memory
Bottom-Up design
Single-CPU system
Complex API
User set-up IPC

- Process Synchronization
  - Events
  - Semaphores
- Inter Process Communication
  - Message Queues
  - Mailboxes

IPC: Asynchronous Direct Message Passing

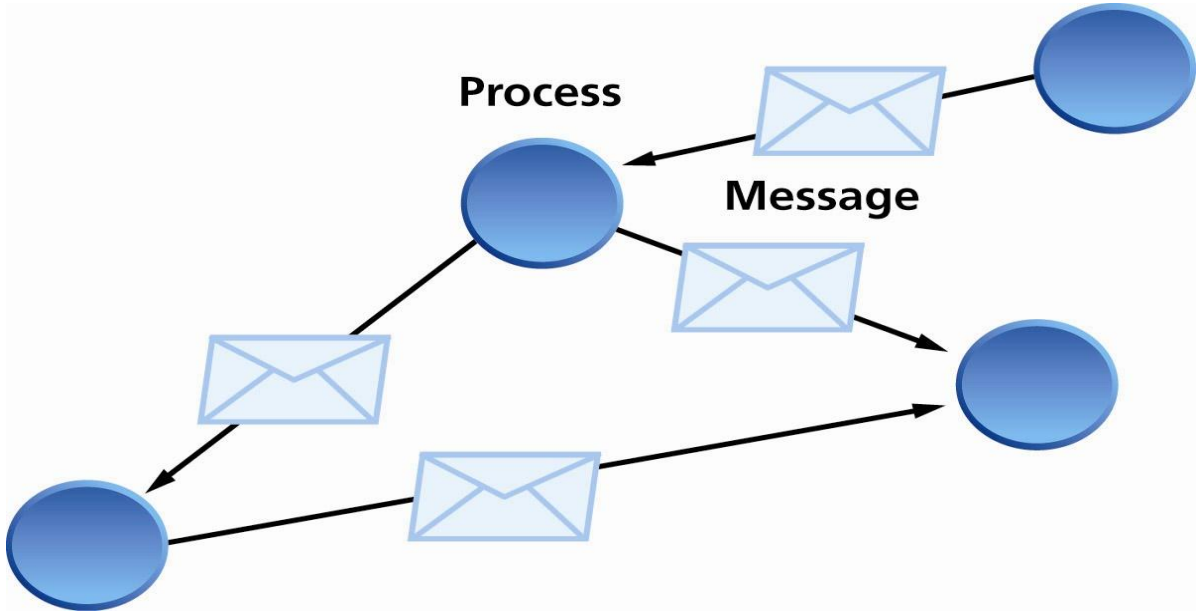


Direct message passing
Black-box approach
Top-Down design
Multi-CPU systems
Simple API
Automatic IPC

SCIOPTA
QNX
ENEA

IPC: Asynchronous Direct Message Passing

- Message-passing simplifies system design
- Ensures both Process Synchronisation & Process Communication



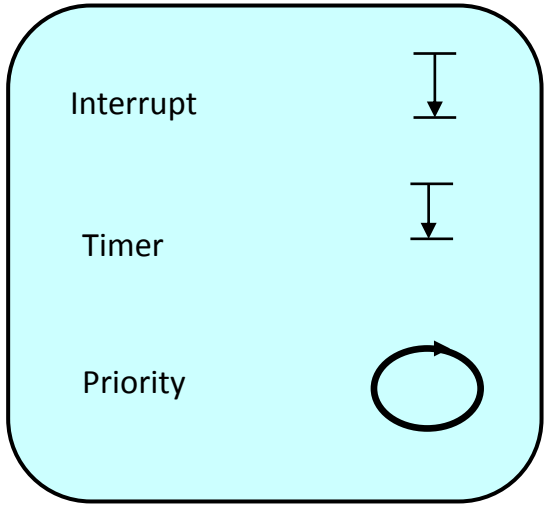
- Top-down system level design
- No set-up of queues required
- Direct process to process
- Ownership is never shared
- No synchronisation is needed
- Messages can be traced and monitored

SCIOPTA

Processes

Sciopta Processes

- Contains application code
- Each is a **separate** object
- Each has its **own message** queue
- **Sends and receives** messages
- 2 main types : **Prioritized or Interrupt**



SCIOPTA

Messages

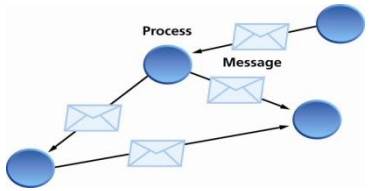
- No fixed format for data
- No requirement for shared memory
- High performance
- Can be traced and monitored
- Safe

ID (name) : START_MOTOR
Owner :
Sender : control
Receiver : motor
Data : RPM=2000
Display="motor started"

C-Structure (union)

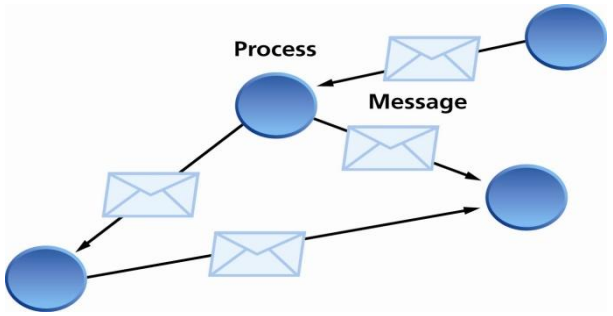
Administration block	ID	Data Field	End mark

Internal	Size	Sender	Receiver	Owner



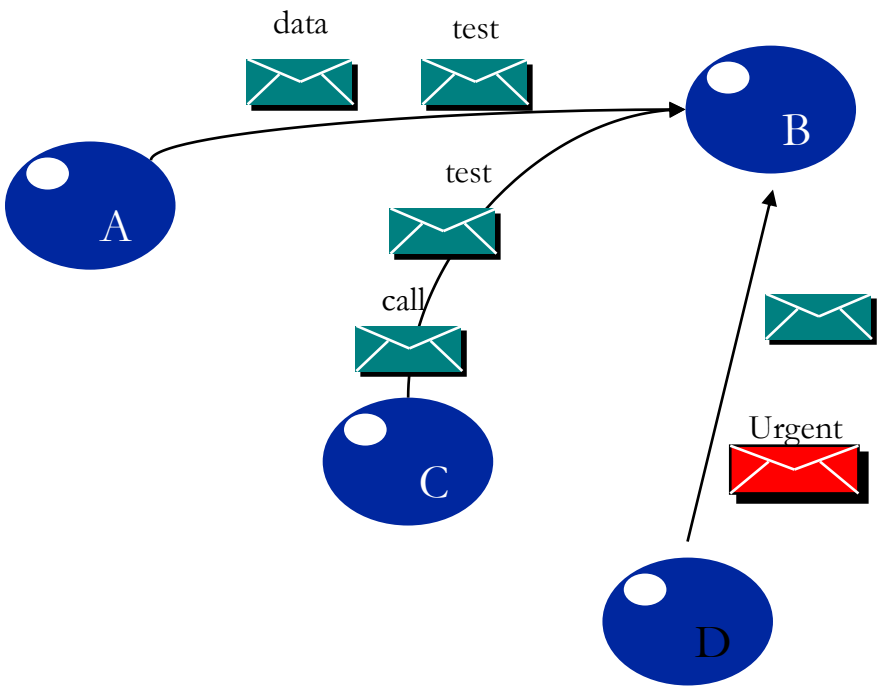
Basic call set

• sc_msgAlloc	➢ Allocate message
• sc_msgTx	➢ Send message
• sc_msgRx	➢ Receive Message
• sc_msgFree	➢ Free Message buffer
• sc_msgSndGet	➢ Who sent the message ?
• sc_sleep	➢ Delay process



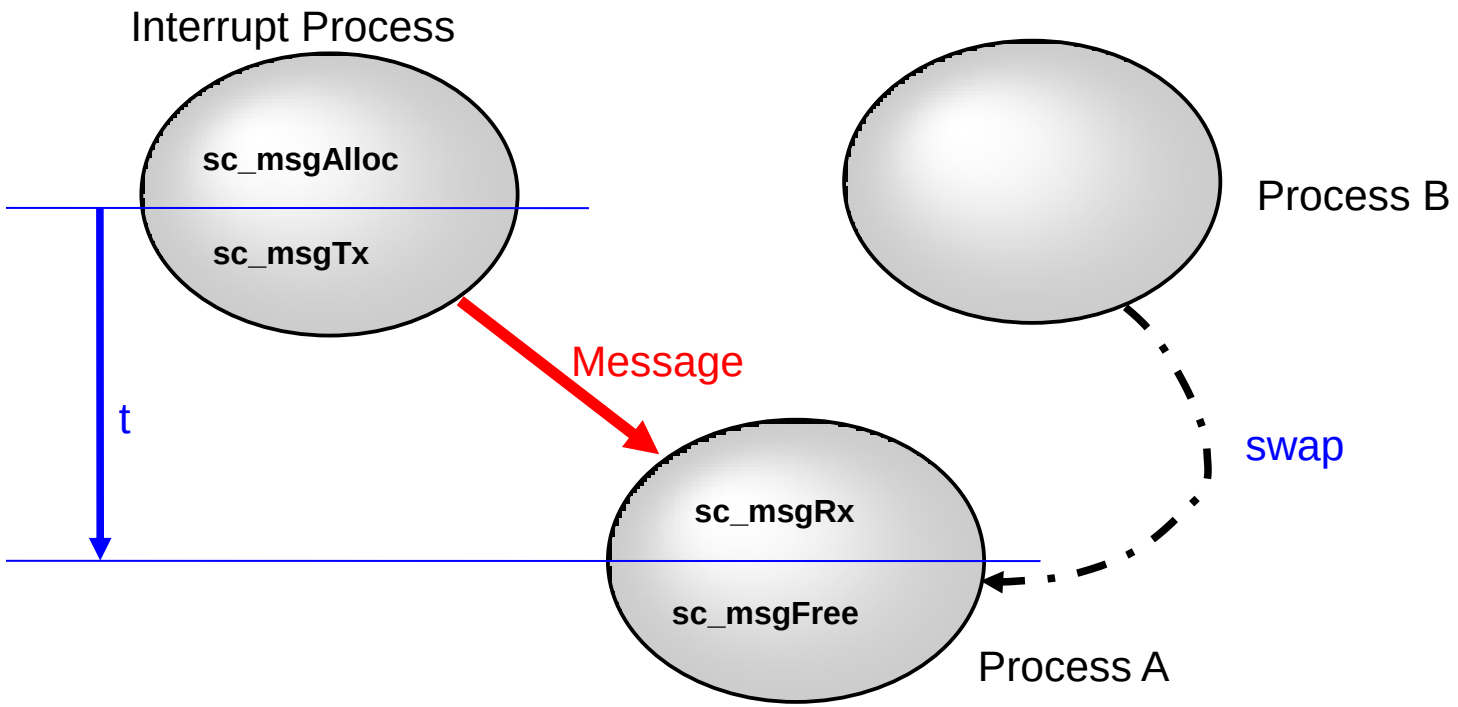
Message selection

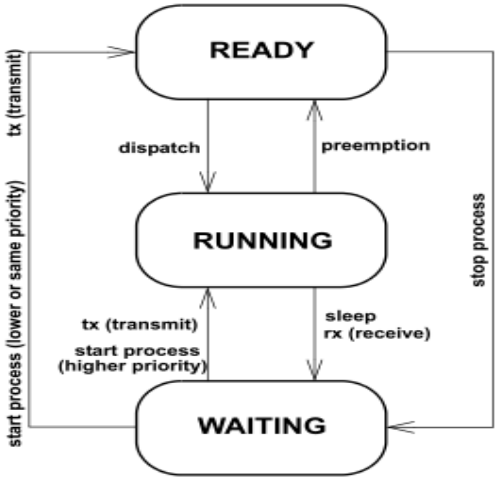
Message Selection



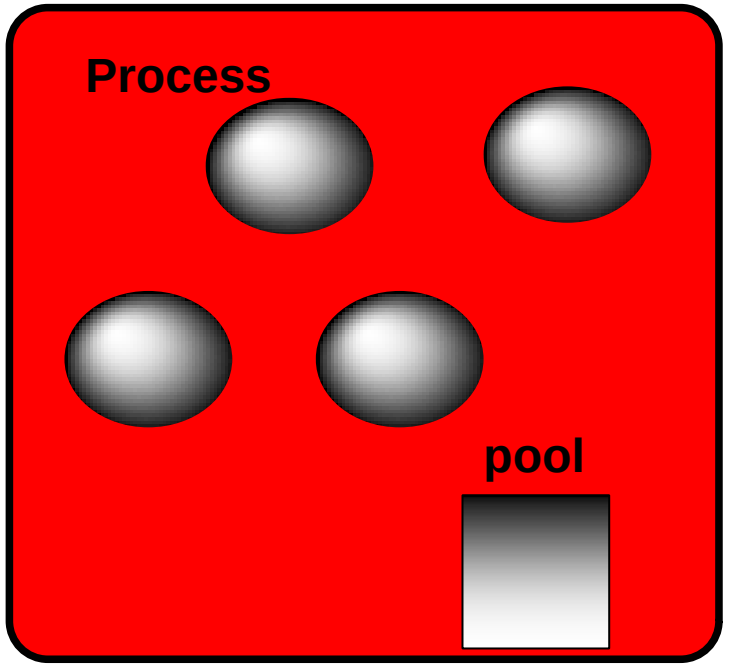
- ‘receive’ primitive allows message selection
- Message can be selected by MsgID
- Message can be selected by Sending Proc
- Only the needed message is received
- ‘Urgent’ message can be read first

Message selection





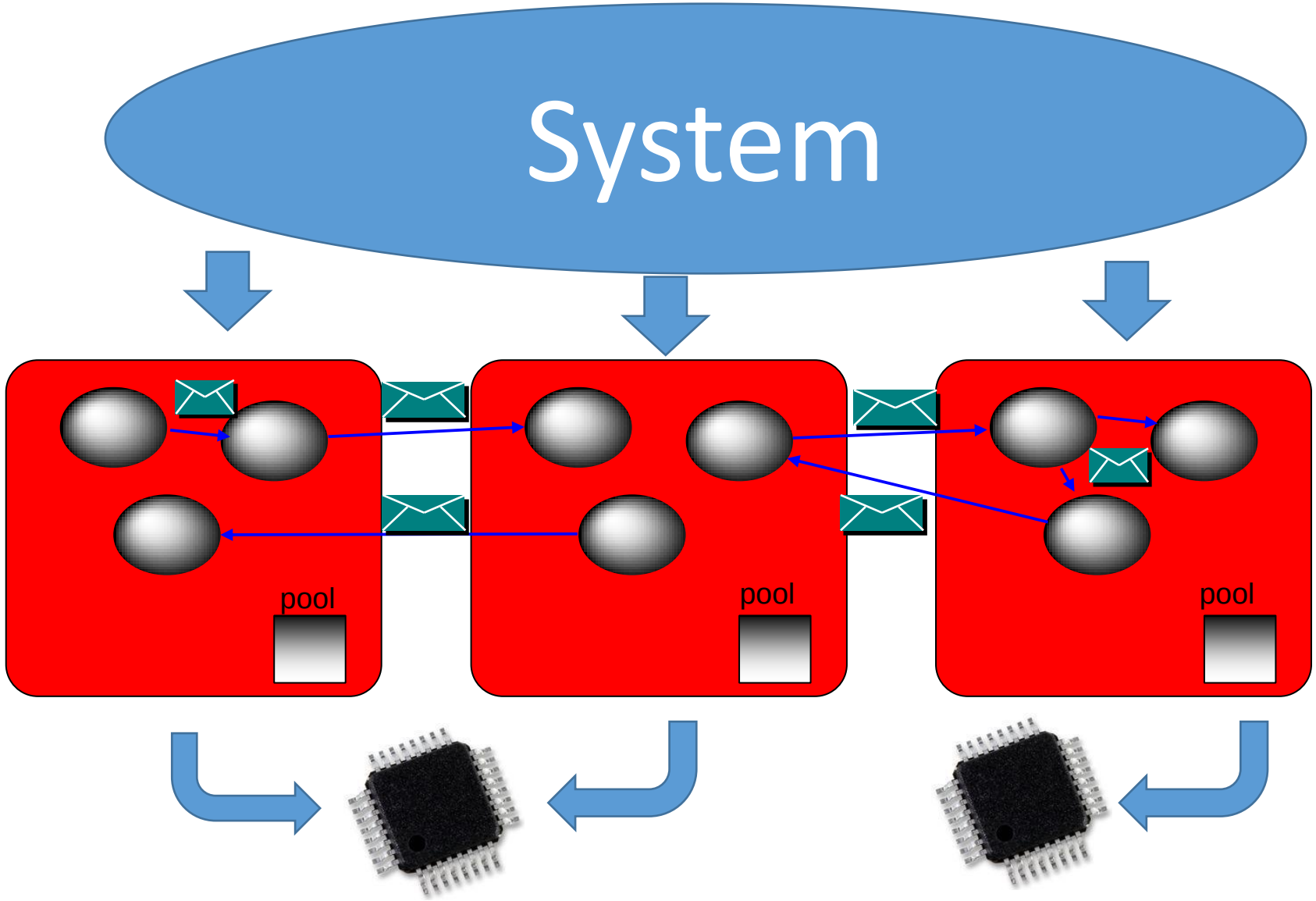
Modules



Module

- Processes can be grouped into modules
- Modules can be started and killed
- Automatic clean-up of resources

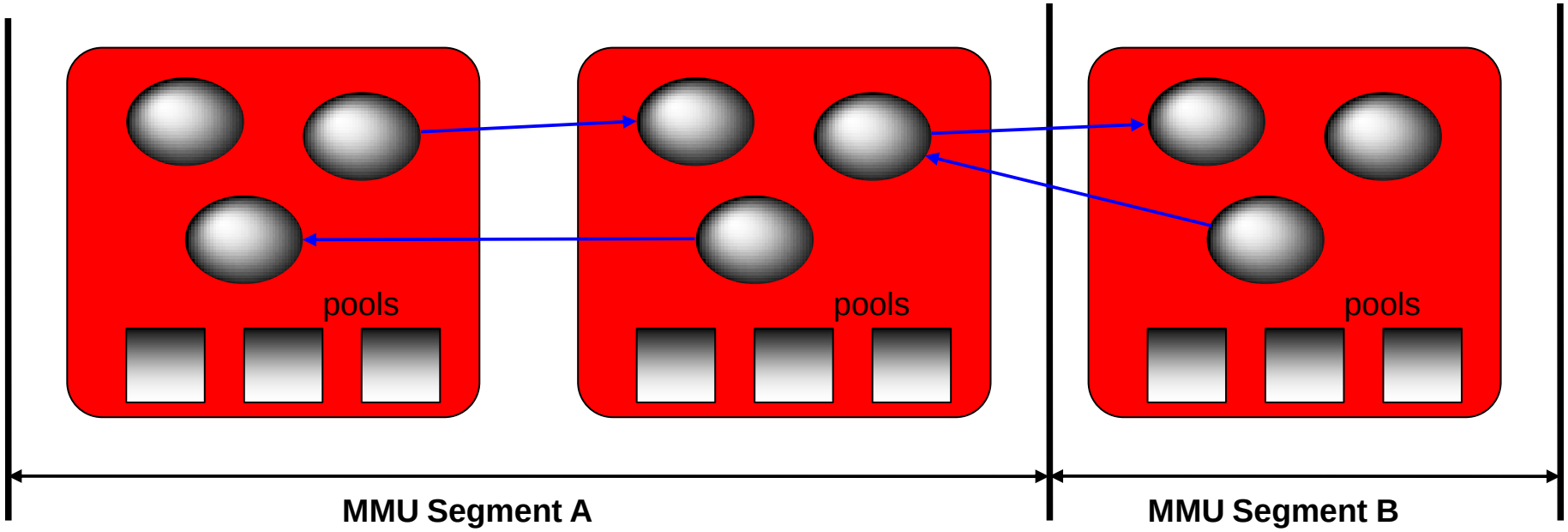
Top-Down Design - Modules



Top-Down Design – Memory Protection

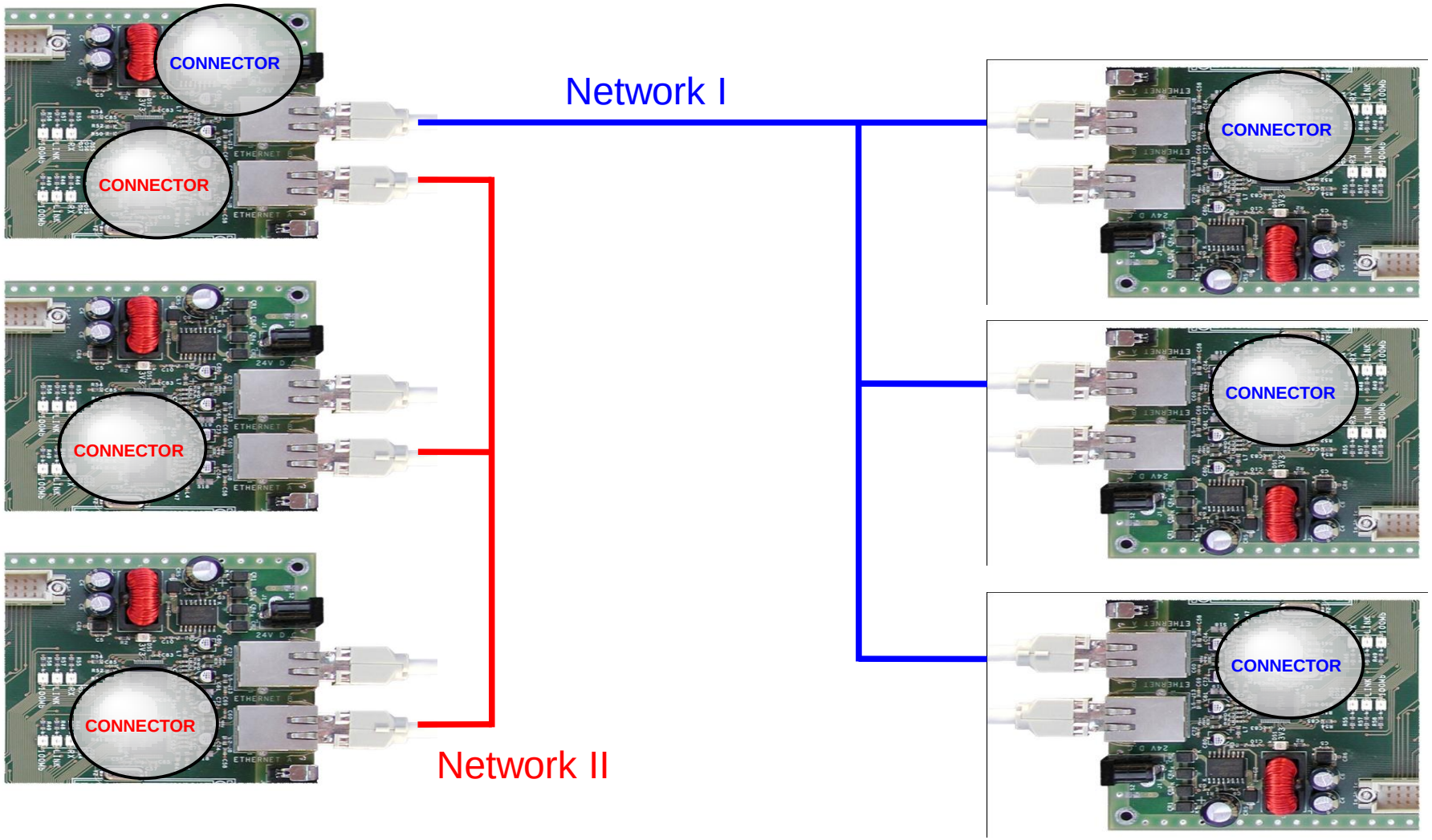
Safety certified :

IEC-61508 (SIL-3)
ISO-26262

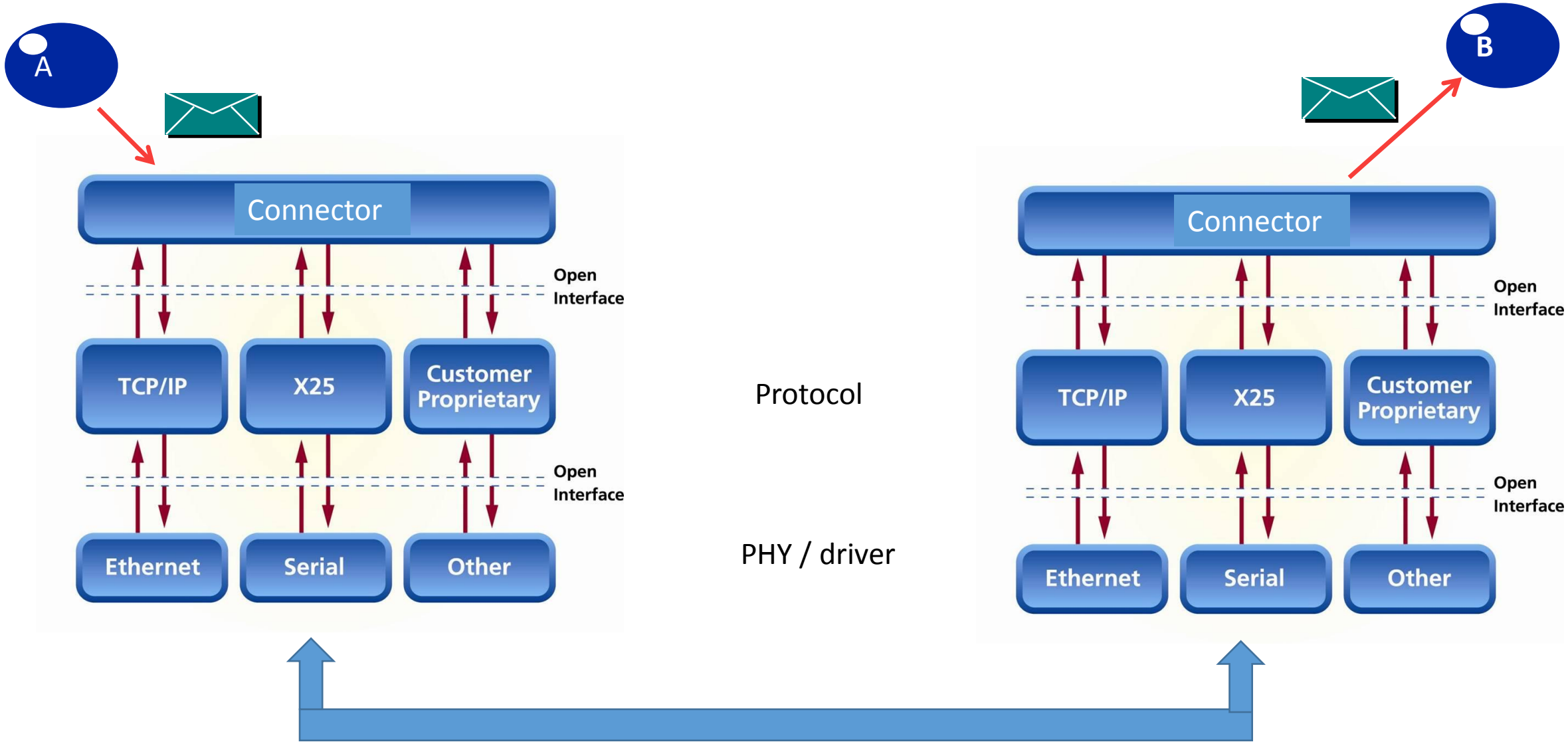


- Message copy between modules
- No message copy in same segment
- System protection (MMU / MPU)

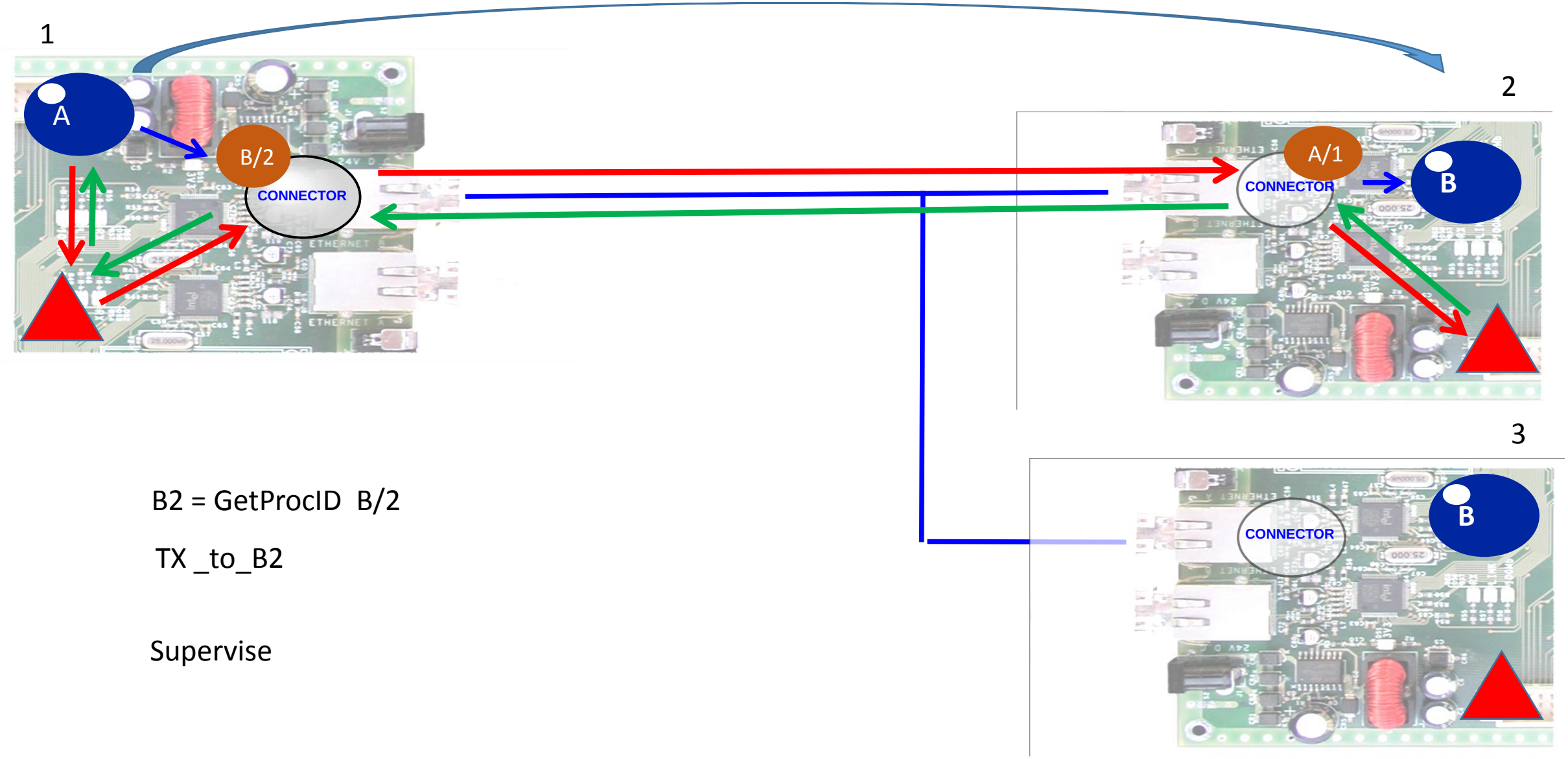
Distributed Systems



Connector


A
B
Connector
Open Interface
TCP/IP
X25
Customer Proprietary
Open Interface
Ethernet
Serial
Other
Protocol
PHY / driver
Connector
Open Interface
TCP/IP
X25
Customer Proprietary
Open Interface
Ethernet
Serial
Other

Distributed systems

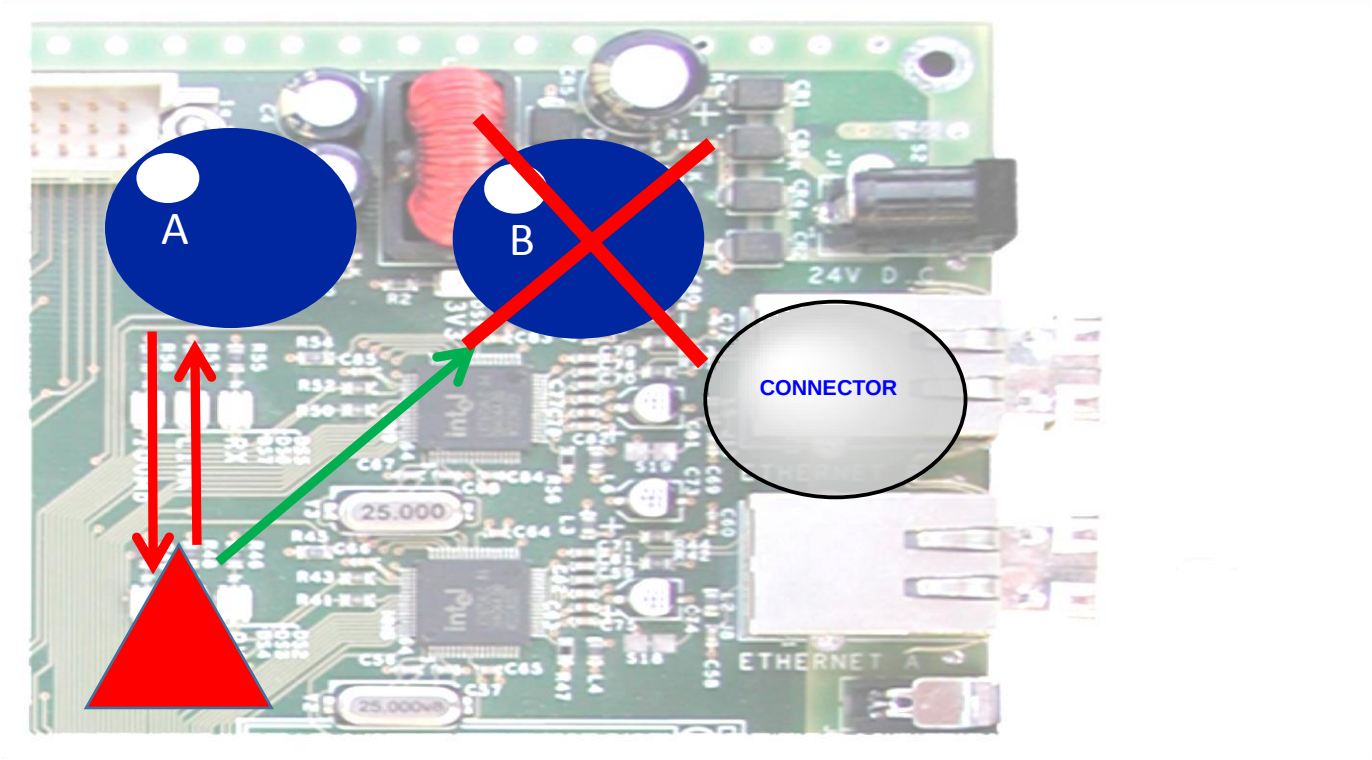


B2 = GetProcID B/2

TX _to_B2

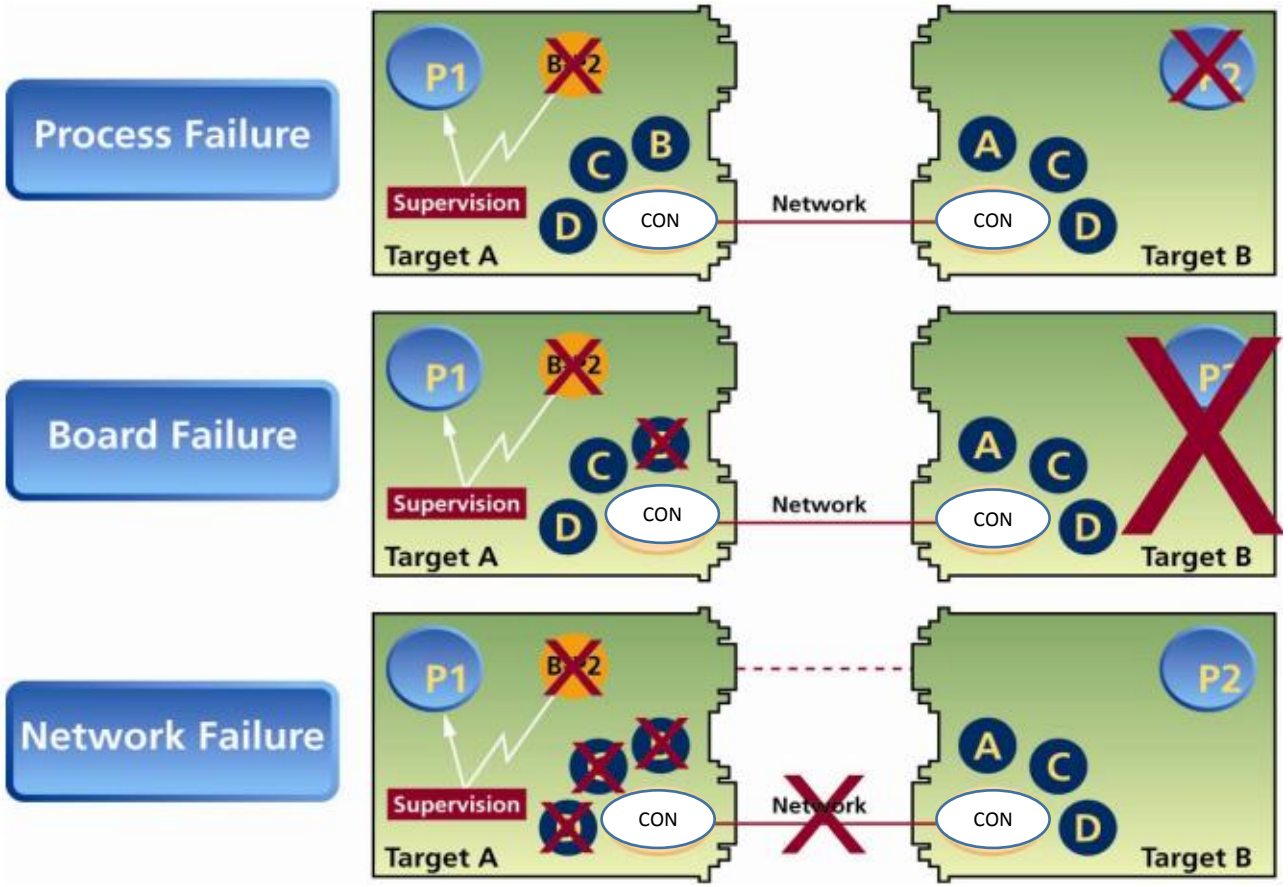
Supervise

Supervision call



Supervise_B

Base for High-Availability



How to select an RTOS ?
Do not get trapped by :

- Pretty pictures (everybody can make them)

  Get informed about the KERNEL & API

- RTOS's that do not fit your application

  One RTOS does NOT fit all !!

- Hard-to-use / learn Operating Systems

  Analyze some code examples

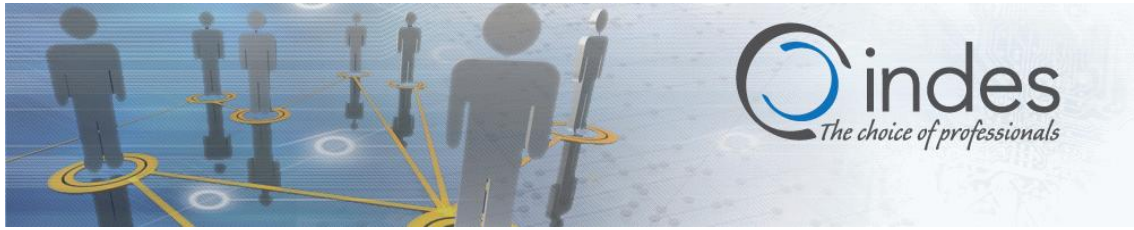
- Training & Introduction

  Get assistance to get through critical introduction phase

For more information :

Gerard Fianen

gerard@indes.com

Tel : 0345 – 545.535

www.indes.com/embedded