Fault-tolerant communication architectures

Claire Pagetti
claire.pagetti@enseeiht.fr

ENSEEIHT - Département Télécommunication et Réseaux
2, rue Camichel, 31000 Toulouse

Course Objectives

- **Introduction to fault-tolerant communication:**
  - What is fault-tolerance?
  - What are the communication failures?
  - Examples
  - Classical means for ensuring fault-tolerance

- **Presentation of TTP**

- **Invited conference on MIL-1553 - spacewire**

Course organisation

- **Lectures 1 : introduction**
- **Lecture 2 : TTP (and TT Ethernet ?)**
- **Lecture 3 : seminar SpaceWire**
- **Lab session 4: modelling TTP with AltaRica**

Notation:

– Exam

Outline

Part I: introduction to fault-tolerance

Part II: TTP

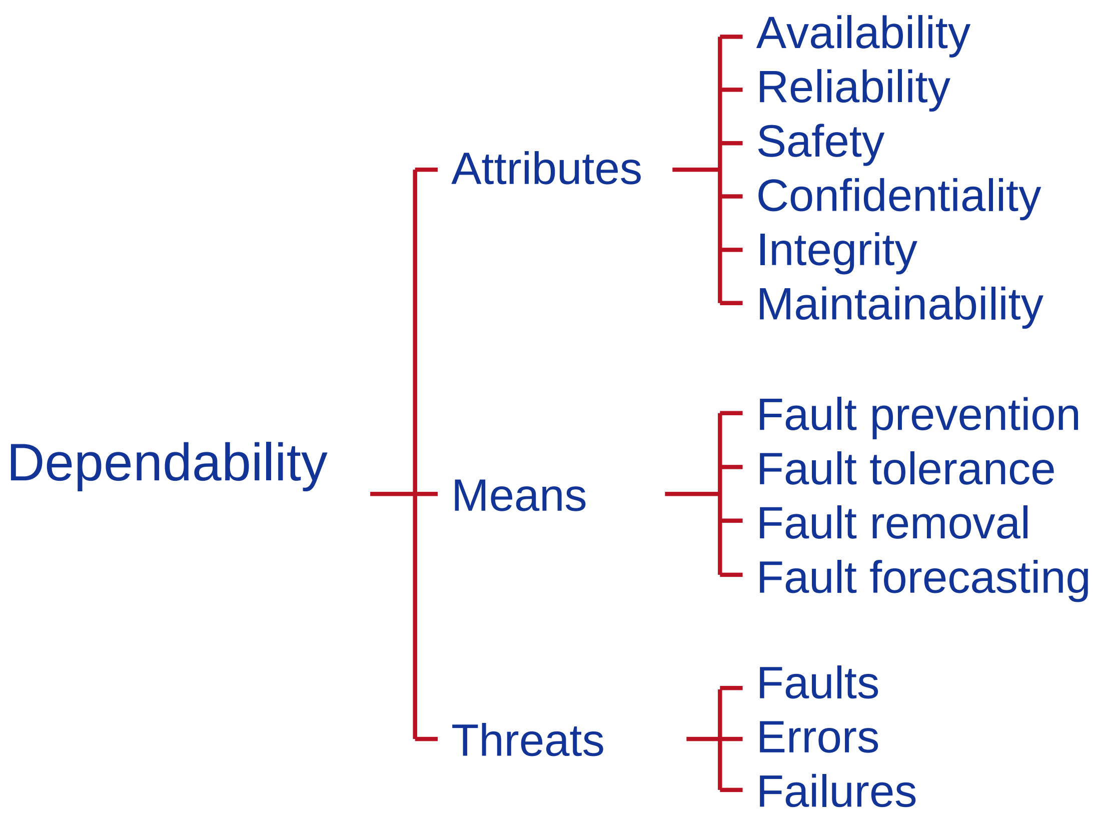
Part IV: SpaceWire

Outline – Part I

1. **First definition**
    1. Fault and failures
    2. Examples
    3. How often will the system fail?

2. **Means for the fault-tolerance**
    1. Error detection
    2. System recovery
    3. Prevention

Dependability [Laprie85]



Fault-tolerance

Fault-tolerance or graceful degradation [Wikipedia]

- is the property that enables a system (often computer-based) to continue operating properly in the event of the failure of (or one or more faults within) some of its components. Fault-tolerance is particularly sought-after in high-availability or life-critical systems.

Threats in the context of communication:

- channel with loss of messages
- failure of emitters
- loss of data coherence
- …

Fault model

When developping a fault-tolerant system, the designer makes assumptions about the types of failures => fault model

“A fault model is nothing more than a statement of how the system is expected to fail.” Philip Koopman

Types of faults:

- **Non faulty The correct message is received at the scheduled time.**
- **Loss The message is never delivered**
- **Erroneous A corrupted message is received and/or the message arrives too late**
  - Benign The message is detected faulty by all receivers:
    - The message is received outside the communication window.
    - The message is corrupted (or not present).
  - Symmetric Same erroneous message sent to all receivers.
  - Asymmetric (Byzantine) The messages received are arbitrary (in time and value).

Fault model

Duration of faults:

- **Permanent once they occur they do not disappear.**
- **Transient appear for a short time and disappear (e.g. upset from electromagnetic interference)**
- **Intermittent appear, disappear, appear…**

Outline – Part I

1. **First definition**
    1. Fault and failures
    2. Examples
    3. How often will the system fail?

2. **Means for the fault-tolerance**
    1. Error detection
    2. System recovery
    3. Prevention

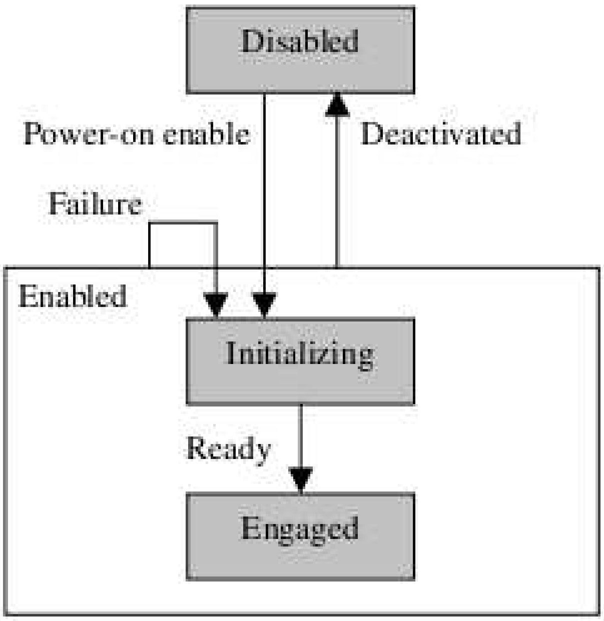
Example of fault model: ROBUS

All bus failures are presumed to be transient.

“The bus is deactivated by cutting off power to the system. When enabled, it executes an initialization routine and then proceeds to

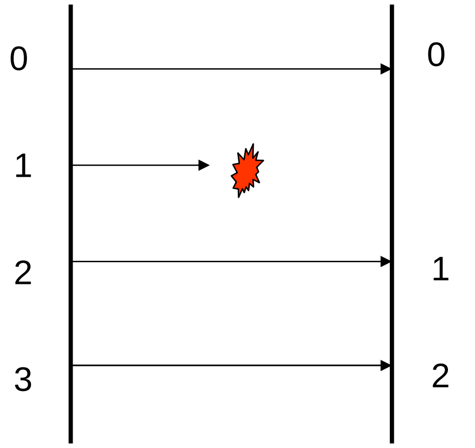
begin service delivery. The bus will remain engaged until it is deactivated or a failure condition is

detected. If a failure occurs, the bus will try to re-establish service delivery as soon as possible.”



Alternating bit protocol

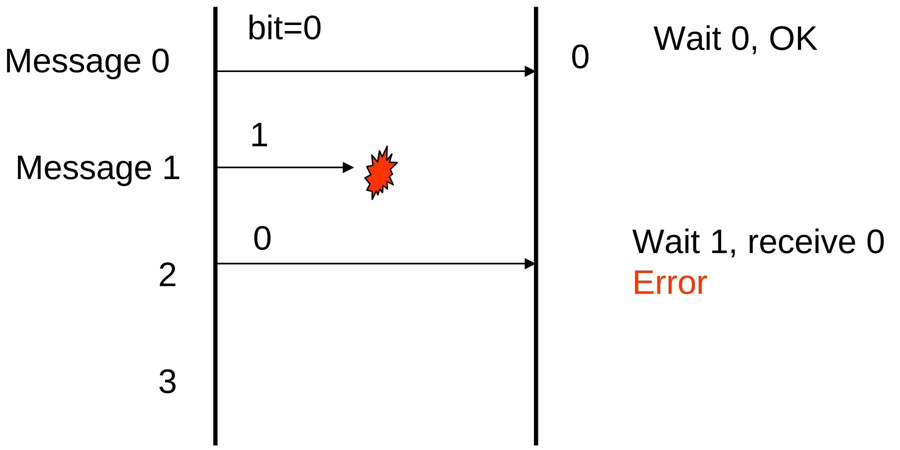
- Bartlett and Scantlebury in 1969
- simple data link layer network protocol that retransmits lost or corrupted messages.



Receiver has lost ordered messages

ABP

Idea: counting the message exchange using a bit

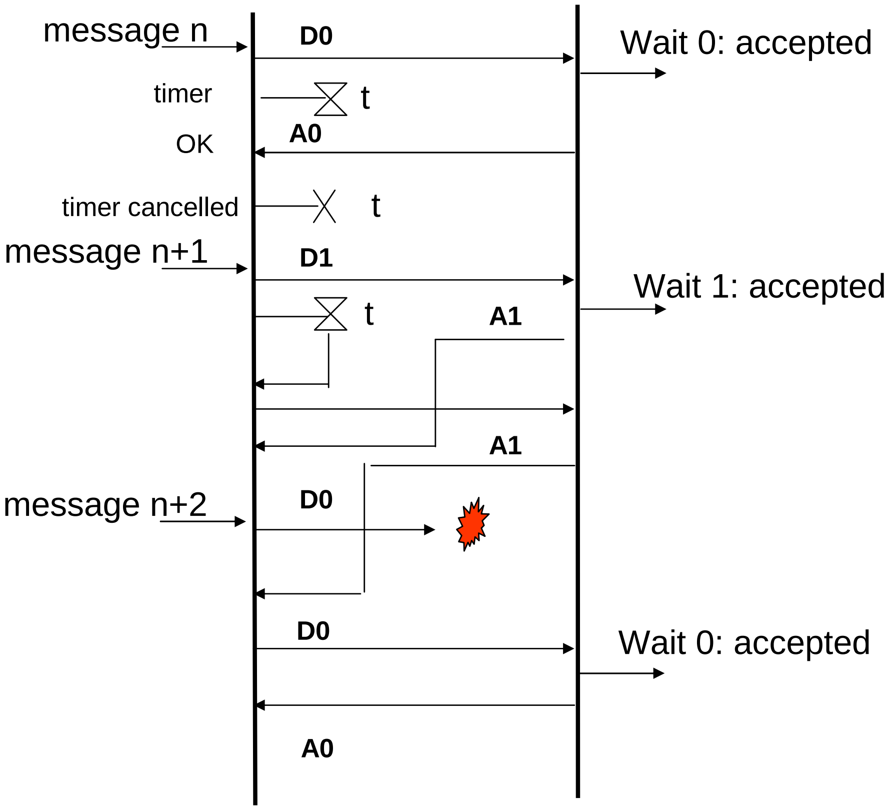


Receiver must make some aknowledgement:

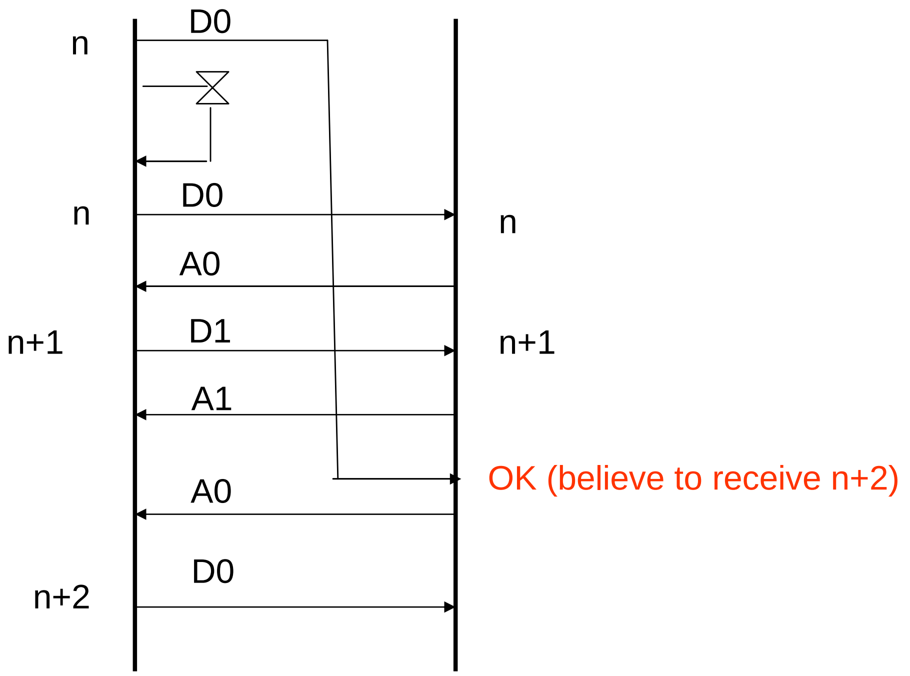
- A NACK1 when message 1 has not been received
- An ACKi for each message

➔ **ACK and NACK can be lost**

ABP: stop and wait protocol



ABP: example of problem



Properties of ABP

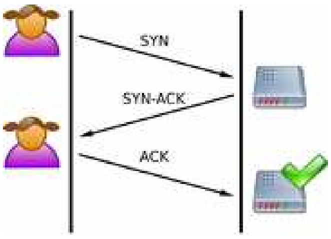
- **Receiver and emitter must be at the same rate**
- **If there is no double overlaping, the ABP respects the data order:**
  - The received messages are in the same order than the emitted messages
- **Self-stabilizing protocol:**
  - ABP can return in a nominal mode after any error (the user may have receive some corrupted data)

Second fault-tolerant protocol: TCP

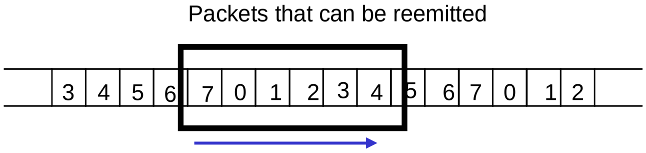
TCP (Transmission Control Protocol)

RFC 793 of IETF

- **Establishment of a connection**
  - based on handshake



- **Data transfer**
  - sliding window
  - sequence number to identify each data
  - cumulative acknowledgment: receiver sends an acknowledgment signifying that it has received all data preceding the acknowledged sequence number



Outline – Part I

1. **First definition**
   1. Definition and examples
   2. Fault and failures
   3. How often will the system fail?

2. **Means for the fault-tolerance**
   1. Error detection
   2. System recovery
   3. Prevention

System failure rate

Philip Koopman

"The fault model I use for message losses is that the intended message does not get delivered within a specified deadline. This deadline takes into account any retries or other corrective action that the lower layers of the protocol may take."

Example: 37 ms message deadline, 10 ms timeout

Number of message transmission attempts:

$$\max attempts = \left[\frac{deadline}{retry\ delay} \right] + 1$$

4 for the example (1 initial + 3 retries)

System failure rate

System failure rate = probability that all attempts fail. Assume that the failure are independent

$$P_{system\_failure} = \left[P_{message\_fail} \right]^{\max\_attempts}$$

If 4 messages lost causes a system failure, if the message failure rate is 1% and the message rate is 200 msg/s, how many times will occur a system failure by year?

$$NB_{system\_failure\_per\_year} = \left[P_{message\_fail} \right]^{\max\_attempts} \times \left[\frac{messages}{\sec ond} \right] \times \left[\frac{31557600\, s}{year} \right]$$

Message failure rate

How to compute $P_{message\_fail}$

Two main sources of loss:

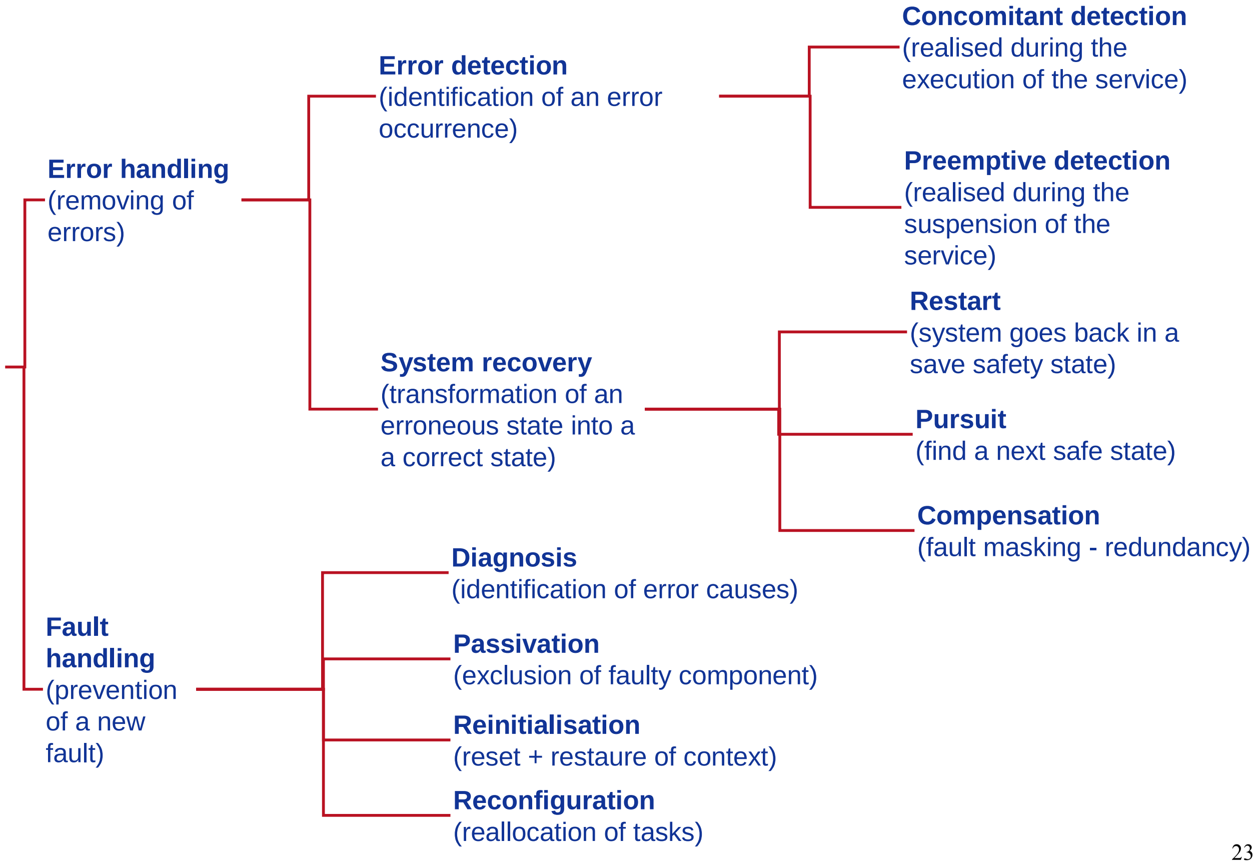
- Bit errors due to noise
- Collisions of message

Outline – Part I

1. **First definition**
    1. Fault and failures
    2. Examples
    3. How often will the system fail?

2. **Means for the fault-tolerance**
    1. Error detection
    2. System recovery
    3. Prevention

Fault-tolerance methods tree [Laprie90]



Error handling (removing of errors)

Error detection (identification of an error occurrence)

Concomitant detection (realised during the execution of the service)

Preemptive detection (realised during the suspension of the service)

System recovery (transformation of an erroneous state into a a correct state)

Restart (system goes back in a save safety state)

Pursuit (find a next safe state)

Compensation (fault masking - redundancy)

Fault handling (prevention of a new fault)

Diagnosis (identification of error causes)

Passivation (exclusion of faulty component)

Reinitialisation (reset + restaure of context)

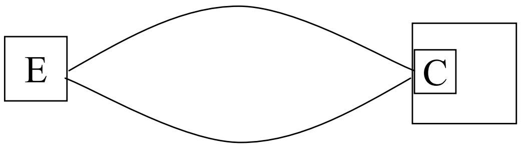
Reconfiguration (reallocation of tasks)

Error detection

Ex: ABP - error handling: concomitant detection + restart

Main error detection techniques:

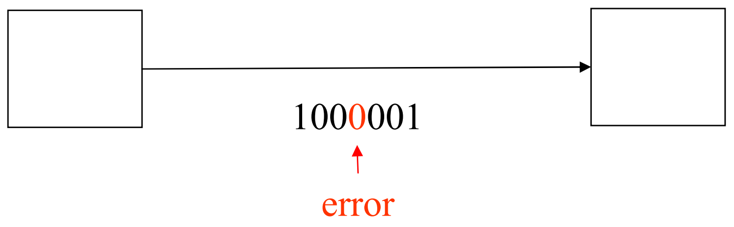
- **Doubling and comparison**



- Doubling of emission
- Consolidation at reception

- **Temporal control**
  - A *watchdog timer* is a computer hardware timing device that triggers a system reset if the main program, due to some fault condition, neglects to regularly service the watchdog. The intention is to bring the system back into normal operation.
- **Wrapping**
  - The wrappers intercept all method invocations on interfaces of the wrapped sub-system / service instance.
- **Error-detecting code**
  - Error detection is the ability to detect the presence of errors caused by noise or other impairments during transmission from the transmitter to the receiver.
  - Error correction is the additional ability to reconstruct the original, error-free data.

Error-detecting code: problem



- **How to detect the occurrence of an error?**
- **How to localise the error?**
- **How to correct the error?**

Naive solution: repetition

Detection:

- Initial message is duplicated
- 10010011001001 is sent instead of 1001001
- Receiver detects a problem if the duplicata are not identical

Correction

- Triplication of initial message
- 100100110010011001001 is sent instead of 1001001
- Correct message is the one which appears twice

Necessity of redundancy:

- Adding redundancy helps in detecting and correcting errors in transmission (and in storage)

Non corrected errors:

- Give examples of not detected, not corrected, badly corrected errors

Parity bit (or Vertical Redundancy Check)

Idea: add a bit that encodes the number of 1s in the message

- 0 if the number is even
- 1 if it is odd

Coding	0	1	1	0	0	1	1	0
Detected error	0	1	1	0	0	1	1	1
Undetected error	0	1	1	0	0	1	0	1

Cyclic redundancy check: polynomial approach

Procedure:

- To each sequence of bits, we associate an equivalent polynomial
  - 1001001 ⇔ $X^6 + X^3 + 1$
  - $u_0.u_1.u_2..u_n \Leftrightarrow P(X) = u_0 + u_1 .X + u_2 .X^2 + ... + u_n .X^n$
- Agreed generating polynomial
  - g(X)

Coding:

- Compute $P'(X) = X^k.P(X)$ (highest power in g(x)) and Q, R such that $P'(X) = Q(X).g(X) + R(X)$
- Send $P'(X) + R(X)$
- Warning: computation are made on Z/2Z
  - $1+1=0$; $X+X=0$; $X=-X$

Example: 1101

- $g(x) = X^3 + X + 1$
- $P(x) = X^3 + X^2 + 1$
- $P'(X) = P(X).X^3 = X^6 + X^5 + X^3$

Cyclic redundancy check: polynomial approach

Example: 1101

$$
\begin{array}{l|l}
X^6 + X^5 + X^3 & X^3 + X + 1 \\
\underline{X^6 + X^4 + X^3} & \overline{X^3 + X^2 + X + 1} \\
X^5 - X^4 & \\
\underline{X^5 + X^3 + X^2} & \\
X^4 + X^3 + X^2 & \\
\underline{X^4 + X^2 + X} & \\
X^3 + X & \\
\underline{X^3 + X + 1} & \\
1 &
\end{array}
$$

➔ R(X) = 1

➔ Coding 1101001

Cyclic redundancy check: polynomial approach

Decoding:

- – Received message M(X)
- – Divide M(X) by g(X)
  - If the rest is 0, no error detected
  - Otherwise an error is detected

Quality of protection depends on the generating polynomial

- – If g(x) has at least 2 terms then detection of 1 bit error
- – If g(x) has an irreductible factor of 3 terms, detection of double bit errors
- – If g(x) is a multiple of x+1 then detection of odd bit errors.

Outline – Part I

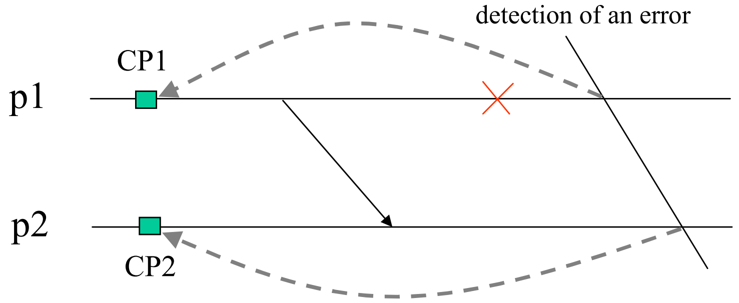
1. **First definition**
   1. Fault and failures
   2. Examples
   3. How often will the system fail?

2. **Means for the fault-tolerance**
   1. Error detection
   2. System recovery
   3. Prevention

Restart – rollback

Idea:

- Multiple processes that communicate via message passing in a distributed system.
- Detect an error
- Revert the system state back to some earlier, correct version, for example using checkpointing,
- Resume the execution from there.



Checkpoints for rollback

Checkpoints

- snapshot of state saved by process to a stable storage
- must be coherent (same context, same view of the system …)
- bad choice can lead to several roll-backs (domino effect) until the initial state.

Strategies:

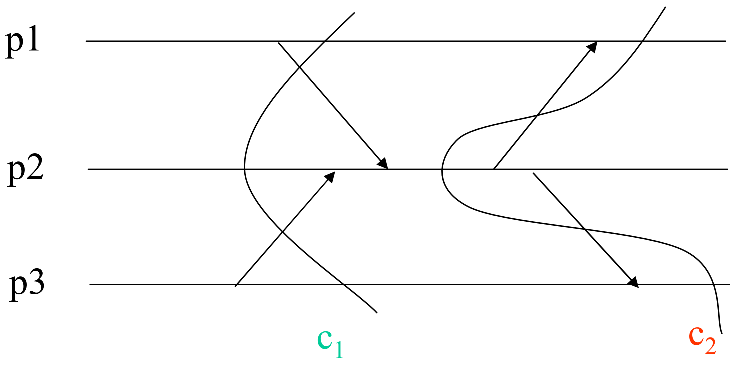
- Consistent checkpointing (synchronous): definition off-line of checkpoints in order to constitute a global coherent state. Synchronisation is required and a unique checkpoint for each process is sufficient
  - Usually require a higher overhead in control messages
  - Less concurrency in process execution can be achieved
- Independent checkpointing (asynchronous): no collaboration between processes on taking checkpoints. The system tries to reconstitute a coherent global state.
  - Need of several checkpoints (eventually all the checkpoints that have been taken since program initialisation)
  - Domino effect

Example

- **Crash recovery with little overhead. T. Juang and S. Venkatesan in 11th International Conference on Distributed Computing Systems, 1991, pp 454-461.**

Consistent state:

- Set of processors states in which each pair of processors agrees on the communication that has taken place between them (c_1 consistent and c_2 unconsistent)



Example

Principle:

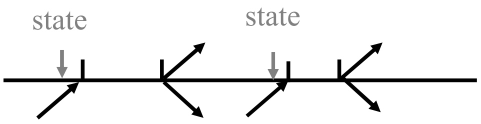
- detection of orphan processes by counting the number of sent and received messages

Hypothesis:

- reliable channels and applications are piloted by events (emission, treatment, reception)

Algorithm:

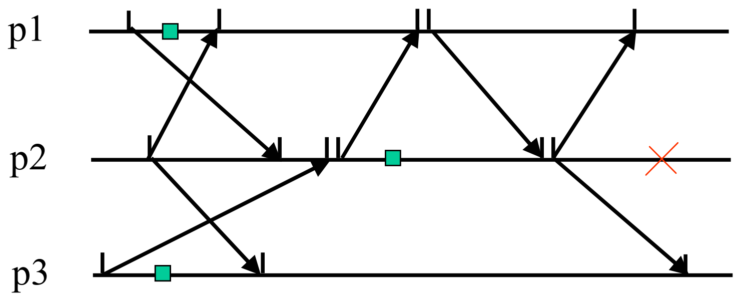
- each process saves in a volatile memory each event encoded by the tuple *(state, received message, sent messages)*, this constitutes a checkpoint in case another process is failed
- regularly, the event is stored in permanent memory, this constitutes a checkpoint in case the process is failed



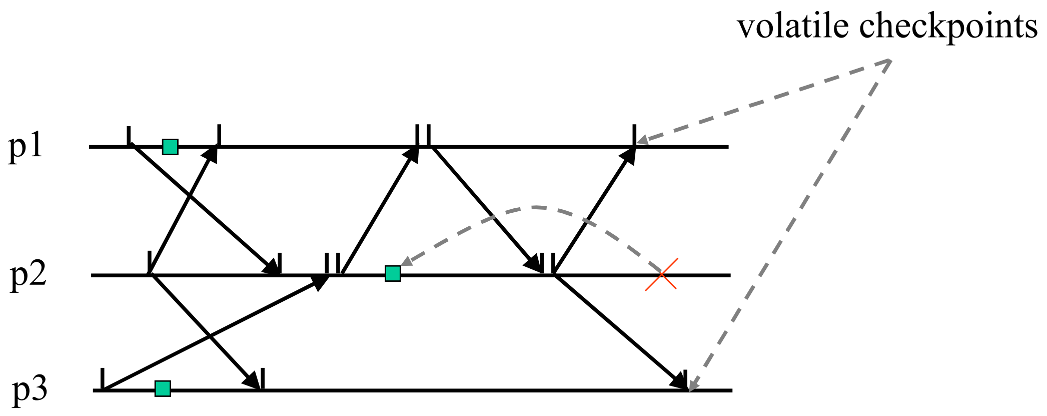
Example

- Let $RCV_{i \leftarrow j}$ denote the number of received messages by i from j stored in the checkpoint Cpi
- Let $SENT_{j \rightarrow i}$ denote the number of messages sent by j to i stored in the checkpoint Cpj
- A set of checkpoints is valid if:

$$\forall i, j, RCV_{i \leftarrow j} \leq SENT_{j \rightarrow i}$$



Example



- Message from p2 to inform that there is a rollback
- Verification that checkpoints are valid
- Replay of the last received message (which was stored)
- Reemission of messages
- Duplicated messages are ignored by other process

Exercise:

- give event memorised by each process and give the scenario played after resume

Pursuit – roll-forward

Idea:

- Detect an error
- Find a next correct state
- Move forward the system in this new state

Examples:

- Reset of the system with a new acquisition of sensors
- Exceptions in a programming language
- Reconstitution of a lost packet from the neighbours packets

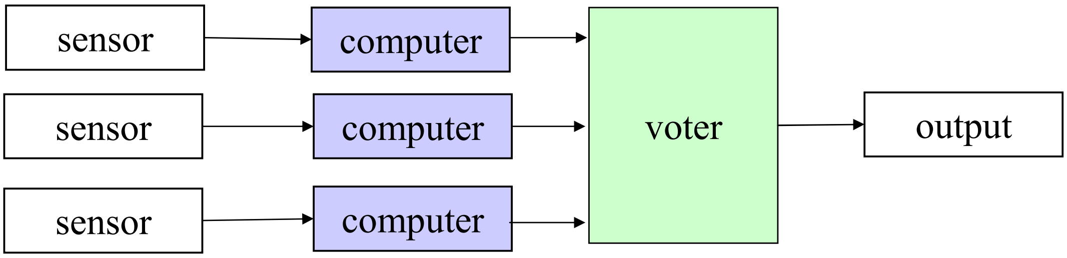
Fault masking

Preventing an error from propagating to a system output.

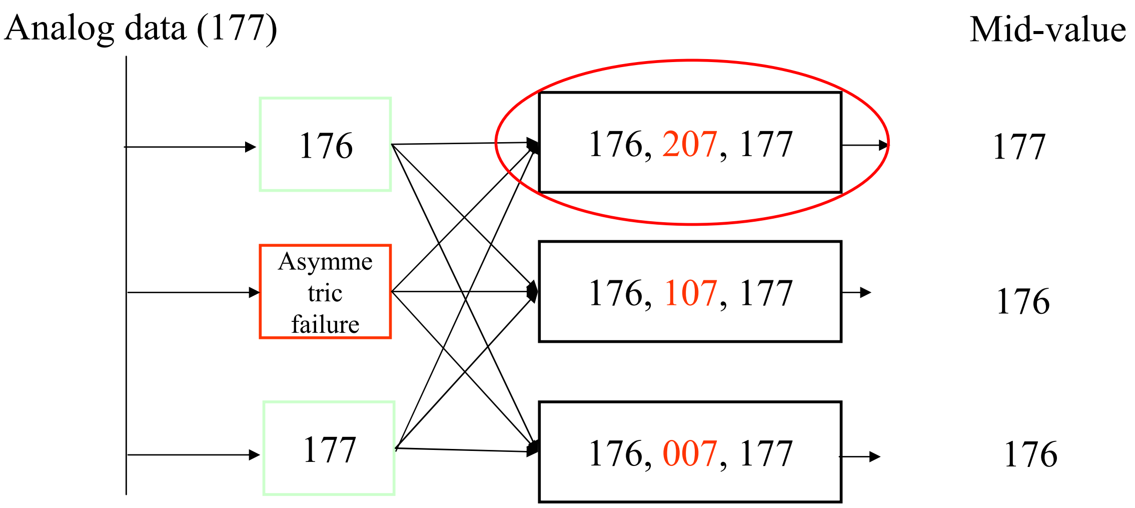
Redundant architectures (using voting for instance).

Voting:

1. Exact match: all replicates are assumed to identical
2. Fault-tolerant average: all replicates are assumed to be close to each other. The medial value is relevant if there is at most 1 failure
3. Mechanical: it can be performed at the actuators



Example



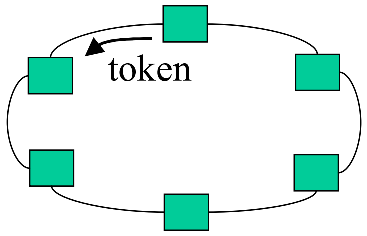
Will look faulty in future majority votes

Outline – Part I

1. **First definition**
   1. Fault and failures
   2. Examples
   3. How often will the system fail?

2. **Means for the fault-tolerance**
   1. Error detection
   2. System recovery
   3. Prevention: Media access protocols to avoid collision

Token ring

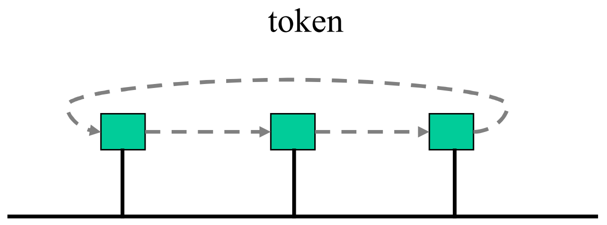


Principle:

- Token turns on the ring
- A station receives the token. If it wants to emit, it replaces the token by a message
- The message has the address of the destination
- The message is broadcasted and goes back to the emitter
- The emitter releases the token

Example: FDDI (Fiber Distributed Data Interface) uses dual counter-rotating rings

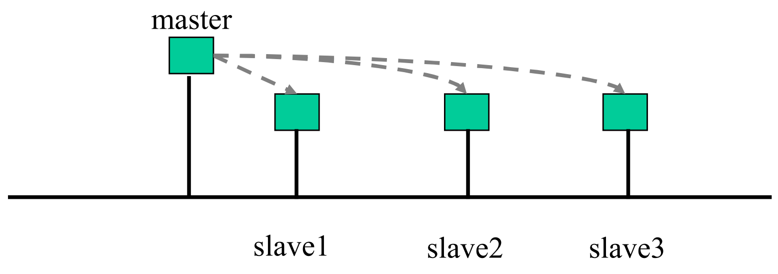
Token bus



– Logical ring for the token
– Broadcast of messages
– Limited transmission time when a processor has the token

Example: ARCnet (Attached Resource Computer Network)

Polling



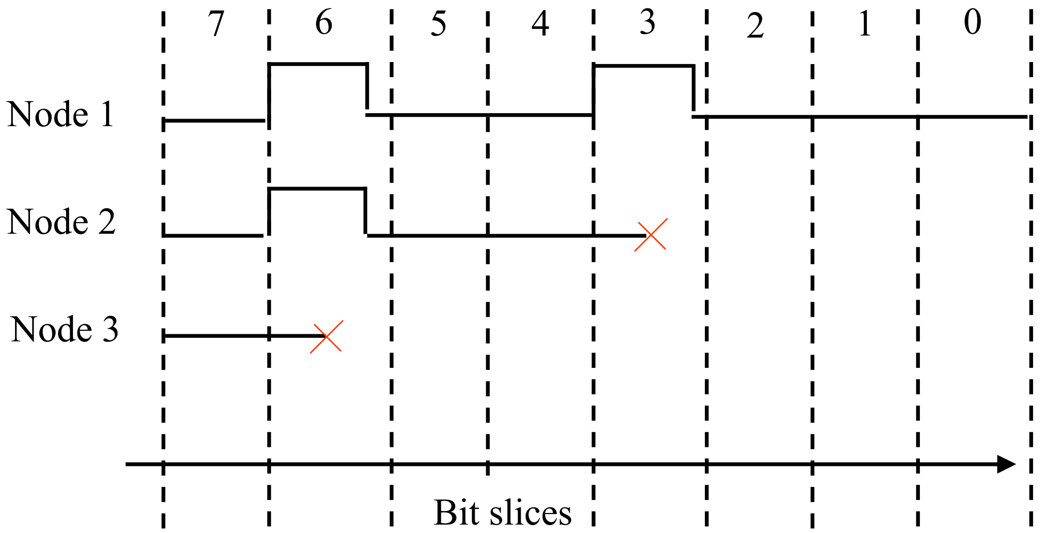
– Centrally assigned master periodically polls (by sending a polling message) slave nodes, giving thme explicit permission to transmit
– Extensions with multiple master

Examples: 1553, profibus

Binary countdown (Bit dominance protocol)

Principle:

- Transmission medium has an overriding value (say « 1 »)
- Node broadcasts its message
- During a transmission, a node drops out of the competition if it detects a predominant signal

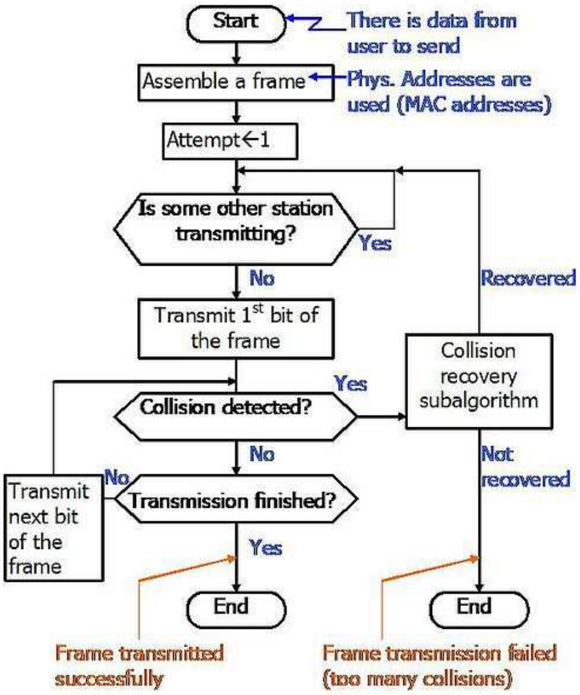


CSMA/CD

Carrier Sense Multiple Access /Collision Detection

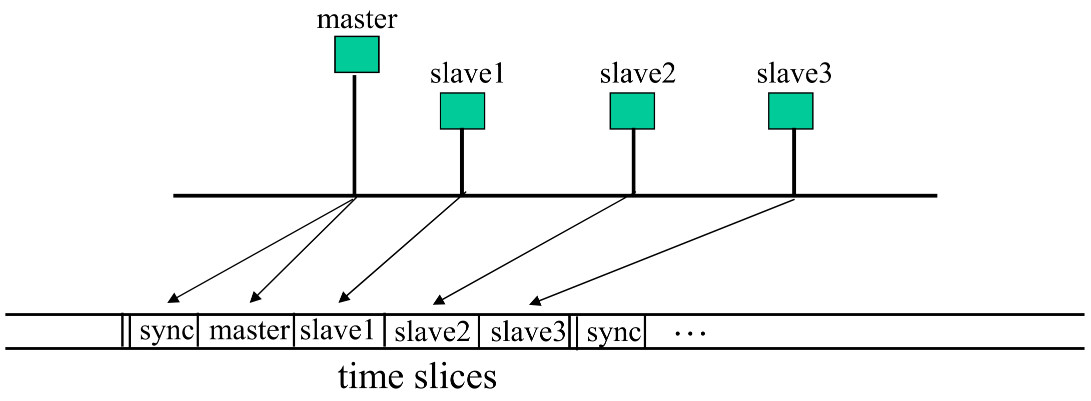
– a carrier sensing scheme is used

– a transmitting data station that detects another signal while transmitting a frame, stops transmitting that frame, transmits a jam signal, and then waits for a random time interval before trying to send that frame again.


Start
There is data from user to send
Assemble a frame
Phys. Addresses are used (MAC addresses)
Attempt←1
Is some other station transmitting?
Yes
No
Transmit 1st bit of the frame
Collision detected?
Yes
No
Collision recovery subalgorithm
Recovered
Not recovered
Transmission finished?
No
Yes
Transmit next bit of the frame
End
End
Frame transmitted successfully
Frame transmission failed (too many collisions)

Example: Ethernet

TDMA (Time Division Multiple Access)



- Time access is decomposed into slices
- Each frame starts with a sync message sent by the master to synchronise slaves
- Each slave starts a countdown timer that expires at the beginning of its slice
- Each node sends its messages during its slice

Example: Datac (Digital autonomous terminal access communication), TTP, Flexray

References

- **Wikipedia**
- **LIS sous la direction de J.-C. Laprie, Guide de la sûreté de fonctionnement, Cépaduès, Toulouse, Mai 1995, 369 p. (ISBN 9782854283822)**
- **Course L. Logrippo** www.uqo.ca/luigi
- **Lost Messages and System Failures. Philip J. Koopman, Carnegie Mellon University in Embedded Systems Programming, 9(11), October 1996, pp. 38-52**
- **Communication protocol for embedded systems. P. Upendar and P. J. Koopman. Embedded Systems Programming, 7(11):46–58, November 1994.** http://www.ece.cmu.edu/~koopman/protsrvy/protsrvy.html
- **P. J. Koopman. « Time Division Multiple Access Without a Bus Master ». Technical Report RR-9500470, United Techologies Research Center, June 1995.**

References

- Middleware Architecturewith Patterns and Frameworks. Sacha Krakowiak. Chapter 11. 2008. http://proton.inrialpes.fr/~krakowia/MW-Book/Chapters/FaulTol/faultol-body.html#tth_chAp11
- Course on network. Rushed Kanawati http://www-gtr.iutv.univ-paris13.fr/Cours/MatOld/Reseaux1/cours/chapitre2

Outline

Part I - introduction to fault-tolerance

Part II - TTP/C: fault-tolerant real-time bus and protocol for embedded systems

Part III: SpaceWire

Outline – Part II

1. **Application domain**
   1. Generalities
   2. Needs

2. **TTP/C**
   1. Presentation
   2. TDMA functioning
   3. Clock synchronisation
   4. Membership mechanism
   5. Fault-tolerance

X-by-wire applications

- **Domain « chassis » and « engine control »**

- **Mechanical connection replaced by numerical connection between components of the system**

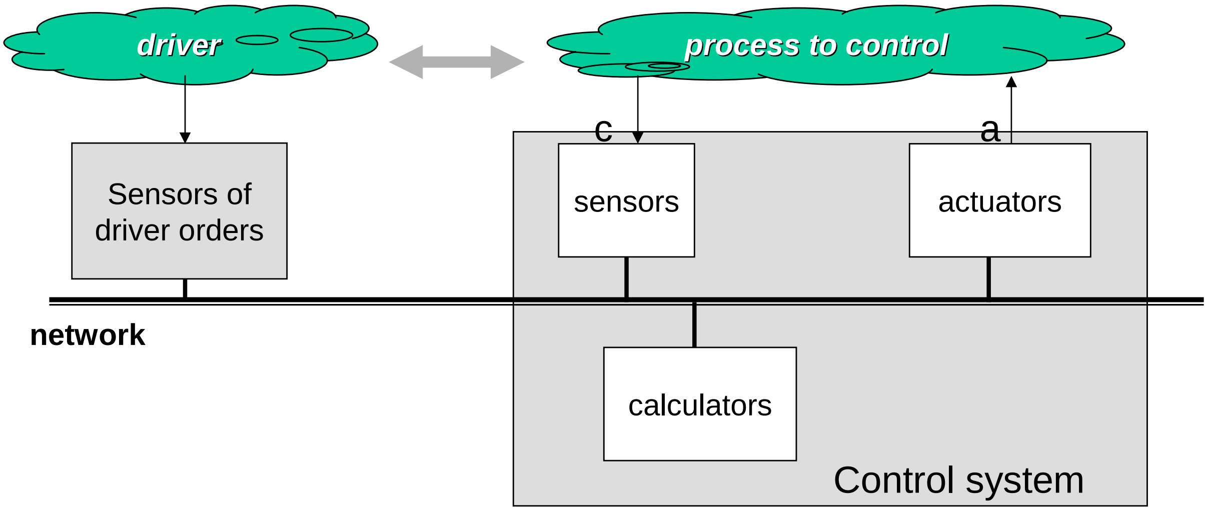
- **Advantages**
  - Reduction of weight, space …
  - Reduction of maintenance costs …
  - System evolutivity
  - Software integration to increase the comfort in driving, drive assistance (safety city Volvo), …

- **Problem**

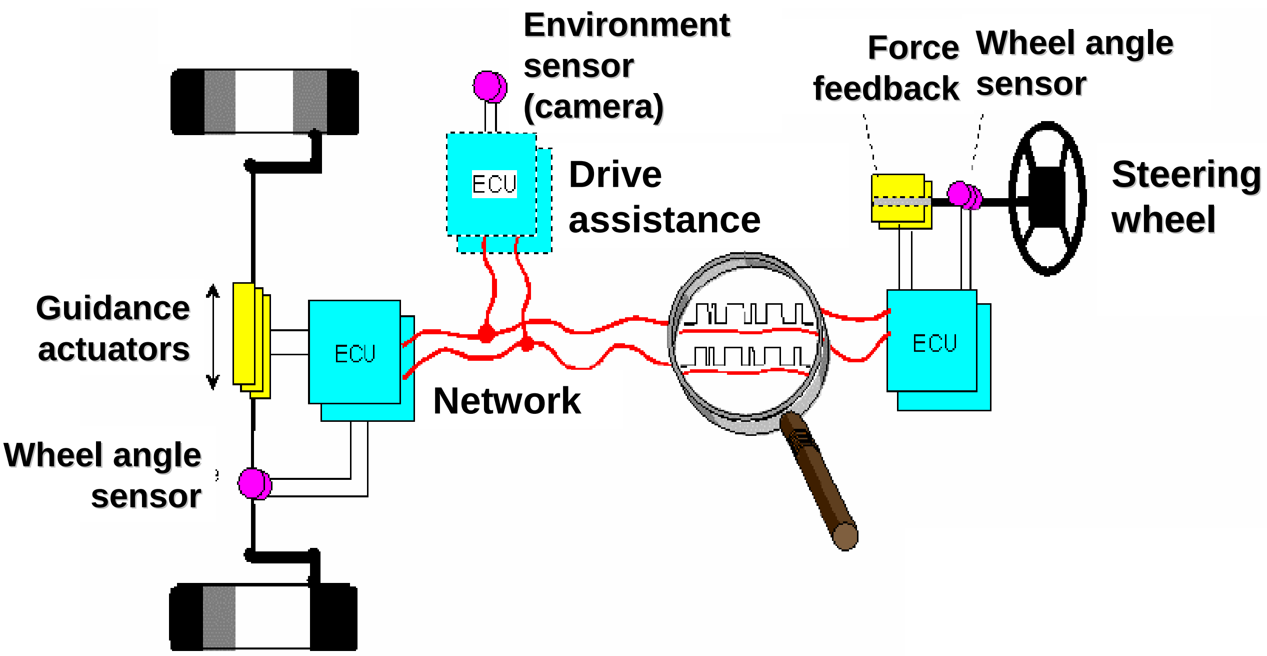
  dependability

Example of architecture

Use of a share network for sensors / actuators / treatment



Second example



Needs

In terms of communication

- Determinism, robustness, fault-tolerance (detection, monitoring, fault masking)
- Compositionality (integration of several functions in a same ECU or a set of ECUs)

=> Conception of architectures and applications« driven by the time »

Outline – Part II

1. **Application domain**
    1. Generalities
    2. Needs

2. **TTP/C**
    1. Presentation
    2. TDMA functioning
    3. Clock synchronisation
    4. Membership mechanism
    5. Fault-tolerance

History of (automotive) TDMA network

TTP/C

- First publication in 1994. Since then, lots of studies, proofs, tests.
- Hermann Kopetz – University of Vienna, Austria (patent in 1997)
- Products : TTTech (www.tttech.com/)

FlexRay

- Automotive industrial consortium (1999 – BMW & Co)
- More flexibility
- Some proofs to be done on the protocol to validate it

TTEthernet

- time-triggered technology for the IEEE 802.3 (Ethernet) standard, high performance and TT determinism combined

General features

TTP for « Time Triggered Protocol »

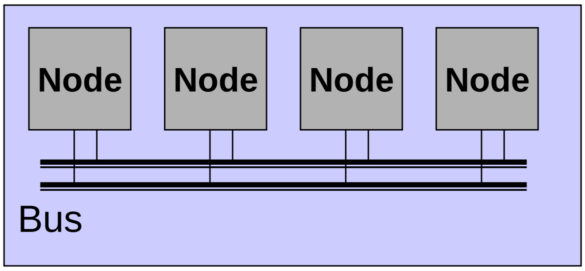
C for hard real-time applications « C class »

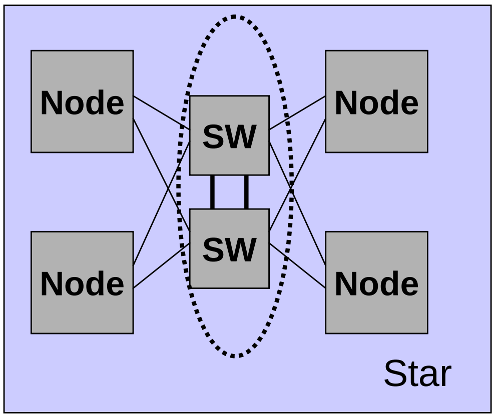
- **TDMA medium access**
  - Collision free
  - Determinism
  - TDMA round € [1ms, 10ms]
  - Cyclic scheduling of messages
- **Bandwidth : 500 Kbps to 5 Mbps**
  - 5ns for each meter
- **Fault-tolerance integrated in the protocol**

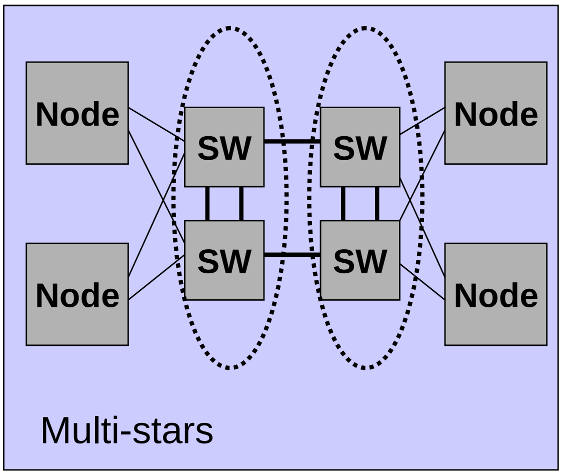
Industrial use

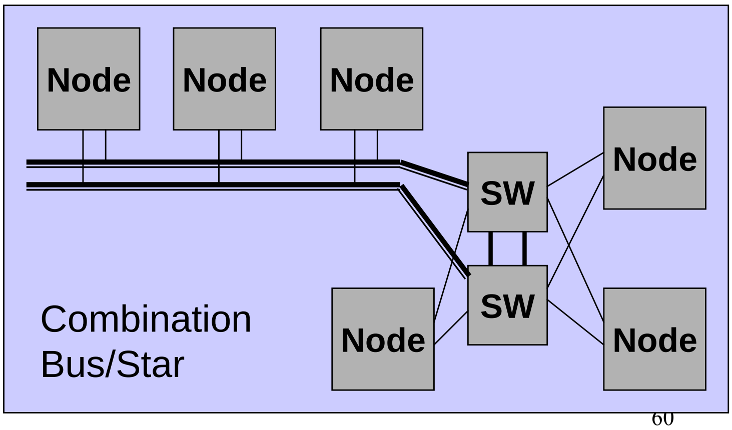
- A380: cabin pressure system
- Military aircraft: engine command
- B787 electrical system
- Bombardier: flight control system
- Embraer

Several topologies

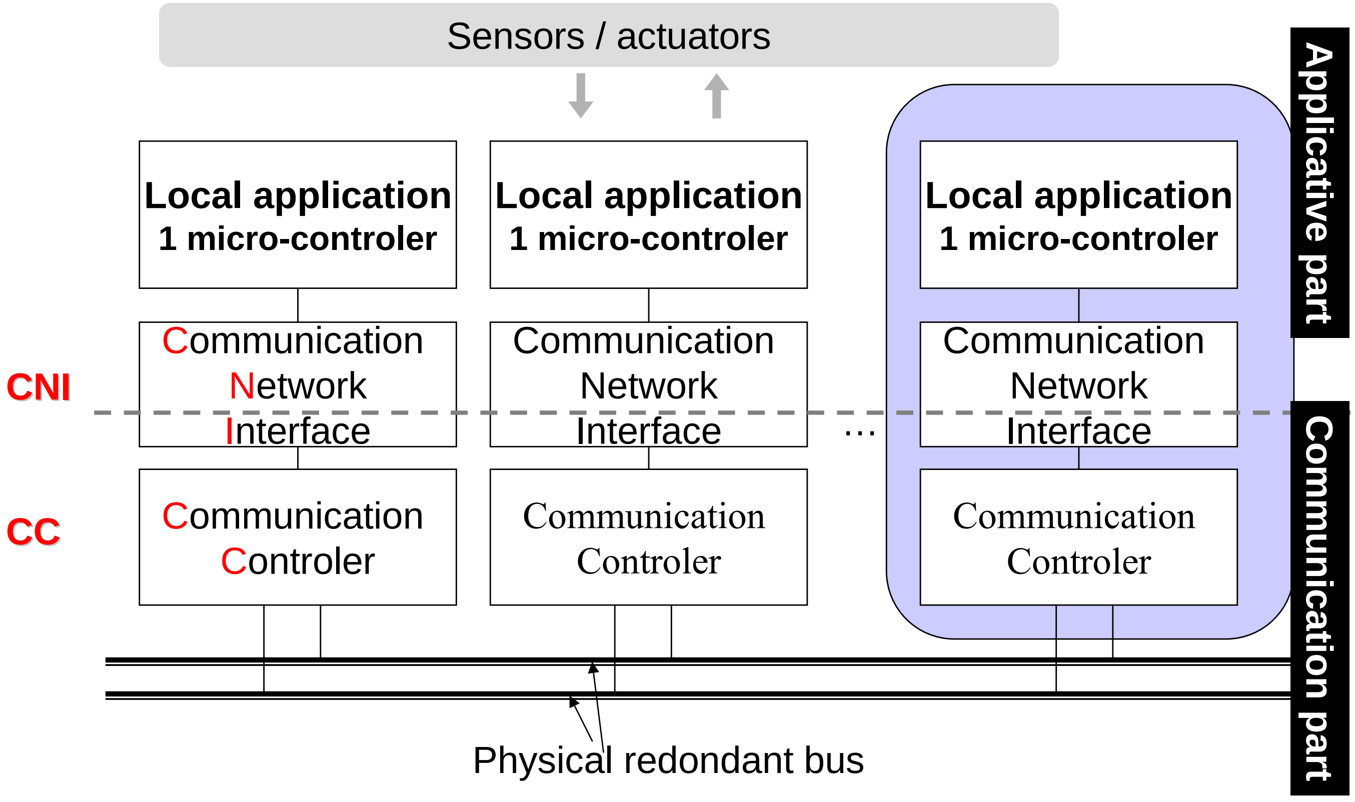




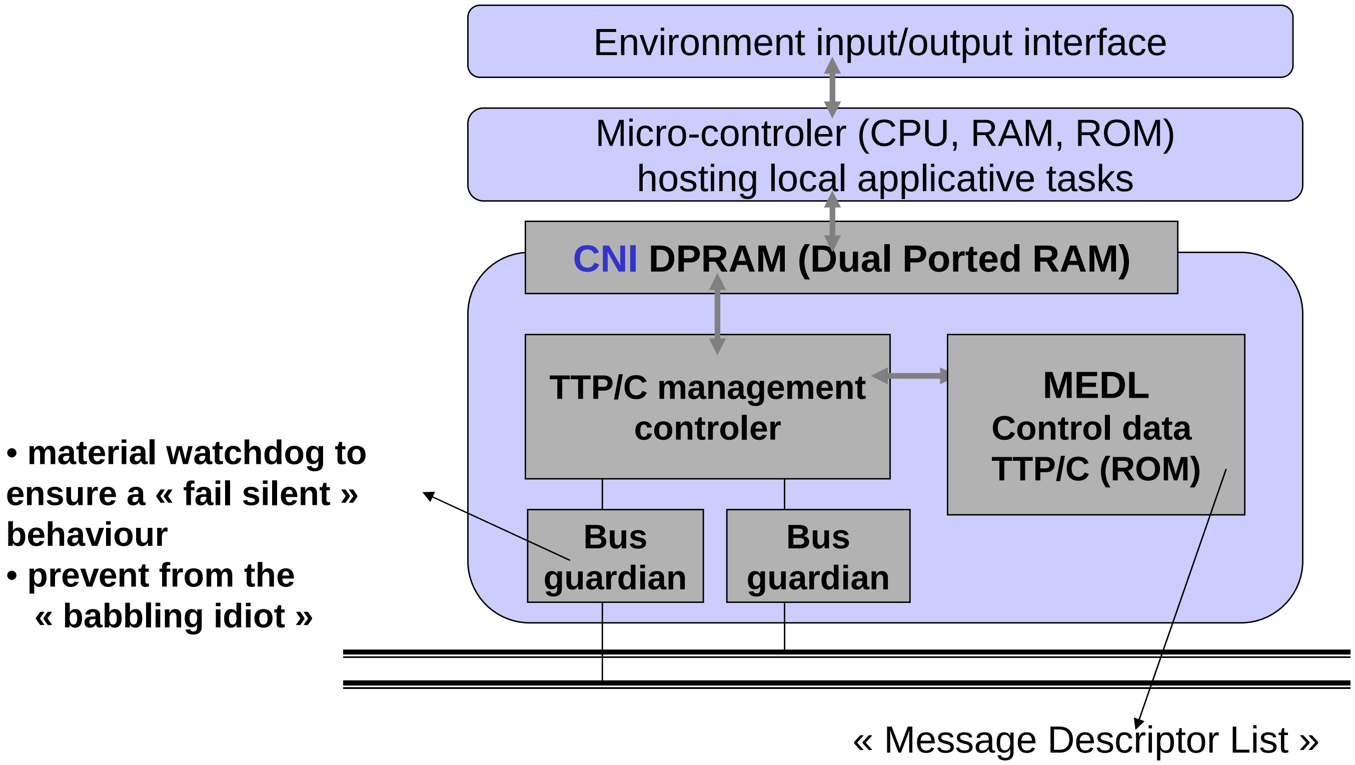




Structure of a TTP network



Structure of a TTP node


Environment input/output interface
Micro-controler (CPU, RAM, ROM)
hosting local applicative tasks
CNI DPRAM (Dual Ported RAM)
TTP/C management
controler
MEDL
Control data
TTP/C (ROM)
Bus
guardian
Bus
guardian

- material watchdog to ensure a « fail silent » behaviour
- prevent from the « babbling idiot »

« Message Descriptor List »

Fault-tolerance in TTP

Based on 4 mechanisms :

1. TDMA

2. Clock synchronisation
   → tolerant to a failure of the clocks

3. Mechanism of Membership (common vision of safe nodes)
   → Implicit acknowledgment
   → Helps a node to detect if it is seen as safe or not by the others

4. Mechanism of clique avoidance
   → Allows to avoid situations where few « crazy » can agree to see each other as safe

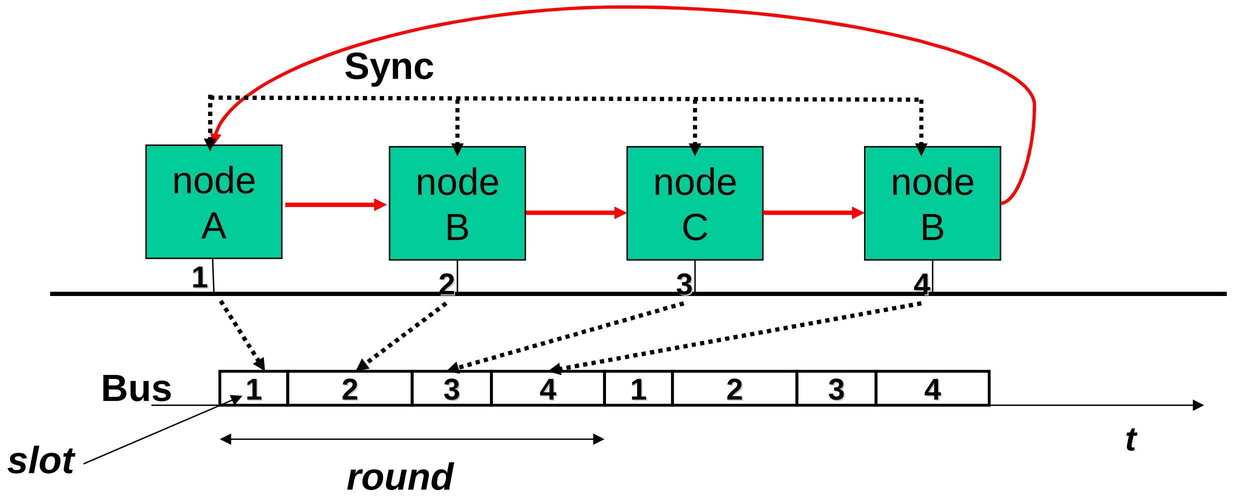
Outline – Part II

1. **Application domain**
   1. Generalities
   2. Needs

2. **TTP/C**
   1. Presentation
   2. TDMA functioning
   3. Clock synchronisation
   4. Membership mechanism
   5. Fault-tolerance

Principles of TDMA

- A « slot » is a time interval, a station can emit a message during its slice
- A TDMA « round » corresponds to a sequence of slots such that each station emits exactly once



TTP/C TDMA

A node may need to transmit several messages

- but 1 slot in each « round »

→ In a TDMA round

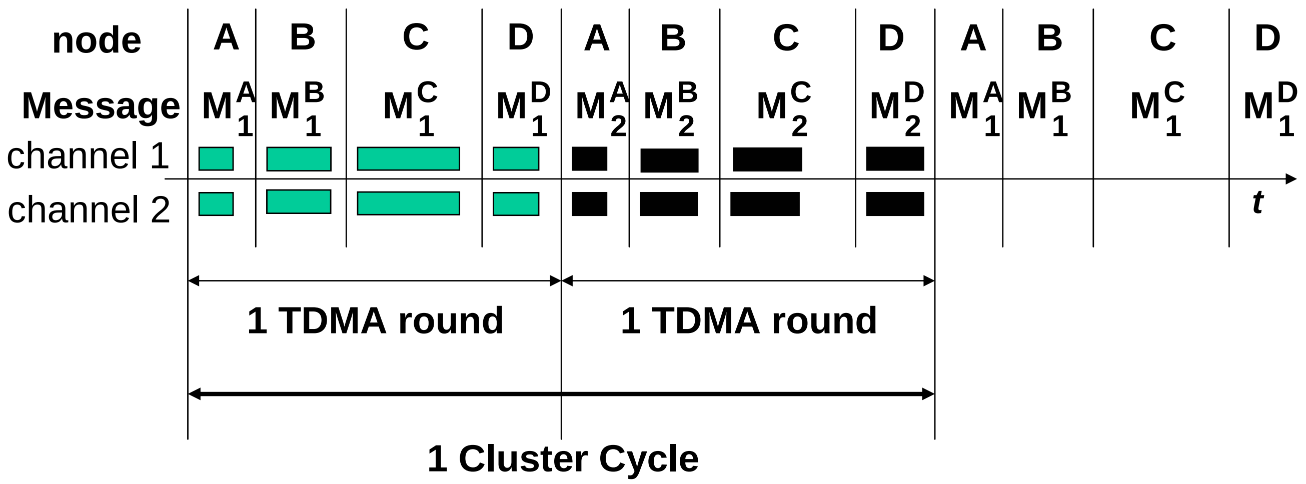
- Each node emits one message in its slot (on each bus)
- The round stops when all nodes have sent a message

Several TDMA « rounds » which are different in terms of messages can be defined but order and size remain unchanged

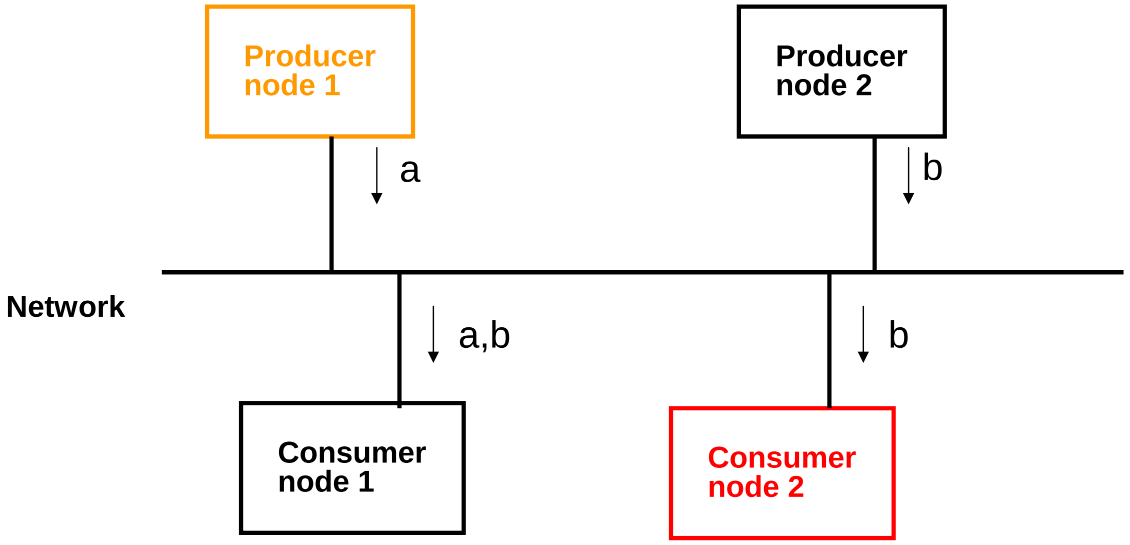
→ A « Cluster Cycle » is the sequence of all TDMA « rounds » and is repeated indefinitely

A « Cluster Cycle » is defined for each mode

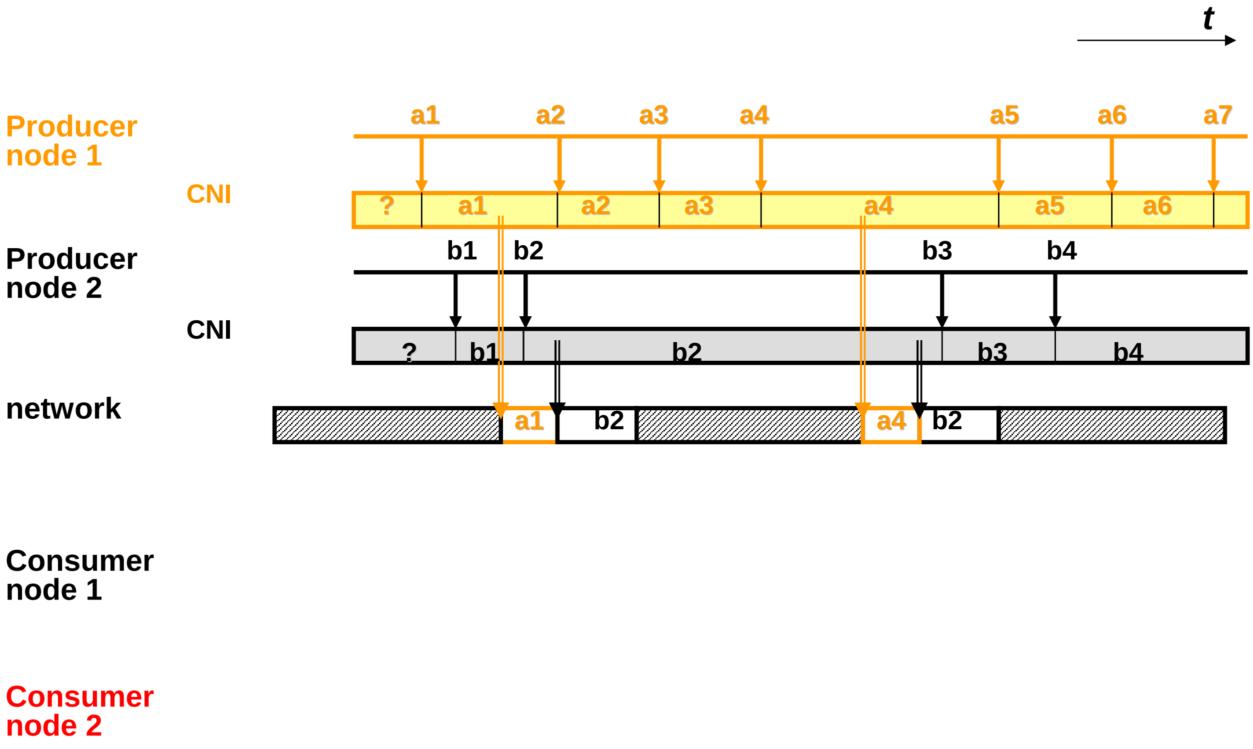
TTP/C TDMA



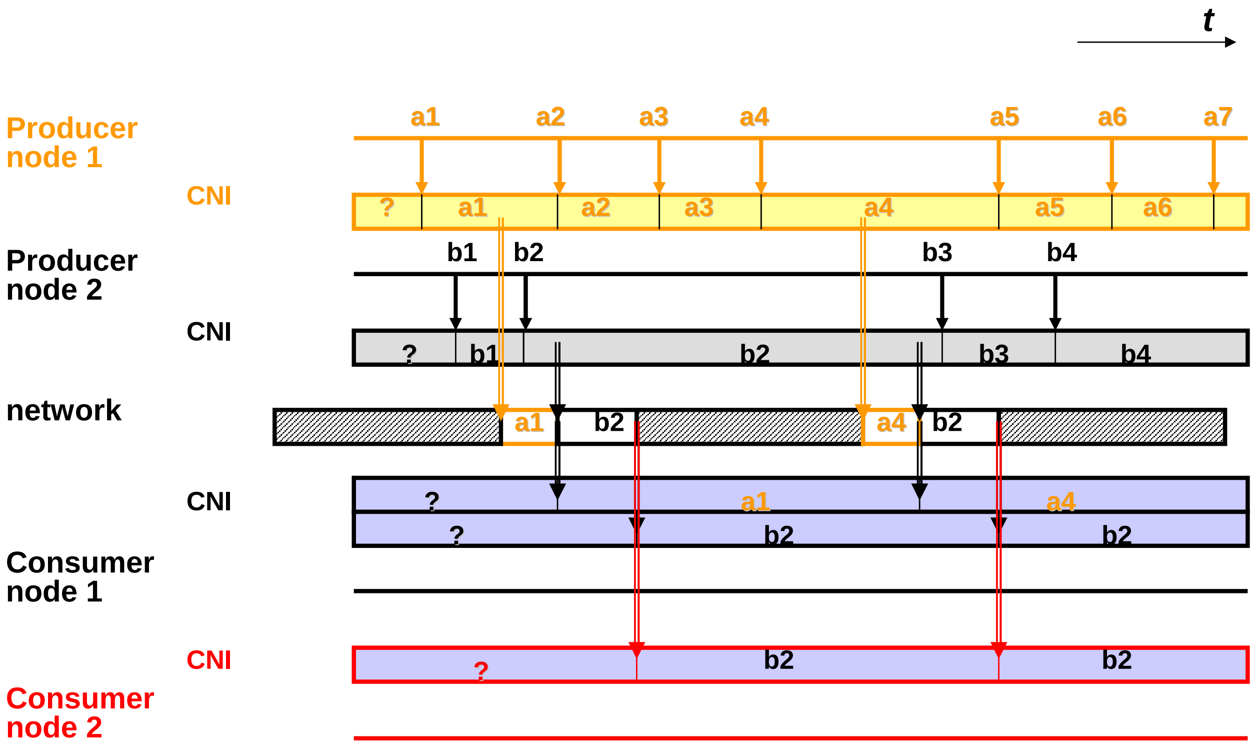
TTP functioning



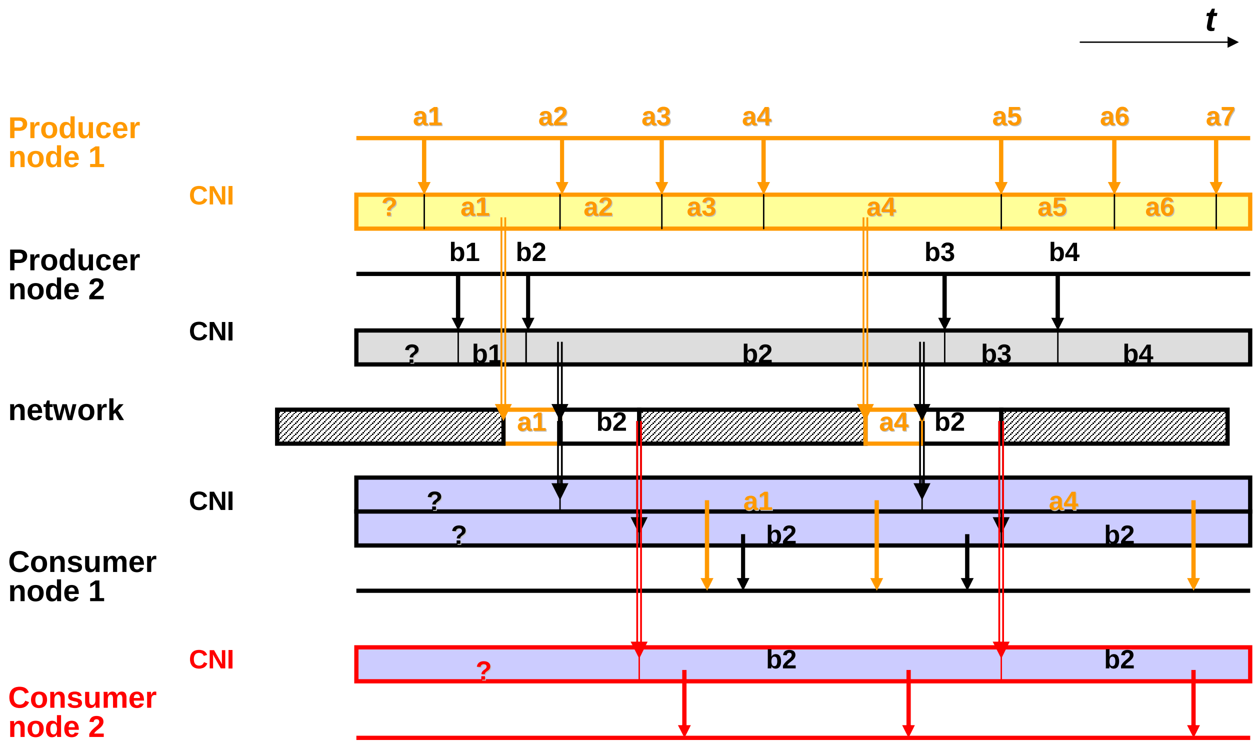
TTP functioning



TTP functioning



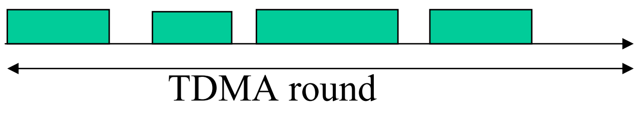
TTP functioning



MEDL (Message Descriptor List)

- **Each node knows the statical scheduling of all the messages of the « Cluster Cycle » and this for each mode**
- **contains following information:**
  - emission
    - time to send
    - the address in the CNI were data is located
  - reception
    - time to receive
    - the address were to store information in the CNI
  - additional information for the protocol
- **In a given mode, in a given instant, in a given « Cluster Cycle » corresponds a unique node and message**
- **No need of arbiter for the medium access**

MEDL (Message Descriptor List)



	time	address	I/O	length	type	additional
1						
2						
3						
4						

Bus guardian

Guaranty that a processor only emits in the correct slot

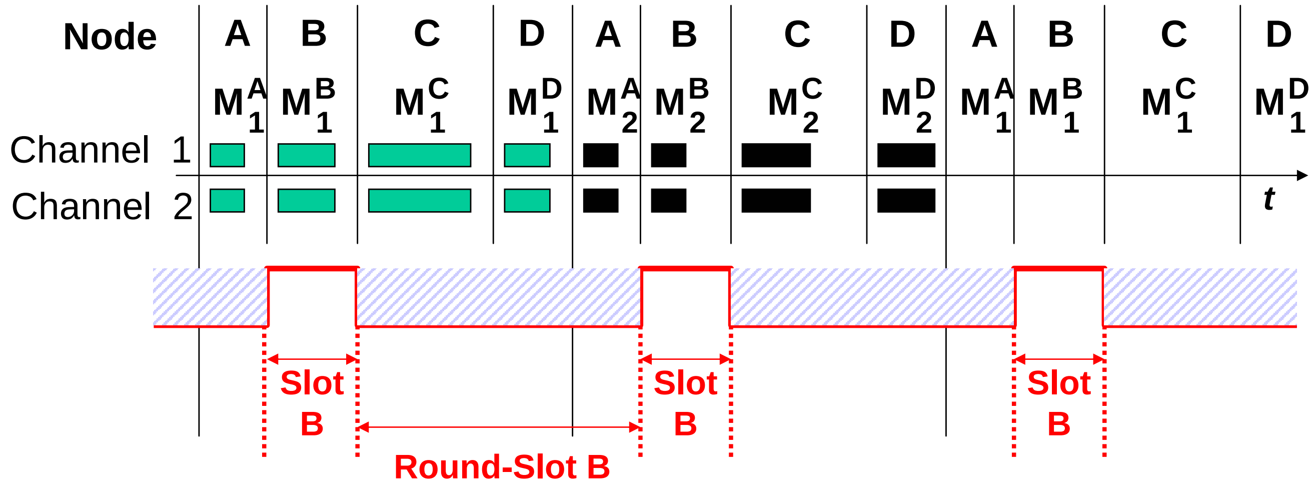
- Protection against desynchronised stations
- Protection against babbling stations (« babbling idiot »)

For this, the bus guardian must :

- have its own clock
- not be too close (physically) of the processor to avoid common failure mode
- have its own power supply

Bus guardian

Give access to the bus only during the instant specified in the MEDL



Outline – Part II

1. **Application domain**
   1. Generalities
   2. Needs

2. **TTP/C**
   1. Presentation
   2. TDMA functioning
   3. Clock synchronisation
   4. Membership mechanism
   5. Fault-tolerance

Clock synchronisation

Idea :

- Adaptation of Welch-Lynch algorithm
- Use of an average of clocks
- No additional traffic to synchronise clock
- Use of knowledge of messages (in the MEDL)

Welch-Lynch algorithm

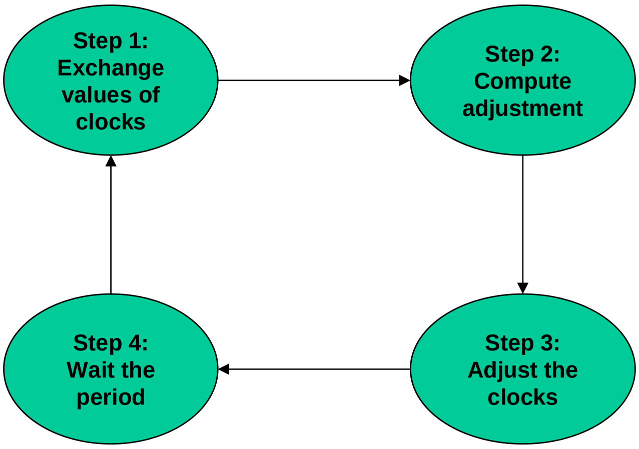
Let us consider n connected processes with at most k failed such that $3k < n$

The channels are assumed to be reliable and the transmission delay is assumed to be bound in time

Each process is equipped with a physical clock PCp(t) (which tends to derivate) and a virtual clock VCp(t) which is the value taking into account for the events

The clocks of the 2 processes p and q are synchronised at instant t if there are enough closed $|VCp(t)-VCq(t)| < \gamma$

Welch-Lynch algorithm principles


Step 1: Exchange values of clocks
Step 2: Compute adjustment
Step 3: Adjust the clocks
Step 4: Wait the period

Welch-Lynch algorithm

Input:

n := number of processes

k := maximal number of fault with n > 3k

(1) Sort the clocks (C1..Cn) from the smaller to the bigger

(2) Remove the k smallest values and the k bigger ones

(3) Compute the average value on the median clocks with the formula

$$cnf([C_1,\ldots,C_n]) = \frac{C_{f+1} + C_{n-f}}{2}$$

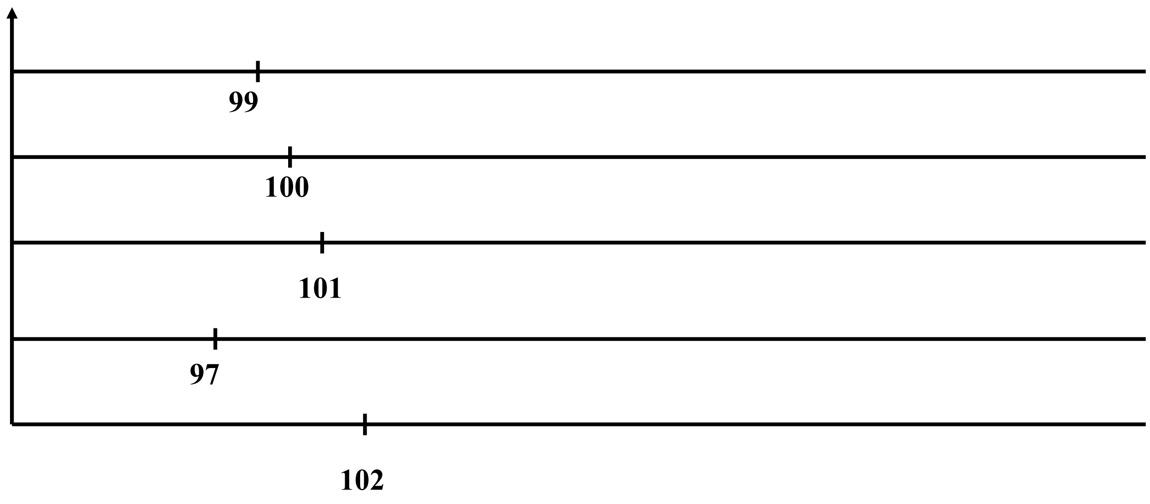
Welch-Lynch algorithm

```
T := P; CORRp := 0;
repeat forever
    wait until VCp = T;
    broadcast SYNC;
    wait for δ time units;
    ADJp := T + d – cfn(ARRp);
    CORRp := CORRp + ADJp;
     T := T + P;
end of loop.

on reception of SYNC message from q do ARRp[q] := VCp.

VCp = PCp + CORRp;
```

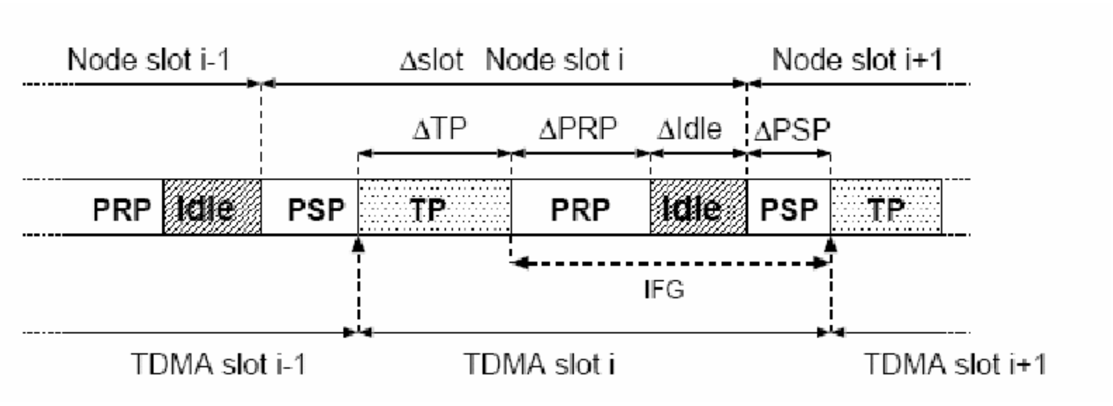
Welch-Lynch algorithm

Apply the algorithm on the five nodes:

- n = 5, P = 100, d = 1, δ = 10



Slot decomposition



1. **PSP – *Pre-Send Phase:* Emitter frame construction**
2. **TP - *Transmission Phase:* Effective emission**
3. **PRP - *Post-Receive Phase:* Receiver reception and status verification**
4. **Idle Phase: Possibility of reconfiguration**

TTP synchronisation

Principle :

- Application of the previous algorithm with k=1
- Synchronisation on 4 particular clocks in the Membership
- At each message reception, each node memorises the local data of the producer (reception date stored in the MEDL)
- At each synchronisation instant, each node reconstructs the average of the reception date
  - If the absolute difference between the local clock and the average is greater than $\pi/2$, the node detects that it is incorrect and disconnects itself
  - Otherwise, it updates its clock

Example

Node	Slot	emission
N1	[0,20]	15
N2	[20,45]	40
N3	[45,75]	70
N4	[75,100]	95

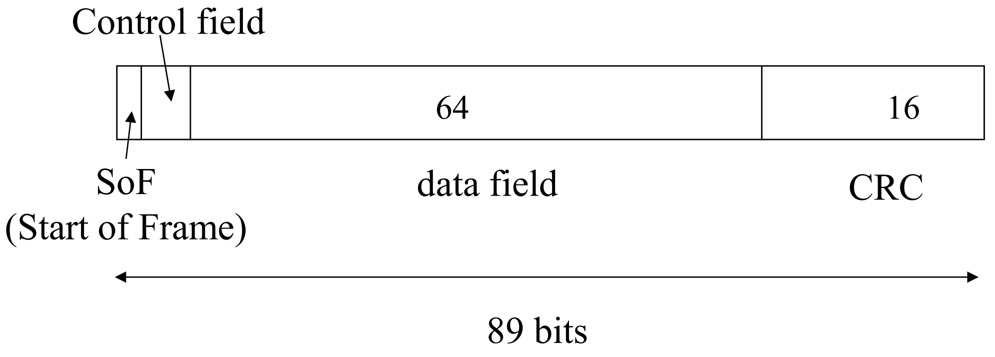
Time	PC(N1)	PC(N2)	PC(N3)	PC(N4)
15	16	15	15	14
40	39	43	42	40
70	70	71	69	70
95	94	95	94	96

Outline – Part II

1. **Application domain**
   1. Generalities
   2. Needs

2. **TTP/C**
   1. Presentation
   2. TDMA functioning
   3. Clock synchronisation
   4. Membership mechanism
   5. Fault-tolerance

Format of a TTP frame



Each node emits during its slot a particular data structure named C-State which contains:

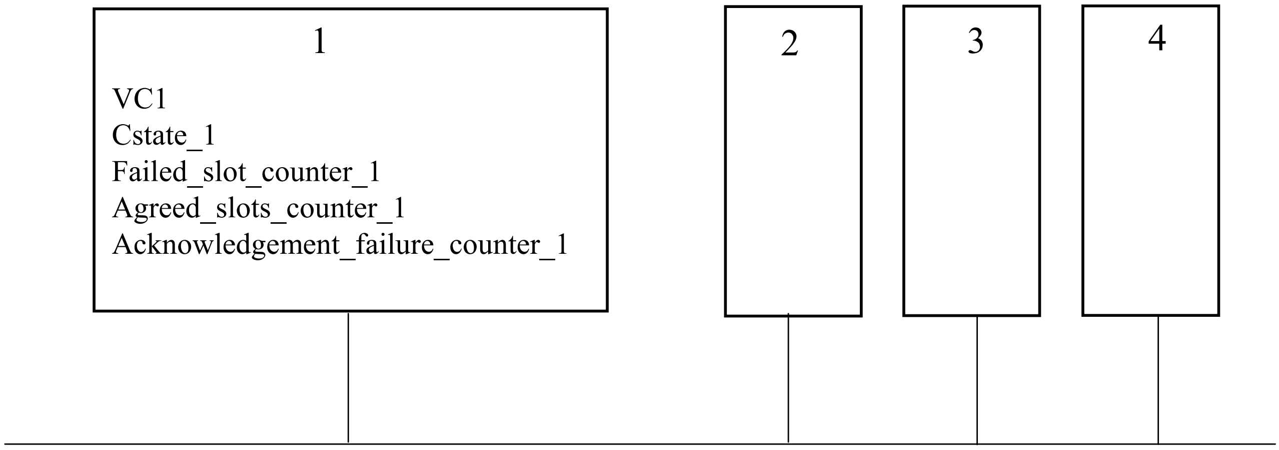
- The number of the slot attached to the node in this « TTDMA round »
- An eventual demand of mode change in the next « Cluster Cycle »
- A local vector of « **Membership** »
- **Vision of the node off all nodes in the cluster (alive or not)**

Membership mechanism

- **Membership mechanism is based on an implicit acknowledgment**
- **An acknowledgment is determined by the node N, emitting in slot i, after reception and treatment of messages from the two next successors of N in the TDMA round**

- **The acknowledgment process consists in comparing the membership vectors of N and of the successors (appearing in the message)**
  - Only valid message (emitted in the correct slot by an alive node) are analysed

Variables used in the algorithm



Each node stores its local variables

Execution of a TDMA round

Initially:

- VC1 = PC1
- Cstate_1: {ok,ok,ok,ok}
- Failed_slot_counter_1 = 0
- Agreed _slots_counter_1 = 0
- Acknowledgement_failure_counter_1 = 0

At reception of message:

- update of the variables following the algorithm

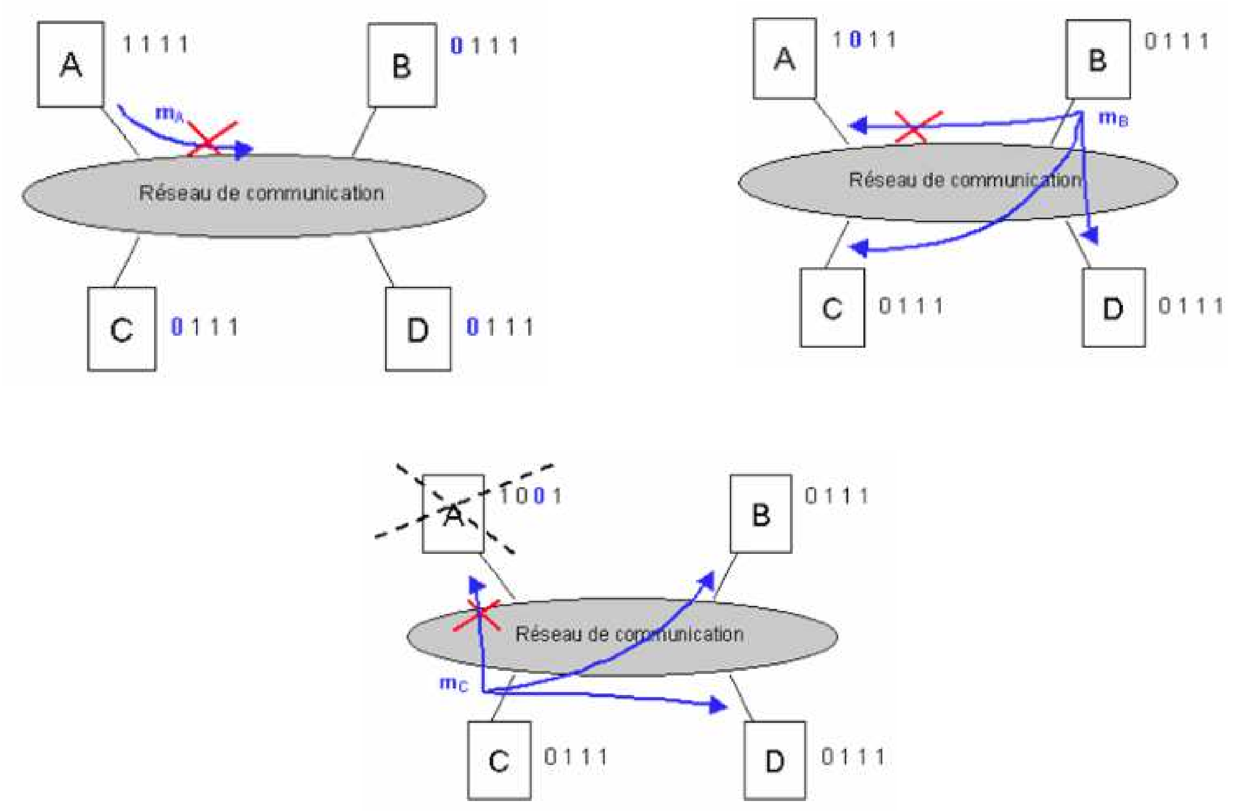
At the end of 1 TDMA round:

- adjustment of clock

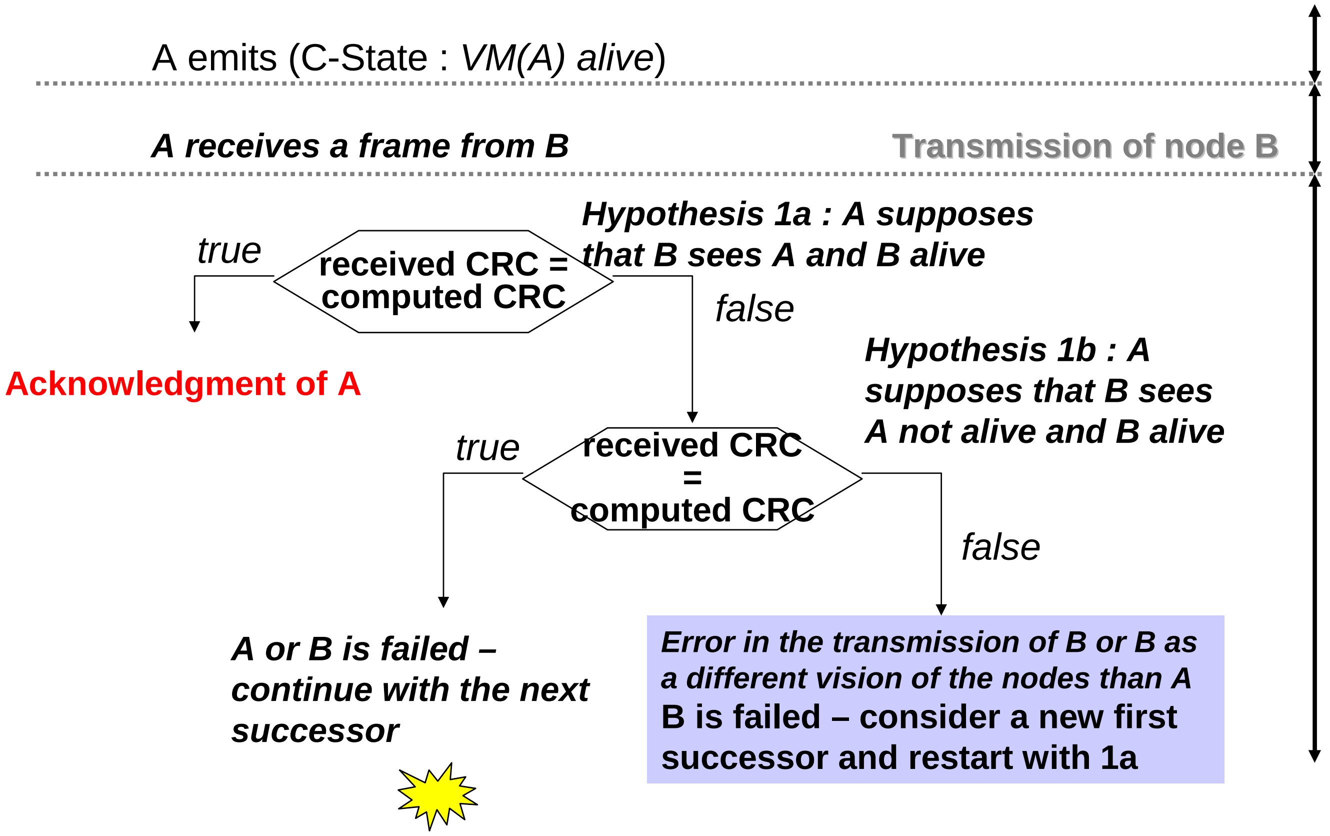
When the node is in its slot of emission:

- Verification:
  - it is alive: Cstate1[1]= ok
  - it has a correct vision of a system: agreed_slots_counter_1 > failed_slots_counter_1, otherwise the node disconnects because there are two many nodes that have a different vision of the world
- If every thing is fine, Failed_slot_counter_1 = 0, Agreed _slots_counter_1 = 0, Acknowledgement_failure_counter_1 = 0

Example of the membership functioning



Membership algorithm



A emits (C-State : *VM(A) alive*)

A receives a frame from B

Transmission of node B

received CRC = computed CRC

true

Acknowledgment of A

false

Hypothesis 1a : A supposes that B sees A and B alive

received CRC = computed CRC

true

A or B is failed – continue with the next successor

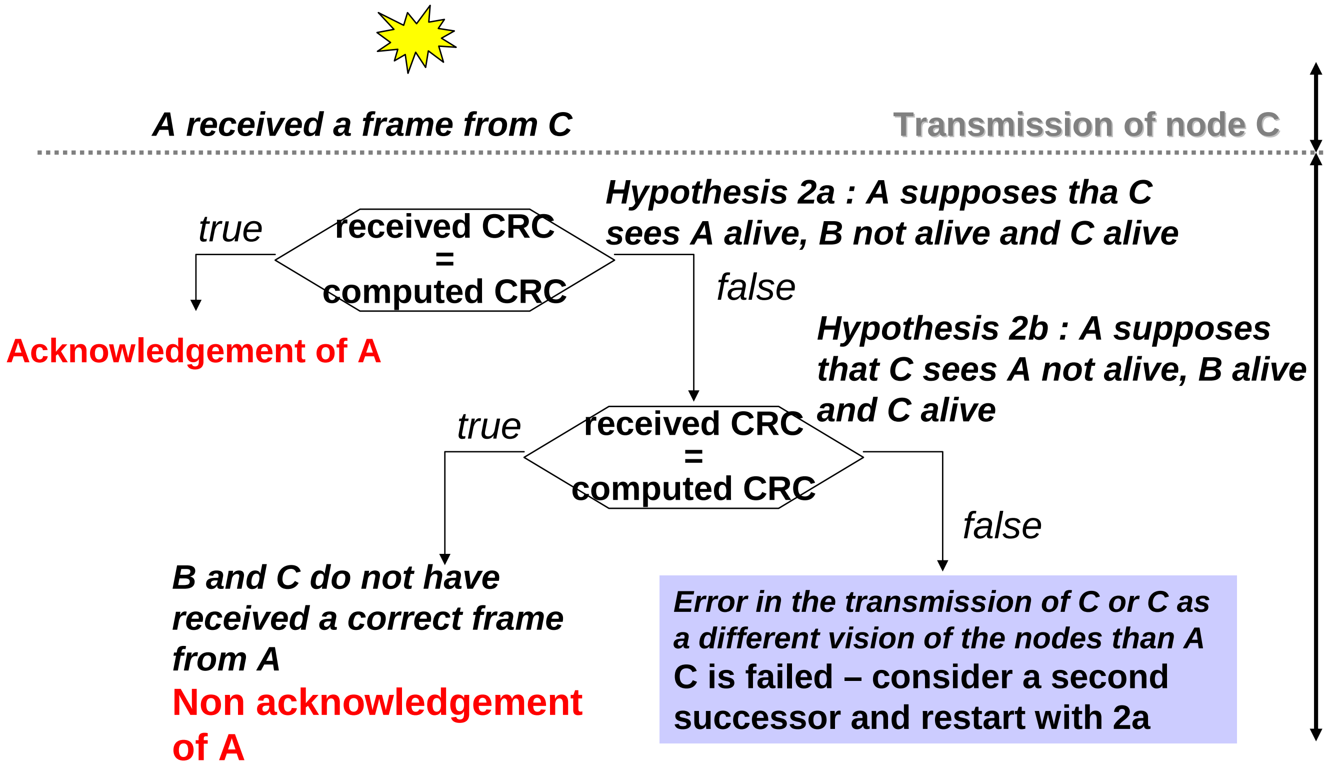
false

Hypothesis 1b : A supposes that B sees A not alive and B alive

Error in the transmission of B or B as a different vision of the nodes than A

B is failed – consider a new first successor and restart with 1a

Membership algorithm



A received a frame from C

Transmission of node C

received CRC = computed CRC

true

Acknowledgement of A

false

Hypothesis 2a : A supposes tha C sees A alive, B not alive and C alive

received CRC = computed CRC

true

B and C do not have received a correct frame from A

Non acknowledgement of A

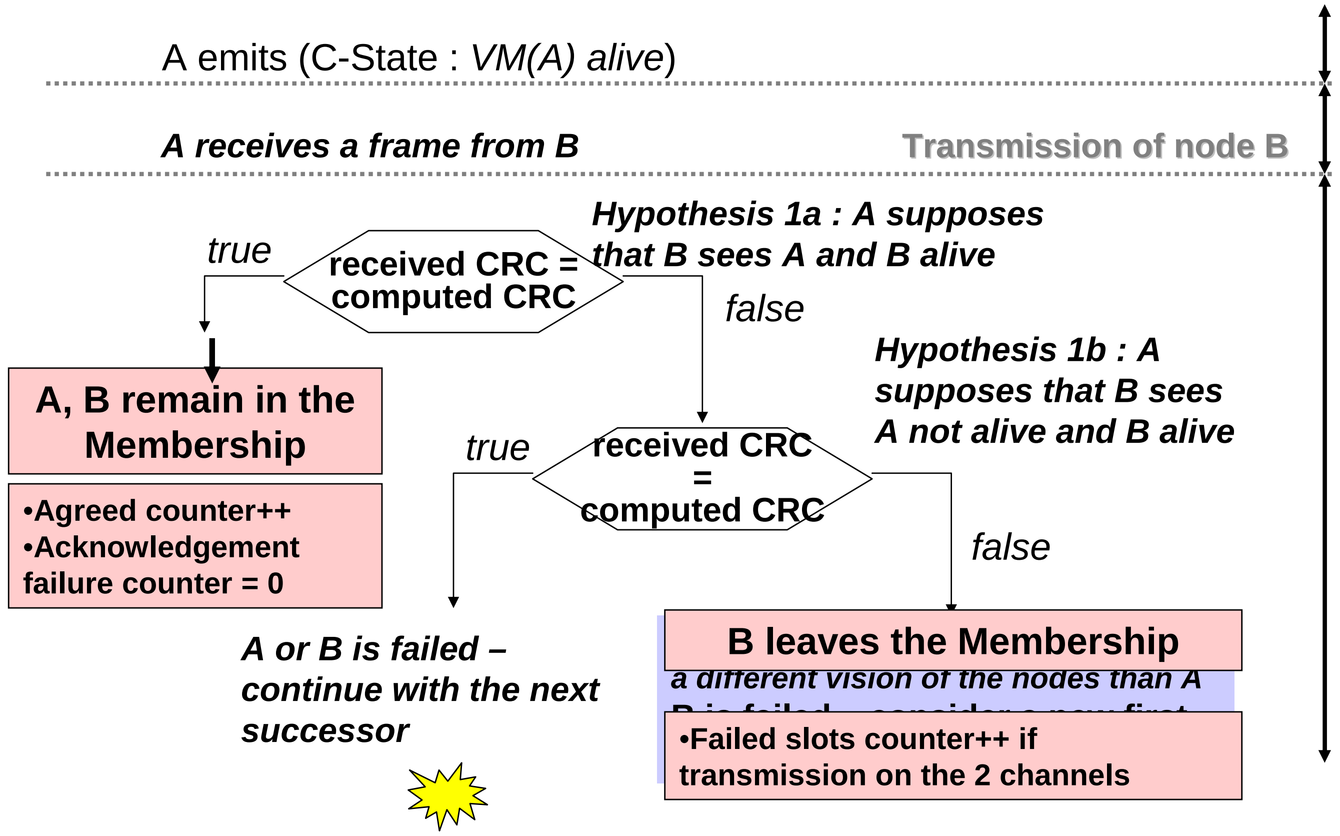
false

Hypothesis 2b : A supposes that C sees A not alive, B alive and C alive

Error in the transmission of C or C as a different vision of the nodes than A

C is failed – consider a second successor and restart with 2a

Membership algorithm



A emits (C-State : *VM(A) alive*)

A receives a frame from B

Transmission of node B

Hypothesis 1a : A supposes that B sees A and B alive

received CRC = computed CRC

true

A, B remain in the Membership

•Agreed counter++
•Acknowledgement failure counter = 0

false

Hypothesis 1b : A supposes that B sees A not alive and B alive

received CRC = computed CRC

true

A or B is failed – continue with the next successor

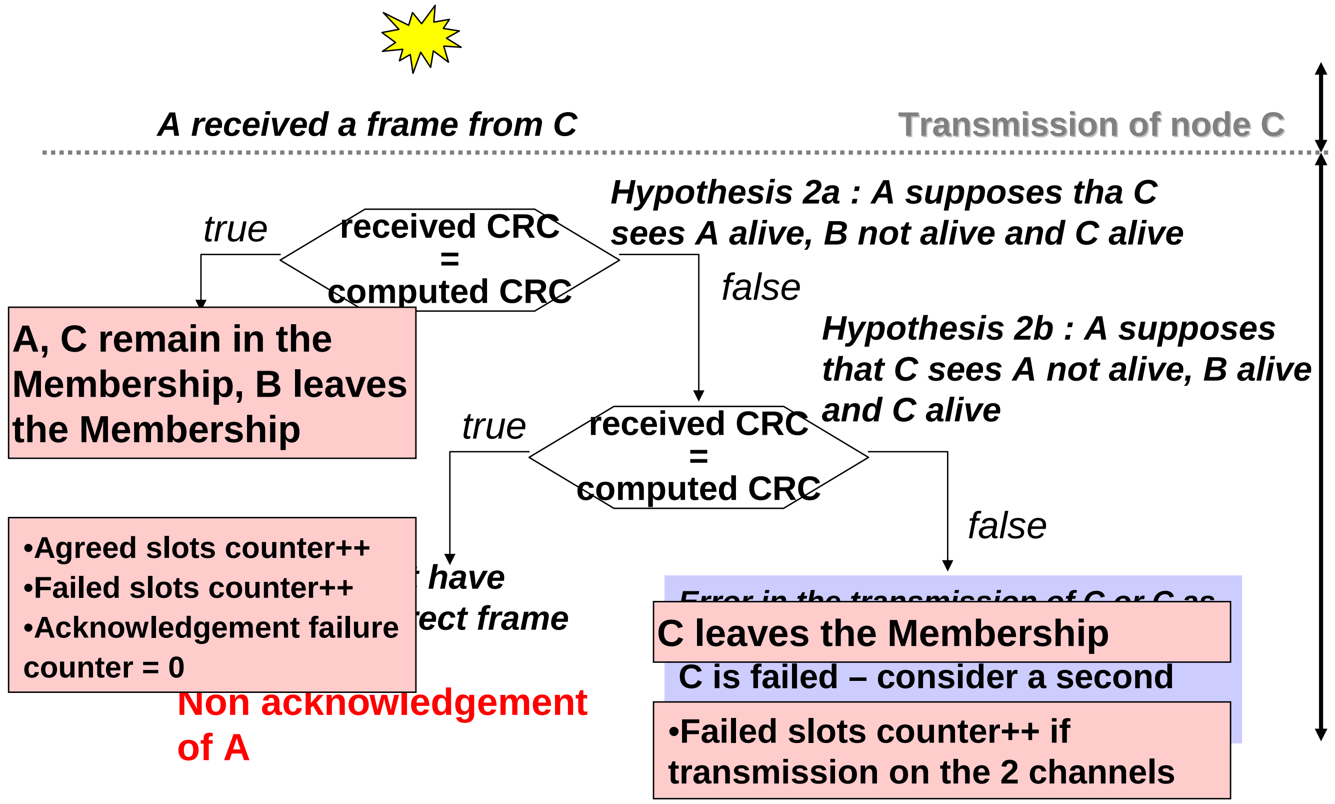
false

B leaves the Membership

a different vision of the nodes than A

•Failed slots counter++ if transmission on the 2 channels

Membership algorithm



A received a frame from C

Transmission of node C

Hypothesis 2a : A supposes tha C sees A alive, B not alive and C alive

received CRC = computed CRC

true

A, C remain in the Membership, B leaves the Membership

•Agreed slots counter++
•Failed slots counter++
•Acknowledgement failure counter = 0

Non acknowledgement of A

have
ect frame

false

Hypothesis 2b : A supposes that C sees A not alive, B alive and C alive

received CRC = computed CRC

true

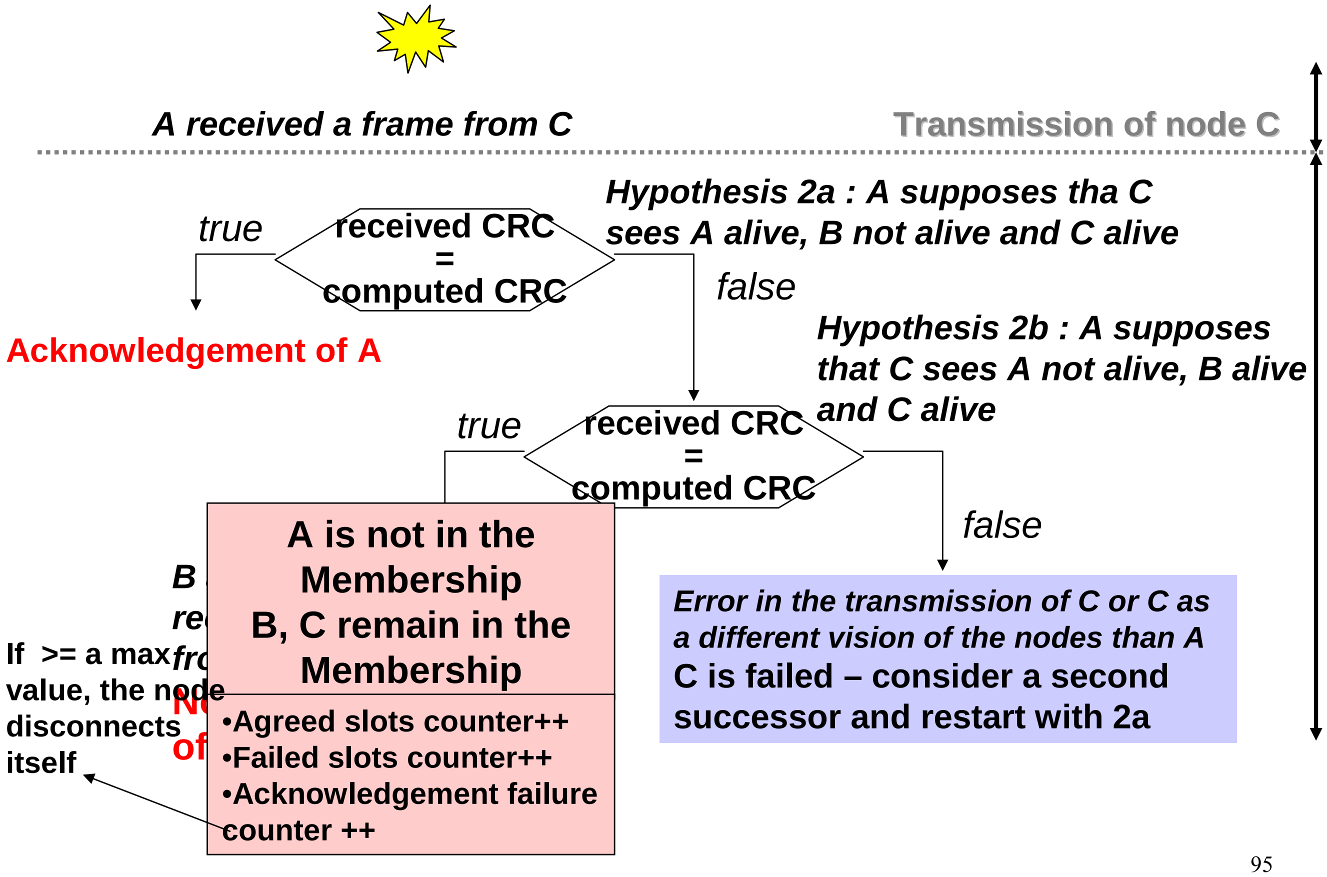
false

C leaves the Membership

C is failed – consider a second

•Failed slots counter++ if transmission on the 2 channels

Membership algorithm



A received a frame from C

Transmission of node C

received CRC = computed CRC

true

Acknowledgement of A

false

Hypothesis 2a : A supposes tha C sees A alive, B not alive and C alive

received CRC = computed CRC

true

A is not in the Membership
B, C remain in the Membership

•Agreed slots counter++
•Failed slots counter++
•Acknowledgement failure counter ++

If >= a max value, the node disconnects itself

B
re
fro
N
of

false

Hypothesis 2b : A supposes that C sees A not alive, B alive and C alive

Error in the transmission of C or C as a different vision of the nodes than A
C is failed – consider a second successor and restart with 2a

Membership mechanism

Clique detection

– Once a round, in the PSP (Pre Send Phase) before the slot of node A :
  - if agreed slots counter < failed slots counter, then the node disconnects
  - if agreed slots counter - failed slots counter < 2, then global error of the communicating system

Example

apply the algorithm on 4 nodes when N2 does not emit its message

Outline – Part II

1. **Application domain**
   1. Generalities
   2. Needs

2. **TTP/C**
   1. Presentation
   2. TDMA functioning
   3. Clock synchronisation
   4. Membership mechanism
   5. Fault-tolerance

Redundancy

Communication redundancy

- Critical data are emitted on both busses
- Non critical data can be emitted only on one bus

Message redundancy

- checksum on data: data is resent on the next round

Computation redundancy

- Active redundancy: two nodes execute in parallel and emit on their own slot
- Passive redundancy or « Shadow Node »
  - Two nodes : a master and a shadow
  - The shadow node emits only when the master is disconnected after a failure
  - A unique slot is shared by the node and its shadow in each round
  - A shadow node can be shared by several nodes

Fault model

Give the fault model of a TTP architecture

Give some failures which are not supported

References

- Course on TTP, Françoise Simonot-Lion.
- H. Kopetz. TTP/A - A Time-Triggered Protocol for Body Electronics Using Standard UARTS. In /International Congress and Exposition Detroit, Michigan/, Commonwealth Drive, Warrendale, Febr.-March 1995. The Engineering Society For Advancing Mobility Land Sea Air and Space, SAE International
- John Rushby. A Comparison of Bus Architectures for Safety-Critical Embedded Systems pageweb CSL Technical Report, September 2001.
- Karen Godary. Les réseaux des systèmes embarqués, le cas de TTP, 2009. pageweb