

**Distributed Systems**  
**Master of Science in Engineering in Computer Science**

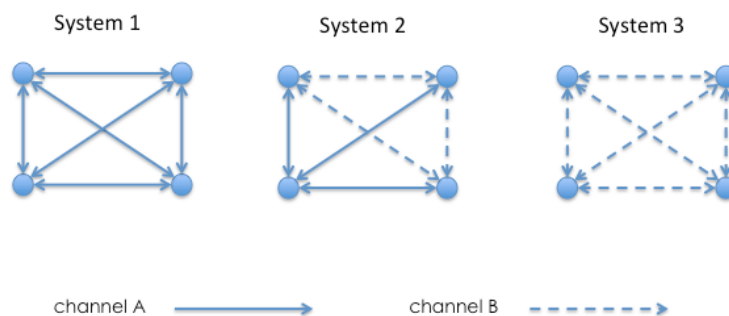
**AA 2020/2021**

**Lecture 7 – Exercises**

**Ex 1:** Let channel A and channel B be two different types of point-to-point channels satisfying the following properties:

- channel A: if a correct process  $p_i$  sends a message  $m$  to a correct process  $p_j$  at time  $t$ , then  $m$  is delivered by  $p_j$  by time  $t + \delta$
- channel B: if a correct process  $p_i$  sends a message  $m$  to a correct process  $p_j$  at time  $t$ , then  $m$  is delivered by  $p_j$  with probability  $p_{\text{cons}}$  ( $p_{\text{cons}} < 1$ ).

Let us consider the following systems composed by 4 processes  $p_1, p_2, p_3$  and  $p_4$  connected through channels A and channels B.



Assuming that each process  $p_i$  is aware of the type of channel connecting it to any other process, answer to the following questions:

1. is it possible to design an algorithm implementing a perfect failure detector in system 2 if only processes having an outgoing channel of type B can fail by crash?
2. is it possible to design an algorithm implementing a perfect failure detector in system 2 if any process can fail by crash?
3. is it possible to design an algorithm implementing a perfect failure detector in system 3?

For each point, if an algorithm exists write its pseudo-code, otherwise show the impossibility.

**Ex 2:** Consider a distributed system composed by  $n$  processes  $\{p_1, p_2, \dots, p_n\}$  that communicate by exchanging messages on top of a line topology, where  $p_1$  and  $p_n$  are respectively the first and the last process of the network.

Initially, each process knows only its left neighbour and its right neighbour (if they exist) and stores the respective identifiers in two local variables LEFT and RIGHT.

Processes may fail by crashing, but they are equipped with a perfect oracle that notifies at each process the new neighbour (when one of the two fails) through the following primitives:

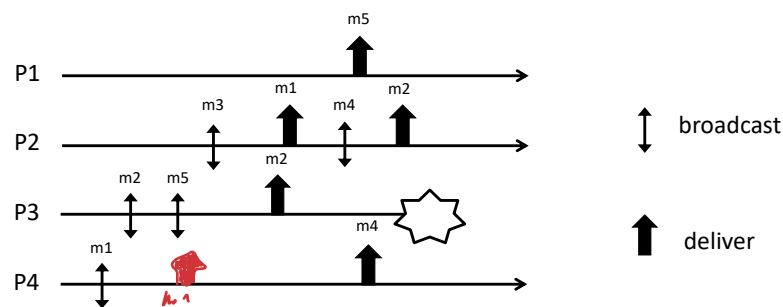
- **Left\_neighbour( $p_x$ ):** process  $p_x$  is the new left neighbour of  $p_i$
- **Right\_neighbour( $p_x$ ):** process  $p_x$  is the new right neighbour of  $p_i$

Both the events may return a NULL value in case  $p_i$  becomes the first or the last process of the line.

Each process can communicate only with its neighbours.

Write the pseudo-code of an algorithm implementing a Leader Election primitive assuming that channels connecting two neighbour processes are perfect.

**Ex 3:** Consider the partial execution depicted in the Figure



Answer to the following questions:

1. Complete the execution in order to have a run satisfying Uniform Reliable Broadcast.
2. Complete the execution in order to have a run satisfying Regular Reliable Broadcast but not Uniform Reliable Broadcast.
3. Complete the execution in order to have a run satisfying Best Effort Broadcast but not Regular Reliable Broadcast.

**NOTE:** In order to solve the exercise you can add broadcast, deliver and crash events but you cannot remove anything from the run.