Distributed Systems
Master of Science in Engineering in Computer Science

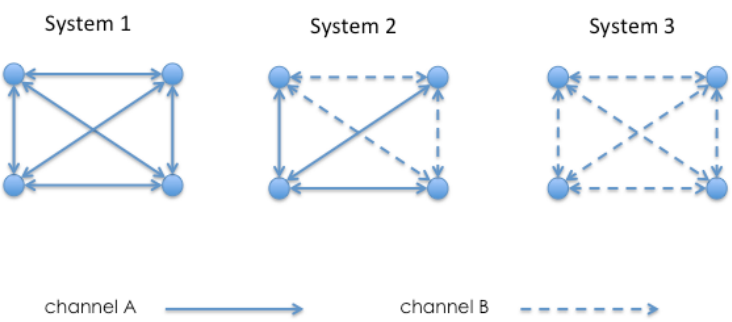
AA 2019/2020

Lecture 6 – Exercises
October 15th, 2019

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_{cons} ($p_{cons} < 1$).

Let us consider the following systems composed by 4 processes p_1, p_2, p_3 and p_4 connected trough 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 $\{p_1, p_2, \ldots, p_n\}$ 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.

Solutions

Exercise 1.1

Init
$correct_i = \{p1, p2, p3, p4\}$
$alive_i = \emptyset$
$detected_i = \emptyset$
for each $p_j \in correct_i$ **do**:
    trigger send (HB_REQ, i) to p_j
start_Timer1(2δ)

upon event deliver (HB_REQ, j) from p_j
    trigger send (HB_REL, i) to p_j

upon event deliver (HB_REL, j) from p_j
    $alive_i = alive_i \cup \{p_j\}$

when timer1 = 0
    for each $p_j \in correct_i$ **do**
        trigger send (ALIVE_LIST, $alive_i$, i) to p_j
    start_Timer2(δ)

upon event deliver (ALIVE_LIST, $alive_j$, j) from p_j
    $alive_i = alive_i \cup alive_j$

when timer2 = 0 do
    for each $p_j \in correct_i$ **do**
        if $p_j \notin alive_i$ AND $p_j \notin detected_i$
            $detected_i = detected_i \cup \{p_j\}$

trigger crash (p_j)

alive$_i$ = $\varnothing$

for each $p_j \in$ correct$_i$ **do**:

  trigger send (HB_REQ, i) to p_j

start_Timer1(2δ)

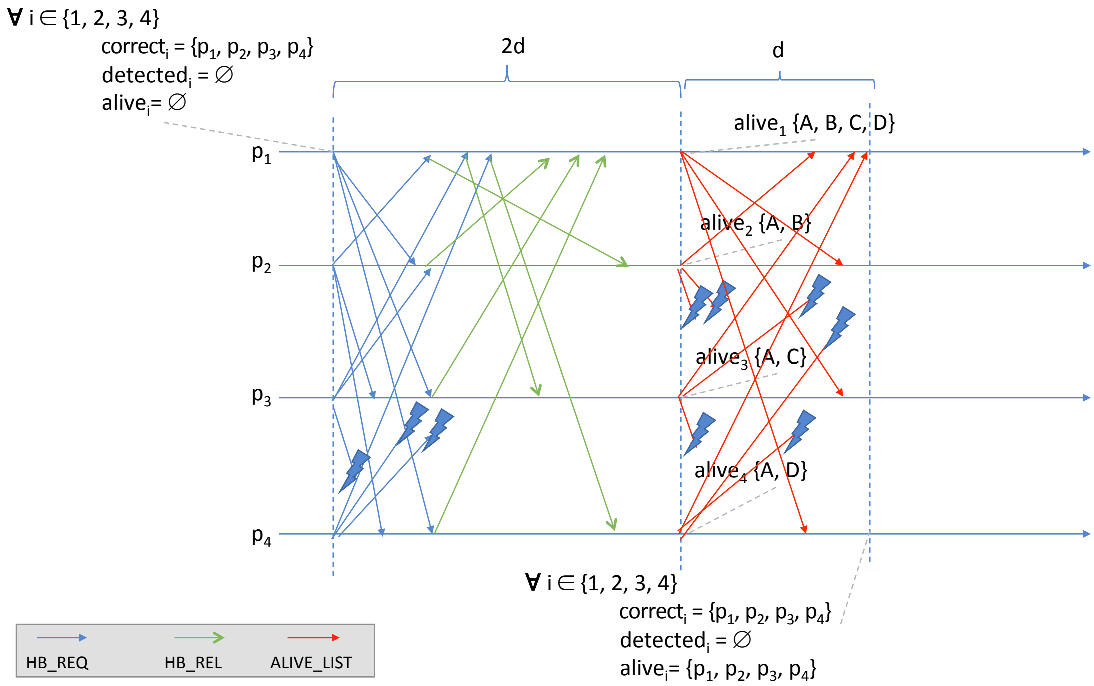


Figure 1 – Example of the Algorithm execution with no failures and having p_1 connected with just channel A links

Exercise 1.2

No, you cannot as once the process having all channel A fails (i.e., p_1 in the example) you may lose accuracy

Exercise 1.3

No, because channels B are like fair loss and you cannot implement P on fair loss links.

Exercise 2

Uses

Oracle O_i

Perfect point to point link

Init

leader$_i$ = $\perp$

left$_i$ = get_left() %initialize left with my current left neighbour

right$_i$ = get_right() %initialize left with my current right neighbor

when $left_i$ = null **do**
    $leader_i = p_i$
    trigger leader(p_i)
    trigger send (NEW_LEADER, $leader_i$) to $right_i$

upon event deliver (NEW_LEADER, l) from $left_i$
    if $leader_i \neq l$
        $leader_i = l$
        trigger leader(l)
        trigger send (NEW_LEADER, $leader_i$) to $right_i$

upon event left_neighbour(p_j)
    $left_i = p_j$

upon event right_neighbour(p_j)
    $right_i = p_j$
    trigger send (NEW_LEADER, $leader_i$) to $right_i$