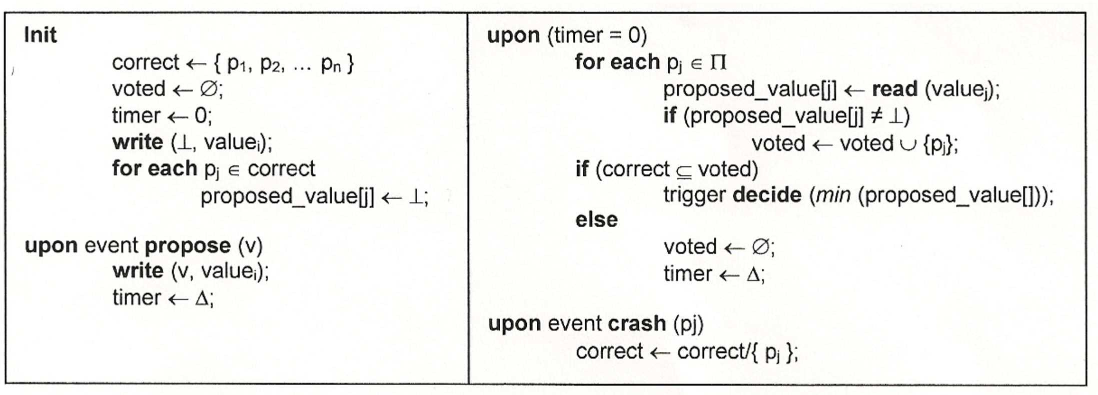
Dependable Distributed Systems
Master of Science in Engineering in Computer Science

AA 2024/2025

Week 7 – Exercises
November 14th, 2024

Ex 1: Consider a system composed of *n* processes $\Pi = \{p_1, p_2, ..., p_n\}$. Every process p_i has access to a failure detector P and to a register $value_i$ of type (1,N). Consider the following protocol executed by every process p_i



```
Init
        correct ← { p1, p2, ... pn }
        voted ← ∅;
        timer ← 0;
        write (⊥, value_i);
        for each p_j ∈ correct
                proposed_value[j] ← ⊥;

upon event propose (v)
        write (v, value_i);
        timer ← Δ;

upon (timer = 0)
        for each p_j ∈ Π
                proposed_value[j] ← read (value_j);
                if (proposed_value[j] ≠ ⊥)
                        voted ← voted ∪ {p_j};
        if (correct ⊆ voted)
                trigger decide (min (proposed_value[]));
        else
                voted ← ∅;
                timer ← Δ;

upon event crash (pj)
        correct ← correct/{ p_j };
```

The student answers the following questions

1. Assuming that processes are not going to fail during the execution of the algorithm, which is the weakest register specification that makes the protocol correct with respect to the specification of the consensus problem?
2. Assuming that processes may fail by crash, does the protocol shown in the figure implement some kind of consensus? If yes, is it uniform or non-uniform consensus?

Justify both answers appropriately (if necessary, also with examples).

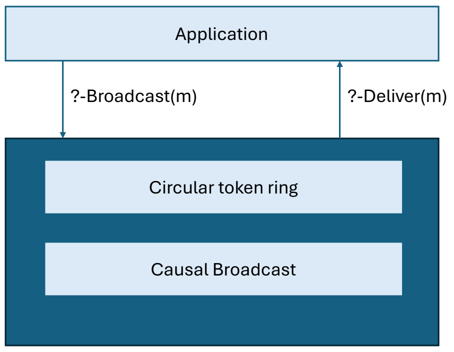
Ex 2: Let us consider a distributed system composed of *k* groups of processes g_1, g_2, … g_n (with $k>1$). Each group g_i is composed by a number of processes n_i and it has a leader. Write the pseudo-code of a distributed protocol implementing a total order broadcast primitive among all the processes of the distributed system considering that:

- Each process may communicate with its leader using a FIFO perfect point-to-point link;
- Processes do not crash;
- Leaders own a consensus primitive and can communicate with processes in their group by using FIFO perfect point-to-point links.

Solution for Ex 2

```
INIT
unordered = ∅
delivered = ∅
consensus_running = false
leader = get_my_leader()



upon event TOBroadcast(m)
    trigger FIFO_p2p_send(MSG, m) to leader


upon event FIFO_p2p_Deliver (MSG, m)
    unordered = unordered ∪ {m}


when unordered ≠ ∅ and not consensus_running
    consensus_running = true
    trigger propose(unordered)

upon event decide(list)
    sort(list)
    for each m ∈list
        for each pi ∈ my_group
            trigger FIFO_pp2p_send(DELIVER, m) to pi
    unordered = unordered \ list
    consensus_running = false


upon event FIFO_pp2p_Deliver(DELIVER, m)
    if m ∉delivered
        delivered = delivered ∪ {m}
        trigger TODeliver()
```

Ex 3: Let us consider the following protocol stack implementing a "?-Broadcast" primitive



When the "?-Broadcast(*m*)" event is invoked by the application running on a process p_i, p_i keeps locally *m* in a buffer and waits until it gets the token[1]. When p_i owns the token, it takes m from the buffer and spreads it using the Causal Broadcast primitive. When a process p_j causally deliver a message *m*, it immediately delivers *m* to the application by triggering the "?-Deliver(m)" event.

Which type of broadcast primitive is implemented by the "?-Broadcast()" box? Justify your answer.

[1] The circular token ring block is responsible for the creation of a unique token and its perpetual circulation between processes. It follows that every process will receive the token infinitely often.