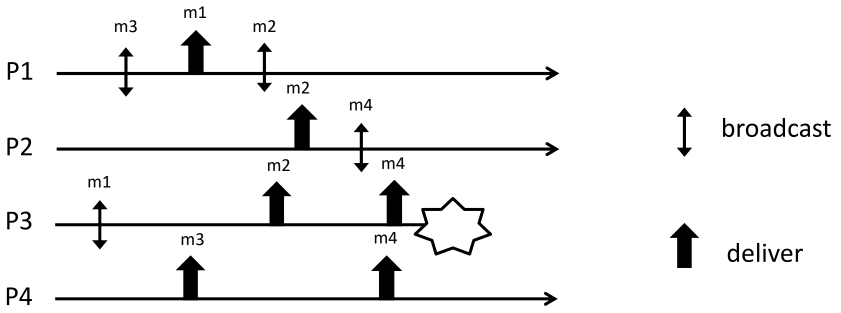
# Dependable Distributed Systems
# Master of Science in Engineering in Computer Science

## AA 2022/2023

## Lecture 19 – Exercises
## November 16th, 2022

**Ex 1:** Consider the partial execution depicted in the Figure

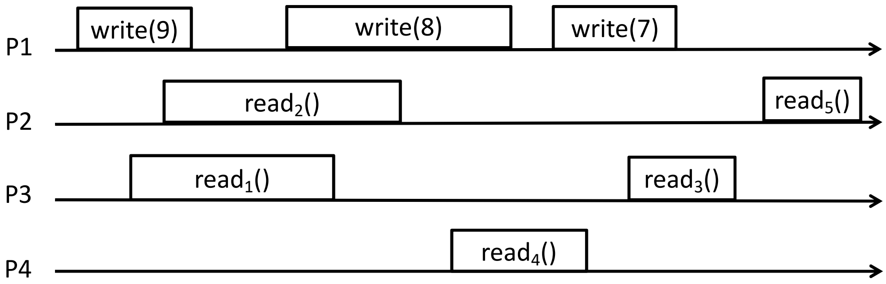


Answer to the following questions:

1. Provide ALL the possible delivery sequences that satisfies causal order and TO (UA, SUTO)
2. Complete the execution in order to have a run satisfying TO (UA WNUTO), FIFO order Broadcast but not Causal Order Broadcast
3. Complete the execution in order to have a run satisfying Regular Reliable Broadcast but not Uniform Reliable Broadcast and not satisfying Total Order.

**NOTE:** In order to solve the exercise, you can only add broadcast, deliveries and failures.

**Ex 2:** Consider the partial execution depicted in the Figure

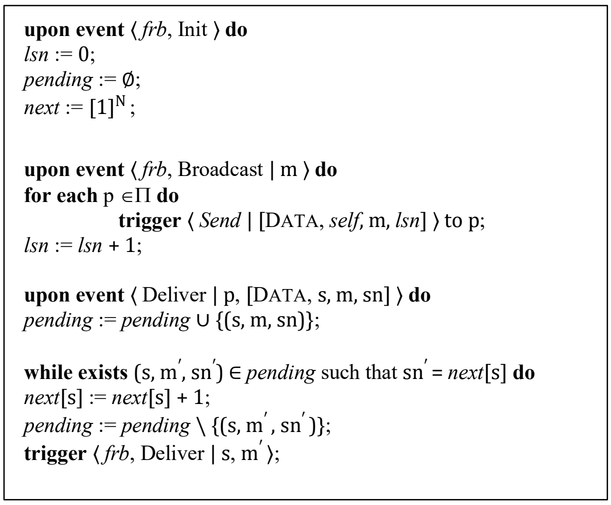


Answer to the following questions:

1. Define <u>ALL</u> the values that can be returned by read operations (Rx) assuming that the run refers to a regular register.
2. Define <u>ALL</u> the values that can be returned by read operations (Rx) assuming that the run refers to an atomic register.

3. Assign to each read operations (Rx) a return value that makes the execution linearizable.

**Ex 3:** Let us consider the following algorithm



```
upon event ⟨frb, Init⟩ do
lsn := 0;
pending := ∅;
next := [1]^N ;

upon event ⟨frb, Broadcast | m⟩ do
for each p ∈Π do
        trigger ⟨ Send | [DATA, self, m, lsn] ⟩ to p;
lsn := lsn + 1;

upon event ⟨ Deliver | p, [DATA, s, m, sn] ⟩ do
pending := pending ∪ {(s, m, sn)};

while exists (s, m', sn') ∈ pending such that sn' = next[s] do
next[s] := next[s] + 1;
pending := pending \ {(s, m', sn')};
trigger ⟨frb, Deliver | s, m'⟩;
```

Let us consider the following properties:

- ***Validity:*** If a correct process p broadcasts a message m, then p eventually delivers m.
- ***No duplication***: No message is delivered more than once.
- ***No creation:*** If a process delivers a message m with sender s, then m was previously broadcast by process s.
- ***Agreement***: If a message m is delivered by some correct process, then m is eventually delivered by every correct process.
- ***FIFO delivery***: If some process broadcasts message $m_1$ before it broadcasts message $m_2$, then no correct process delivers $m_2$ unless it has already delivered $m_1$.

Assuming that every process may fail by crash, address the following points:

1. Considering that messages are sent by using *perfect point to point links*, for each property mentioned, discuss if it satisfied or not and provide a motivation for your answer:
2. Considering that messages are sent by using *fair loss links*, for each property mentioned, discuss if it satisfied or not and provide a motivation for your answer.

**Ex 4:** Consider a distributed system constituted by *n* processes $\prod=\{p_1, p_2\ldots p_n\}$ with unique identifiers that exchange messages through perfect point-to-point links and are structured through a ring (i.e., each process $p_i$ can exchange messages only with processes and $p_{i+1(\mathrm{mod}\ n)}$). Processes may crash and each process is equipped with a perfect oracle (having the interface *new_next(p)*) reporting a new neighbor when the previous one is failing.

Write the pseudo-code of an algorithm implementing a Uniform Reliable Broadcast communication primitive.

**Ex 5:** Let us consider the following algorithm implementing a (1, N) atomic register in synchronous system.

```
1  upon event ⟨ onar, Init ⟩ do
2  (ts, val) := (0, ⊥);
3  correct := Π;
4  writeset := ∅;
5  readval := ⊥;
6  reading := FALSE;

7  uponevent⟨P,Crash |p⟩do
8  correct := correct \ {p};

9  upon event ⟨ onar, Read ⟩ do
10 reading := TRUE;
11 readval := val;
12 trigger ⟨ beb, Broadcast | [WRITE, ts, val] ⟩;

13 upon event ⟨ onar, Write | v ⟩ do
   trigger ⟨ beb, Broadcast | [WRITE, ts + 1, v] ⟩;
```

```
14 upon event ⟨ beb, Deliver | p, [WRITE, ts', v'] ⟩ do
15 if ts' > ts then
16          (ts, val) := (ts' , v' );
17 trigger ⟨ pl, Send | p, [ACK] ⟩;

18 upon event ⟨ pl, Deliver | p, [ACK] ⟩ then
19 writeset := writeset ∪ {p};

20 upon correct ⊆ writeset do
21 writeset := ∅;
22 if reading = TRUE then
23         reading := FALSE;
24         trigger ⟨ onar, ReadReturn | readval ⟩;
25 else
26      trigger ⟨ onar, WriteReturn ⟩;
```

Assuming that messages are sent by using perfect point-to-point links and that the broadcast is best effort answer the following questions:

1. Discuss what does it happen to every atomic register property (i.e., termination, validity and ordering) if the failure detector in eventually perfect and not perfect
2. Discuss what does it happen to every atomic register property (i.e., termination, validity and ordering) if we change line 12 with **trigger** ⟨ *beb*, Broadcast | [WRITE, ts+1, val] ⟩;

**Ex 6:** Consider a distributed system composed of *n* processes $\prod=\{p_1, p_2\dots p_n\}$ with unique identifiers that exchange messages through perfect point-to-point links. Processes are connected through a directed ring (i.e., each process $p_i$ can exchange messages only with processes and $p_{i+1(\text{mod } n)}$). Processes may crash and each process is equipped with a perfect oracle (having the interface *new_next(p)*) reporting a new neighbor when the previous one is failing.
Write the pseudo-code of an algorithm implementing a Leader Election primitive at every process $p_i$.