FATTO

Distributed Systems
10/06/2020

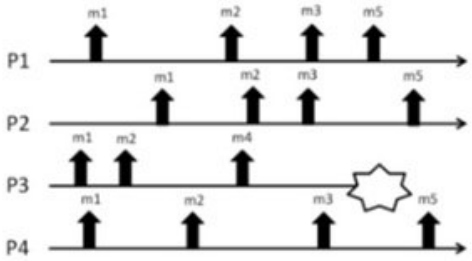
Family Name_____ Name_____ Student ID_____

Please, tick the appropriate option:

- ☐ Master of Science in Engineering in Computer Science
- ☐ Master of Science in Artificial Intelligence and Robotics
- ☐ Erasmus
- ☐ Other

X **Ex 1:** Concerning software replication techniques, describe the primary-backup and the active replication approach with particular emphasis on how each technique handles failures.

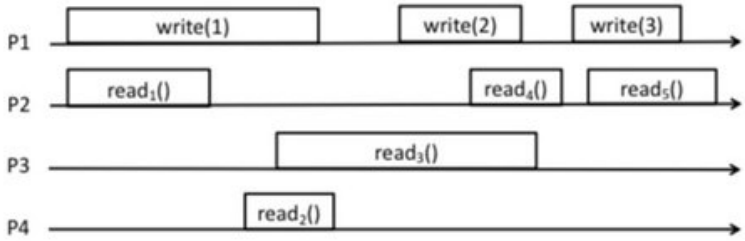
X **Ex 2:** Consider the run depicted in the figure:



1. Which type of total ordering is satisfied by the run? Specify both the agreement and the ordering properties.
2. Modify the run in order to satisfy TO(UA, WUTO) but not TO (UA SUTO)
3. Modify the run in order to satisfy TO(NUA, WNUTO) but not TO(NUA, WUTO)

NOTE: In order to solve the exercise, you can just ADD broadcast, deliveries and failures.

X **Ex 3:** Consider the execution depicted in the following figure and answer the questions



1. Define ALL the values that can be returned by read operations (Rx) assuming the run refers to a regular register.
2. Define ALL the values that can be returned by read operations (Rx) assuming the run refers to an atomic register.
3. Let us assume that values retuned by read operations are as follow: $read_1() \rightarrow 1$, $read_2() \rightarrow 0$, $read_3() \rightarrow 1$, $read_4() \rightarrow 2$, $read_5() \rightarrow 3$. Is the run depicted in the Figure linearizable?

Ex 4: 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
14. trigger ⟨ beb, Broadcast | [WRITE, ts + 1, v] ⟩;
```

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

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

21. upon correct ⊆ writeset do
22. writeset := ∅;
23. if reading = TRUE then
24.         reading := FALSE;
25.         trigger ⟨ onar, ReadReturn | readval ⟩;
26. else
27.         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 happen to every atomic register property (i.e., termination, validity and ordering) if messages can be lost.

MANCA

2. Discuss what happen to every atomic register property (i.e., termination, validity and ordering) if we change line 12 (i.e., **trigger** ⟨ *beb*, Broadcast | [WRITE, ts, val] ⟩; in the read handler) with **trigger** ⟨ *beb*, Broadcast | [WRITE, ts+1, val] ⟩;

Ex 5: Consider a distributed system composed of *n* processes $\prod=\{p_1, p_2 \ldots p_n\}$ with unique identifiers that exchange messages through fair loss 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 Regular Reliable Broadcast.

According to the Italian law 675 of the 31/12/96, I authorize the instructor of the course to publish on the web site of the course results of the exams.

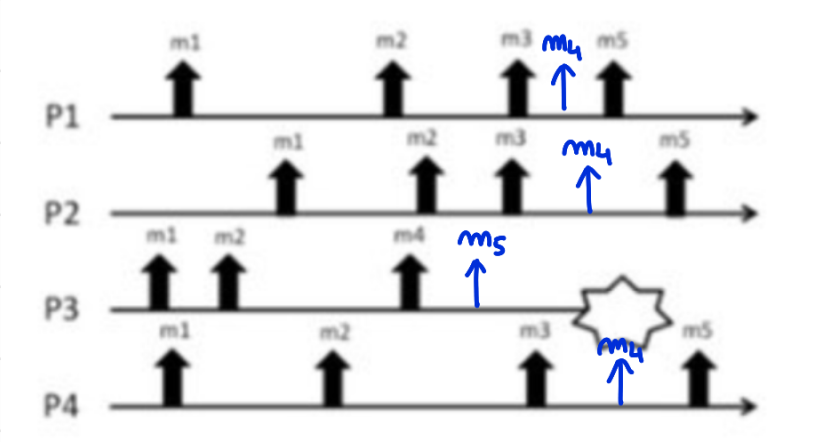
Signature: _____

1) The primary-backup and the active replication are two techniques that implement liveanizability.
The first can be explain as: primary means that receives invocation from clients and sends back the answers; also given an objects x, prim(x) returns the primary of x. While backup means that it interacts with prim(x) and is used to guarantee fault tolerance by replacing a primary when there're crashes.
In the case of NO CRASH scenario, before sending back the response to the client, the primary replica sends an update to all the other correct backups and, only after it gets an ACK from them, it will send a response to the client.
In the case of CRASH, we have different scenarios:
① the primary fails after the client receives the answer and there are two cases:
the client doesn't receive the response due to PP2P link and the client needs to retrasmit the requests; the client receives the answer and it's fine.
② the primary fails before sending update msgs:
client doesn't gets answer and needs to resend the requests that is handled as new.
③ the primary fails after sending update msgs but before receiving all ACK.
The update is received either by all or by no one, so the primary have to be elected among all replicas.

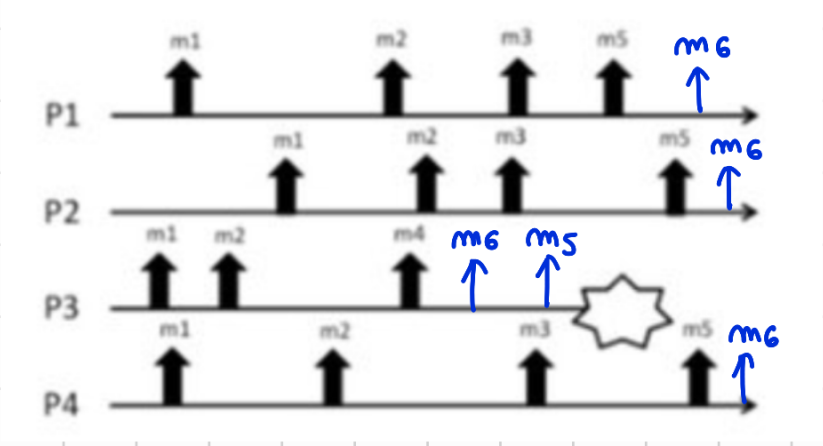
In the active replication there isn't a coordinator, all replicas have the same role. Each replica is deterministic, so the client receives the same response for all replicas. It have to ensure liveaniza- bility and needs to preserve atomicity: if a replica executes an invocation, all correct replicas execute the same invocation, and ordering: no two correct replicas have to execute two invocations in different order.
It doesn't need recovery action upon the failure of a replica.

2) 2.1 TO(NUA, SUTO)

2.2 (UA, WUTO), not (UA, SUTO)



2.3 (NUA, WNUTO), not (NUA, WUTO)



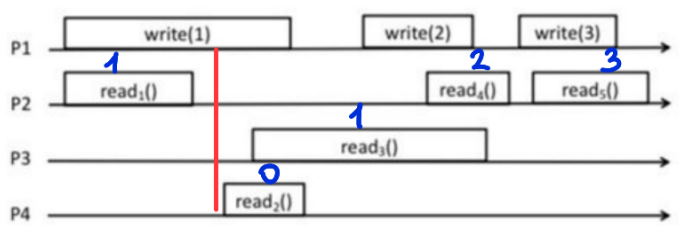
3) 3.1 Regular register

R_1 : 0, 1
R_2 : 0, 1
R_3 : 0, 1, 2
R_4 : 1, 2
R_5 : 2, 3

3.2 Atomic register

R_1 : 0, 1
R_2 : 0, 1
R_3 : 0, 1, 2 or 1, 2 ⟶ depends on R_1 because if R_1 has (0), R_3 can have (0,1,2) otherwise (1,2)
R_4 : 1, 2
R_5 : 2, 3

3.3 $R_1=1$ $R_2=0$ $R_3=1$ $R_4=2$ $R_5=3$

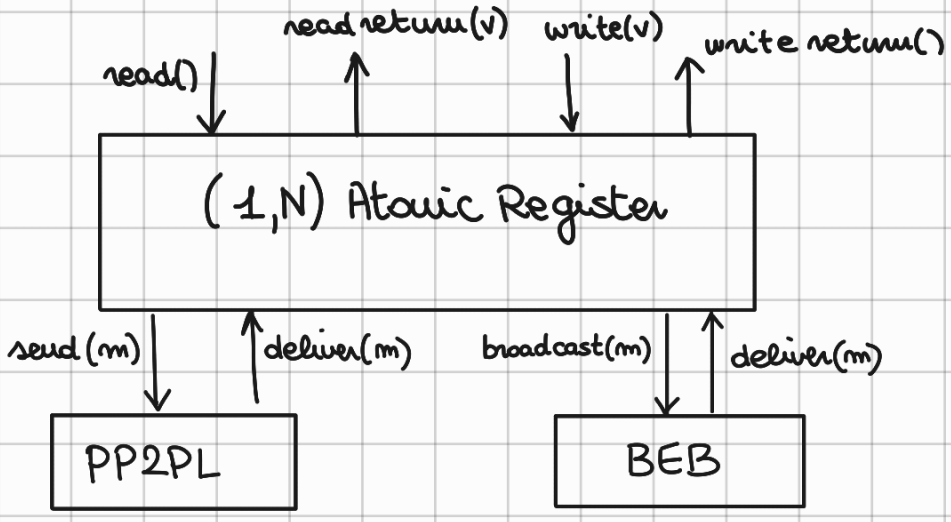


No, because R_1 and R_2 are not concurrent and R_2 returns 0 after $R_1=1$. Violation!

4) PP2PL

BEB

(1,N) atomic register



4.1 Messages can be lost

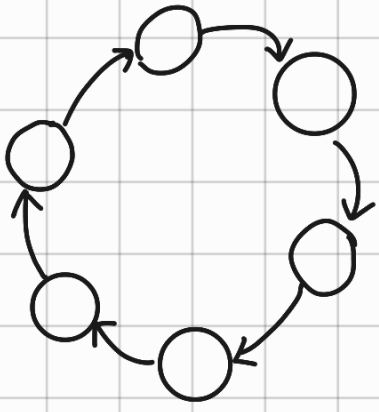
Termination → if a correct process invokes an operation, then the operation eventually completes

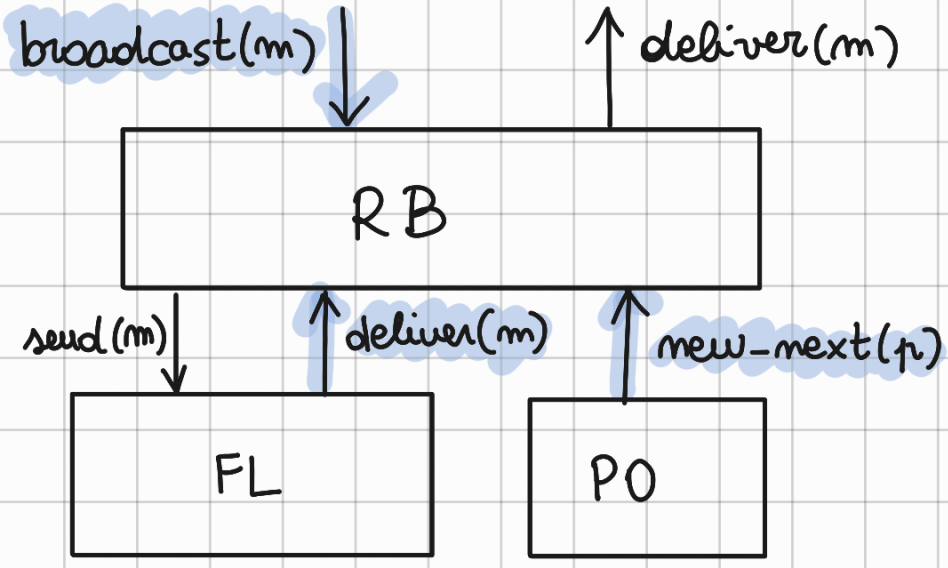
Validity → A read that is not concurrent with a write, returns the last value written;
A read that is concurrent with a write, returns the last value written o the value concurrently written.

Ordering → If a read returns a value v and a subsequent read returns a value w, then the write of w does not precede the write of v.

$$R(v) < R(w)$$
$$W(v) < W(w)$$

Ex 5: Consider a distributed system composed of n processes $\Pi=\{p_1, p_2 \dots p_n\}$ with unique identifiers that exchange messages through fair loss 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 Regular Reliable Broadcast.





The protocol know the set of processes and when receive the broadcast event, spread the msg though the fair loss link. We know that using this type of communication channel, msgs can be lost (properties: fair-loss, finite duplication, no creation).

upon event $\langle$ rb, INIT $\rangle$ do
    correct = Π
    next = $p_{(i+1) \bmod n}$
    from [p] = $[\emptyset]^N$

upon event $\langle$ rb, BROADCAST | m $\rangle$ do