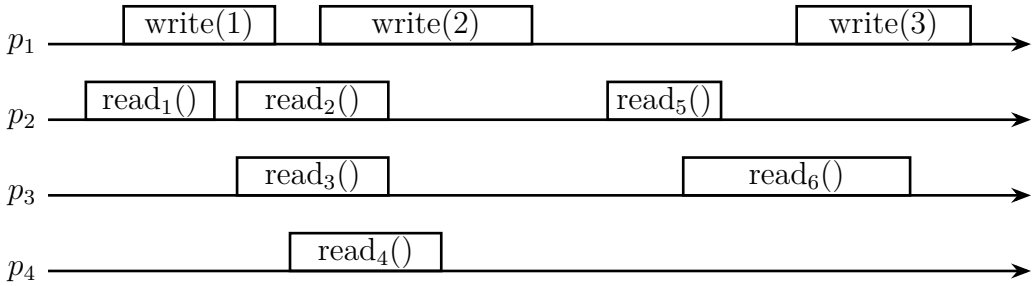
# DDS Exam - 07.06.2023

## Exercise 1

Discuss blockchain from distributed system point of view, the distinction between public, private, permissioned and premissionless. In addition, explain how PoW (Proof-of-Work) mechanism works and why it creates branches in the blockchain data structure.

## Exercise 2

Consider the execution depicted in the following figure.

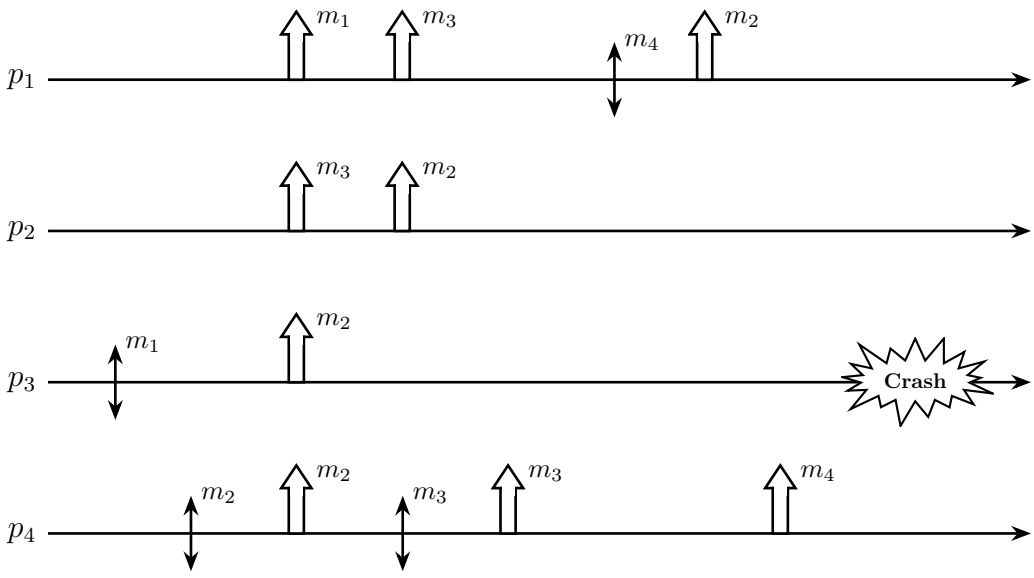



Answer some true/false questions I don't remember, but basically answer the questions:

1. Define ALL values that can be returned by read operations (Rx) assuming the run refers to a regular register
2. Define ALL values that can be returned by read operations (Rx) assuming the run refers to an atomic register
3. Provide a sequence such that the execution is linearizable

## Exercise 3

Consider the message pattern shown in the Figure below.




Answer some true/false questions I don't remember, but basically questions on broadcast communications and ordered communications.

RIP

## Exercise 4

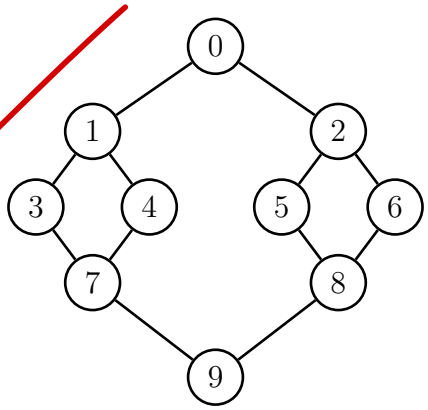
**Algorithm: Beb**

```
1  upon event Init
2      ID := unique integer identifier;
3      Π := IDs of the neighbors;

4  upon event ⟨ beb, Broadcast | m ⟩
5      trigger ⟨ beb, Deliver | ID, m ⟩;
6      forall k ∈ Π do
7          trigger ⟨ pl, Send | k, ⟨ ID, m ⟩ ⟩;

8  upon event ⟨ pl, Deliver | j, ⟨ i, m ⟩ ⟩
9      trigger ⟨ beb, Deliver | i, m ⟩;
10     forall k ∈ Π do
11         if (j < ID ∧ k < i) ∨ (j > ID ∧ k > i) then
12             trigger ⟨ pl, Send | k, ⟨ i, m ⟩ ⟩

13         if k == 9 then
14             trigger ⟨beb, Deliver | i, m ⟩;
```




The distributed algorithm is executed at every process. In particular, $p_0$ is the process that starts the Beb broadcast.

Assume that links are perfect and load independent (i.e., $p_i \to p_j$ and $p_j \to p_i$ are independent and do not impact performance of other links). Answer the questions:

1. Assume that $p_0$ sends a message every 2 seconds, and every process is able to process 1 message per second, and assume that the distribution of arrivals and distribution of services are exponentially distributed, compute the average time to complete the broadcast (i.e., every process has delivered the message).
2. Don't remember :(
3. Don't remember :(

## Exercise 5

Consider a system composed by two disjoint set of processes for clients $(c_1, c_2, \ldots, c_n)$ and for servers $(s_1, s_2, \ldots, s_m)$. There is a set of resources $R = \{R_1, R_2, R_3\}$ that clients want to request. Clients communicate with servers using perfect point-to-point links. The servers receive the requests from clients and they need to do that, at every time $t$, the same resource is allocated to one client, and requests that can't be satisfied need to be stored for being satisfied later.

Clients can request multiple resources at the same time, but every resource can be allocated by just one client. Assume that

1. Clients and servers can crash
2. Clients and servers have access to a perfect failure detector
3. Servers can communicate between them using uniform reliable broadcast
4. Clients and servers communicate through perfect point-to-point links

Answer the questions:

1. Provide the implementation of the resource allocation algorithm (at least Client's code).
2. Dont' remember
3. Dont' remember (something on byzantine failure model)

## Exercise 1

Discuss blockchain from distributed system point of view, the distinction between public, private, permissioned and premissionless. In addition, explain how PoW (Proof-of-Work) mechanism works and why it creates branches in the blockchain data structure.

BLOCKCHAIN IS A DISTRIBUTED SYSTEM IN WHICH A NETWORK OF KNOTS MAINTAINS A SHARED AND IMMUTABLE REGISTER OF TRANSACTIONS. EACH NODE OF THE NETWORK HAS A COPY OF THE BLOCKCHAIN AND CONTRIBUTES TO ITS MAINTENANCE THROUGH CONSENSUS PROTOCOLS. THE BLOCKCHAIN IS DECENTRALIZED AND BYZANTINE FAULT TOLLERANCE.

BLOCKCHAINS ARE CLASSIFIED ON THE BASIS OF ACCESSIBILITY AND PARTECIPATION PERMITS:

- PUBLIC: ANYONE CAN READ BLOCKCHAINS DATA AND SUBMIT TRANSACTIONS.
- PRIVATE: LIMITED ACCESS FOR SAFETY AND PERFORMANCE REASONS.
- PERMISSIONED: CHECKS WHO CAN MANAGE DATA.
- PERMISSIONLESS: ANYONE CAN MANAGE DATA WITHOUT AUTHORIZATIONS.

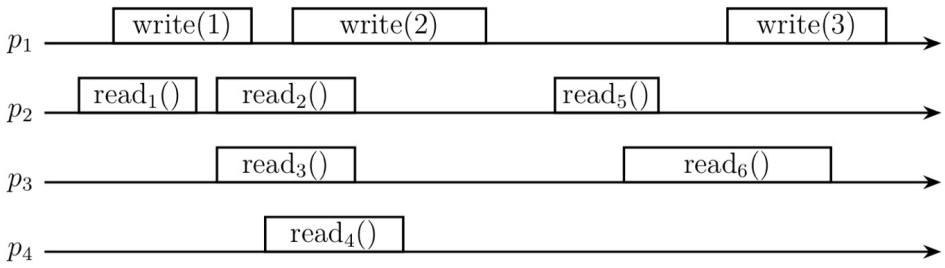
POW IS A SECURITY PROTOCOL USED IN BLOCKCHAINS TO GUARANTEE SECURITY AND TO PREVENT MALICIOUS ATTACKS, MAKING COMPUTATIONALLY DIFFICULT TO ADD A NEW BLOCK:

1. MINERS COMPETE TO SOLVE A COMPUTATIONAL PROBLEM, FINDING A nonce THAT, TOGETHER WITH THE BLOCK DATA, PRODUCES A HASH THAT SATISFIES A CONDITION.
2. THE FIRST MINER TO FIND A VALID HASH TRANSMITS THE BLOCK TO THE NET AND RECEIVES A REWARD.
3. THE OTHER NODES VERIFY THE VALIDITY OF THE SOLUTION AND ADD THE BLOCK TO THEIR BLOCKCHAIN.

OCCASIONALLY TWO OR MORE MINERS MAY FIND A VALID BLOCK SIMULTANEOUSLY (BRANCHES). IN THIS CASE THE NETWORK HAS TO CONVERGE TO THE LONGEST BRANCH.

## Exercise 2

Consider the execution depicted in the following figure.




Answer some true/false questions I don't remember, but basically answer the questions:

1. Define ALL values that can be returned by read operations (Rx) assuming the run refers to a regular register
2. Define ALL values that can be returned by read operations (Rx) assuming the run refers to an atomic register
3. Provide a sequence such that the execution is linearizable

1) $R_1()$: 0, 1
$R_2()$: 0, 1, 2
$R_3()$: 0, 1, 2
$R_4()$: 1, 2
$R_5()$: 2
$R_6()$: 2, 3

2) $R_1()$: 0, 1
$R_2()$: 0, 1, 2 IF $R_1()$:0
1, 2 IF $R_1()$:1
$R_3()$: 0, 1, 2 IF $R_1()$:0
1, 2 IF $R_1()$:1
$R_4()$: 1, 2
$R_5()$: 2
$R_6()$: 2, 3

3) $R_1()$:0 - $W_1()$: 1 - $R_2()$: 1 - $R_3()$: 1 - $W_2()$: 2 - $R_4()$:2 - $R_5()$:2 - $W_3()$:3 - $R_6()$:3

## Exercise 5

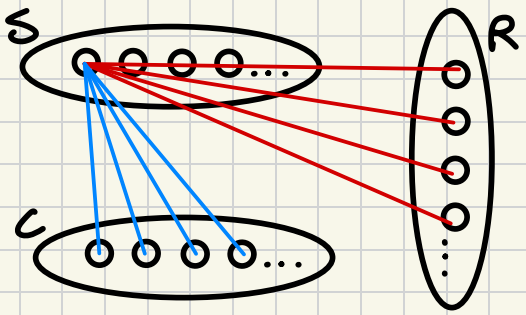
Consider a system composed by two disjoint set of processes for clients $(c_1, c_2, \ldots, c_n)$ and for servers $(s_1, s_2, \ldots, s_m)$. There is a set of resources $R = \{R_1, R_2, R_3\}$ that clients want to request. Clients communicate with servers using perfect point-to-point links. The servers receive the requests from clients and they need to do that, at every time $t$, the same resource is allocated to one client, and requests that can't be satisfied need to be stored for being satisfied later.

Clients can request multiple resources at the same time, but every resource can be allocated by just one client. Assume that

1. Clients and servers can crash
2. Clients and servers have access to a perfect failure detector
3. Servers can communicate between them using uniform reliable broadcast
4. Clients and servers communicate through perfect point-to-point links

Answer the questions:

1. Provide the implementation of the resource allocation algorithm (at least Client's code).




```
UPON EVENT < CL, INIT > DO
  OBTAINED.RESOURCES = ∅
  DESIRED.RESOURCES
  DENIED.RESOURCES = ∅

UPON EVENT < CL, REQUEST > DO
  FOR ALL r IN DESIRED.RESOURCES
    TRIGGER <PP2PL, SEND | [REQUEST, CLIENT_ID, r] > TO SERVERS
    DESIRED.RESOURCES = DESIRED.RESOURCES \ {r}

UPON EVENT < PP2PL, DELIVER | [GRANT, r] > FROM SERVER DO
  OBTAINED.RESOURCES = OBTAINED.RESOURCES ∪ {r}

UPON EVENT < PP2PL, DELIVER | [DENY, r] > FROM SERVER DO
  DENIED.RESOURCES = DENIED.RESOURCES ∪ {r}
  STARTTIMER (Δ)

UPON EVENT < TIMEOUT >
  FOR ALL r IN DENIED.RESOURCES
    TRIGGER <PP2PL, SEND | [REQUEST, CLIENT_ID, r] > TO SERVERS
    DENIED.RESOURCES = DENIED.RESOURCES \ {r}

UPON EVENT < CL, RELEASE | r > DO
  TRIGGER <PP2PL, SEND | [RELEASE, CLIENT_ID, r] > TO SERVERS
  OBTAINED.RESOURCES = OBTAINED.RESOURCES \ {r}
```