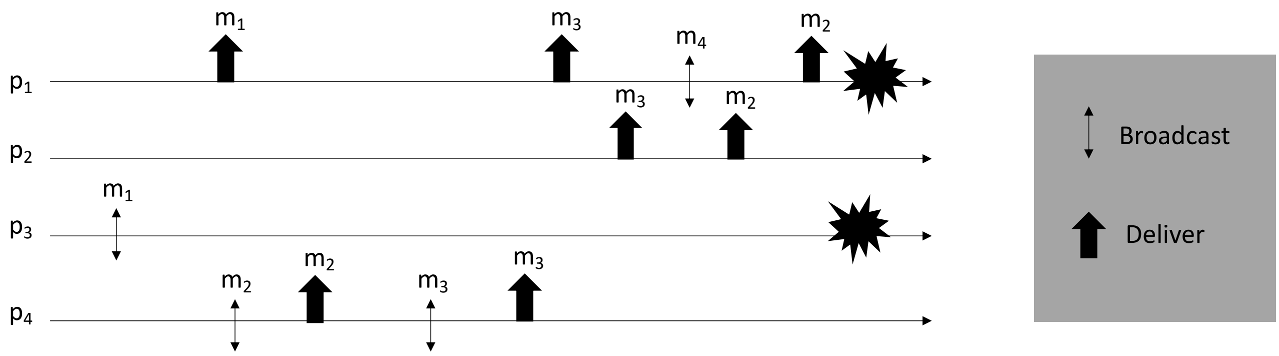
A

Dependable Distributed Systems (9 CFU)
24/01/2023
Exam A

Family Name________________________Name___________________Student ID_____________

Ex 1: Provide the specification of the perfect failure detector and explain how it can be implemented in a synchronous system. In addition, discuss what does it happen to the correctness of the oracle if we use fair-loss point to point links instead of perfect ones.

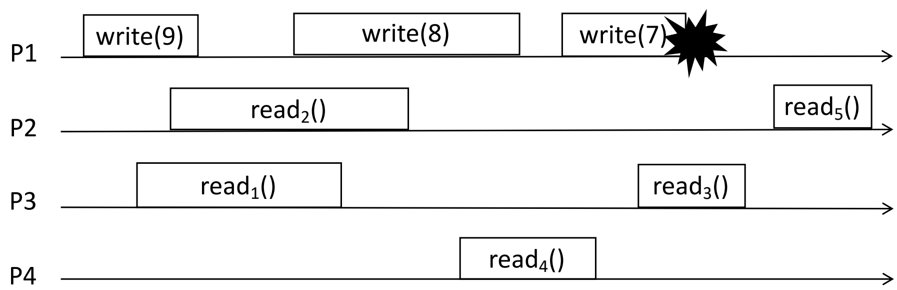
Ex 2: Let us consider the following execution history



Assess the truthfulness of every statement and provide a motivation for your answer:

1	The proposed run satisfies the Best Effort Broadcast specification	T	F
2	The strongest specification satisfied by the proposed run is Uniform Reliable Broadcast	T	F
3	If p_3 delivers m_4, then the resulting run satisfies the Regular Reliable Broadcast specification	T	F
4	If p_4 delivers m_4, then the resulting run satisfies the Regular Reliable Broadcast specification	T	F
5	If p_2 delivers m_1, then the resulting run satisfies the Uniform Reliable Broadcast specification	T	F
6	The run satisfies TO(NUA, WNUTO)	T	F
7	The run satisfies the FIFO Broadcast specification	T	F
8	Let us assume p_4 does not deliver m_2, then the resulting run satisfies TO(NUA, WUTO)	T	F
9	Let us assume p_2 crashes, then the resulting run satisfies TO(UA, WNUTO)	T	F
10	If p_1 does not deliver m_1, then the resulting run satisfies TO(UA, WUTO)	T	F

Ex 3: Consider the execution depicted in the following figure



Answer the following 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.

Ex 4: Let us assume a system working under a single workload component, the inter-arrival times and the service demands of such component are exponentially distributed, and the mean inter-arrival time is equal to 0.2 seconds.

1) Is it possible to estimate the expected response time if the mean service time is 0,05 sec? If yes, what is its value? If no, what is the minimum service time required to provide an estimation?
2) If the service is managed by two components that sequentially handle the requests, having service rates equal to $\mu\_1 = 8$ requests/sec and $\mu\_2 = 12$ requests/sec, is it possible to estimate the expected response time? If yes, what is its value? If no, explain why.
3) If every request must visit each component 2 times, namely, to be handled by the component twice, is it possible to estimate the expected response time? If yes, what is its value? If no, explain why.
4) Consider the setting presented at point 2. If you can replicate one component, splitting its load among two replicas, which component would you replicate to minimize the response time? What will be the expected response time?
5) Consider the configuration you provided at point 4 and assume that the MTBF of component 1 is 10 days, the MTBF of component 2 is 5 days and the repair rates of all components is 0.5 repair/day. Which setting guarantees higher availability between the setting at point 2 and the one you provided at point 4? Provide a motivation to your answer.

NOTE: It is sufficient to set the calculus to answer the questions.

Ex 5: Consider a distributed system composed of *n* processes. Processes are arranged in a ring topology. Every link of the ring is implemented through a perfect point-to-point channel. Each process in the system stores, in a local variable called `next`, the local identifier of the next process in the ring and maintains locally an integer value stored in a local variable `myVal`.
Let us consider the following specification:

Module

Name: sum Oracle

Events:

Request: ⟨ O_{sum}, get_sum ⟩: invoke the oracle to compute the sum of values stored by correct processes

Indication: ⟨ O_{sum}, sum_return | v ⟩: returns to the calling process the computed sum

Properties:

- ***Termination***: If a correct process invoke a get_sum() operation it eventually returns from the operation.
- ***Validity***: If a correct process p_i returns a value v, then v is the sum of values stored by correct processes.

Answer to the following questions:

1. Assuming that (i) processes do not fail, (ii) each process is univocally identified by a unique integer identifier, (iii) the system is asynchronous and (iv) every process p_i does not know the overall number of processes in the system, write the pseudo-code of a distributed algorithm implementing the sum Oracle.

2. Assuming that (i) processes do not fail, (ii) processes are anonymous (i.e., every process has the same integer identifier), (iii) the system is asynchronous and (iv) every process p_i does not know the overall number of processes in the system, discuss if it is possible to implement the sum Oracle. If so, describe shortly the algorithm (no pseudo-code is needed), otherwise provide the intuition of the impossibility.

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: ___________________________

Ex 1: Provide the specification of the perfect failure detector and explain how it can be implemented in a synchronous system. In addition, discuss what does it happen to the correctness of the oracle if we use fair-loss point to point links instead of perfect ones.

A FD IS A SW MODULE THAT PROVIDES INFORMATION ABOUT THE FAILURE STATUS OF PROCESSES. A PERFECT FD SATISFIES TWO PROPERTIES:

STRONG ACCURACY: CORRECT PROCESSES ARE NEVER ERRONEOUSLY SUSPECTED
STRONG COMPLETENESS. EVERY PROCESS THAT CRASHES IS PERMANENTLY SUSPECTED BY EVERY CORRECT PROCESSES.

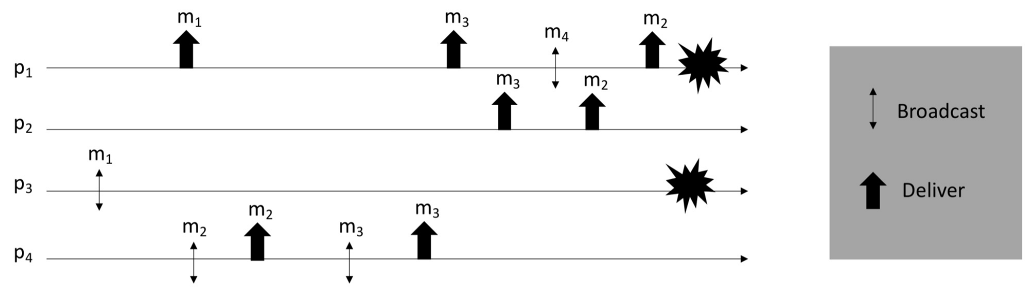
THESE PROPERTIES ENSURE THAT EACH FAILURE IS ALWAYS DETECTED AND THAT THERE ARE NO FALSE POSITIVES.

IN A SYNCHRONOUS SYSTEM, THE PFD CAN BE IMPLEMENTED BY EXPLOITING THE FACT THAT THERE ARE KNOW LIMITS ON THE COMMUNICATION AND PROCESSING TIME BETWEEN THE PROCESSES THIS MEANS THAT, FOR EACH PROCESS, IT'S POSSIBLE TO ESTABLISH A MAXIMUM INTERVAL WITHIN WHICH AN ANSWER MUST BE RECEIVE FROM ANOTHER PROCESS TO CONSIDER IT STILL ACTIVE.
THIS IMPLEMENTATION IS BASED ON A HEARTBEAT AND TIMEOUT MECHANISM: EACH PROCESS PERIODICALLY SENDS MESSAGE OF "VITALITY" (HRBT) TO OTHERS AND, IF IT DOESN'T RECEIVE A RESPONSE WITHIN A MAXIMUM PREDEFINED TIME, CAN CONCLUDE WITH CERTAINTY THAT THE REMOTE PROCESS IS FAULTY.

USING FAIR-LOSS P2PL, MESSAGES CAN BE OCCASIONALLY LOST:

STRONG ACCURACY PROBLEM: IF A PROCESS IS CORRECT BUT ITS HRBT IS LOST, ANOTHER PROCESS COULD ERRONEOUSLY SUSPECT IT TO BE FAULTY.
STRONG COMPLETENESS: IF P CRASHES AND THE MSG OF FAILURE LOST, HOWEVER P WILL NO RESPONDS WITH HEARTHBEAT REPLY.

Ex 2: Let us consider the following execution history



Assess the truthfulness of every statement and provide a motivation for your answer:

1	The proposed run satisfies the Best Effort Broadcast specification	T	F
2	The strongest specification satisfied by the proposed run is Uniform Reliable Broadcast	T	F
3	If p_3 delivers m_4, then the resulting run satisfies the Regular Reliable Broadcast specification	T	F
4	If p_4 delivers m_4, then the resulting run satisfies the Regular Reliable Broadcast specification	T	F
5	If p_2 delivers m_1, then the resulting run satisfies the Uniform Reliable Broadcast specification	T	F
6	The run satisfies TO(NUA, WNUTO)	T	F
7	The run satisfies the FIFO Broadcast specification	T	F
8	Let us assume p_4 does not deliver m_2, then the resulting run satisfies TO(NUA, WUTO)	T	F
9	Let us assume p_2 crashes, then the resulting run satisfies TO(UA, WNUTO)	T	F
10	If p_1 does not deliver m_1, then the resulting run satisfies TO(UA, WUTO)	T	F

1) P_2 AND P_4 SATISFY VALIDITY, NO DUPLICATION AND NO CREATION.

2) P_1 (FAULTY) DELIVERS m_1, WHICH IS NOT DELIVERED BY P_2 AND P_4 (CORRECT).

3) P_2 AND P_4 (CORRECT) SATISFIE AGREEMENT.

4) P_2 DOESN'T DELIVER m_4 (NO AGREEMENT).

5) P_4 DOESN'T DELIVER m_1 (NO UNIFORM AGREEMENT).

6) IT'S NOT TO BECAUSE P_2 AND P_4 HAVE DIFFERENT ORDERS.

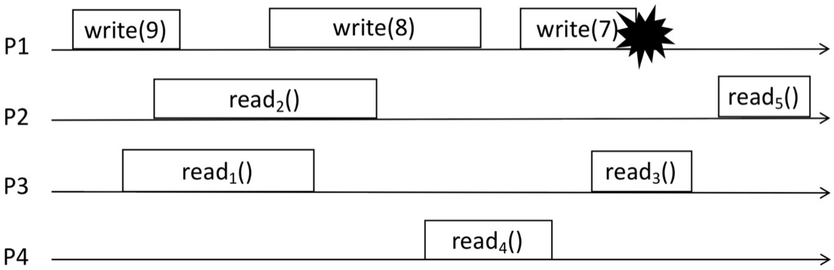
7) $m_2 \rightarrow m_3$ NOT SATISFIED BY P_2

8) P_2 AND P_4 HAVE DIFFERENT SET OF DELIVERY

9) P_4 DOESN'T DELIVER m_1 (NO UNIFORM AGREEMENT).

10) IT'S NOT TO BECAUSE P_2 AND P_4 HAVE DIFFERENT ORDERS.

Ex 3: Consider the execution depicted in the following figure



Answer the following 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.

1)
$R_1()$: 0, 9, 8
$R_2()$: 0, 9, 8
$R_3()$: 8, 7
$R_4()$: 9, 8, 7
$R_5()$: 8, 7

2)
$R_1()$: 0, 9, 8
$R_2()$: 0, 9, 8

$$R_4(): \begin{cases} 9,8,7 & \text{IF } R_2(): 0,9 \text{ AND } R_1(): 0,9 \\ 8,7 & \text{IF } R_2(): 8 \text{ AND } R_1(): 8 \end{cases}$$

$$R_3(): \begin{cases} 8,7 & \text{IF } R_4(): 9,8 \\ 7 & \text{IF } R_4(): 7 \end{cases}$$

$$R_5(): \begin{cases} 8,7 & \text{IF } R_3(): 8 \\ 7 & \text{IF } R_3(): 7 \end{cases}$$

Ex 5: Consider a distributed system composed of n processes. Processes are arranged in a ring topology. Every link of the ring is implemented through a perfect point-to-point channel. Each process in the system stores, in a local variable called `next`, the local identifier of the next process in the ring and maintains locally an integer value stored in a local variable `myVal`.
Let us consider the following specification:

Module

Name: sum Oracle

Events:

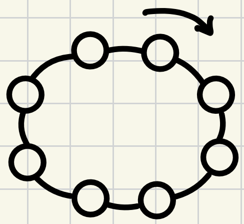
Request: ⟨ O_{sum}, get_sum ⟩: invoke the oracle to compute the sum of values stored by correct processes
Indication: ⟨ O_{sum}, sum_return | v ⟩: returns to the calling process the computed sum

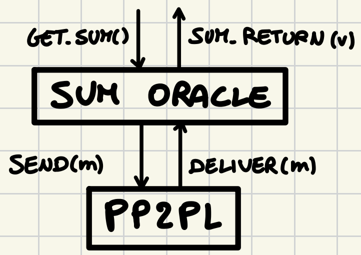
Properties:

- *Termination*: If a correct process invoke a get_sum() operation it eventually returns from the operation.
- *Validity*: If a correct process p_i returns a value v, then v is the sum of values stored by correct processes.

Answer to the following questions:

1. Assuming that (i) processes do not fail, (ii) each process is univocally identified by a unique integer identifier, (iii) the system is asynchronous and (iv) every process p_i does not know the overall number of processes in the system, write the pseudo-code of a distributed algorithm implementing the sum Oracle.
2. Assuming that (i) processes do not fail, (ii) processes are anonymous (i.e., every process has the same integer identifier), (iii) the system is asynchronous and (iv) every process p_i does not know the overall number of processes in the system, discuss if it is possible to implement the sum Oracle. If so, describe shortly the algorithm (no pseudo-code is needed), otherwise provide the intuition of the impossibility.





1) PROCESS DON'T FAIL, EACH P HAS AN ID, SYSTEM ASYNC, N UNKNOW

```
UPON EVENT <SO, INIT> DO
   NEXT := GET.NEXT()
   MYVAL

UPON EVENT <SO, GET.SUM()> DO
   TRIGGER <PP2PL, SEND | [MYVAL, SELF]> TO NEXT

UPON EVENT <PP2PL, DELIVER | [VAL, P]> DO
   IF SELF == P THEN        SIAMO TORNATI AL PRIMO
      TRIGGER <SO, SUM.RETURN | VAL>
   ELSE
      TRIGGER <PP2PL, SEND | [VAL + MYVAL, P]> TO NEXT
```

2) IMPOSSIBLE BECAUSE IN THE DELIVERY WE CHECK IF ALL PROCESSES SEND THEIR VALUE BY USING THEIR IDs. WITHOUT ID WE DON'T KNOW WHEN TO STOP.