A

FATTO

Dependable Distributed Systems (9 CFU)

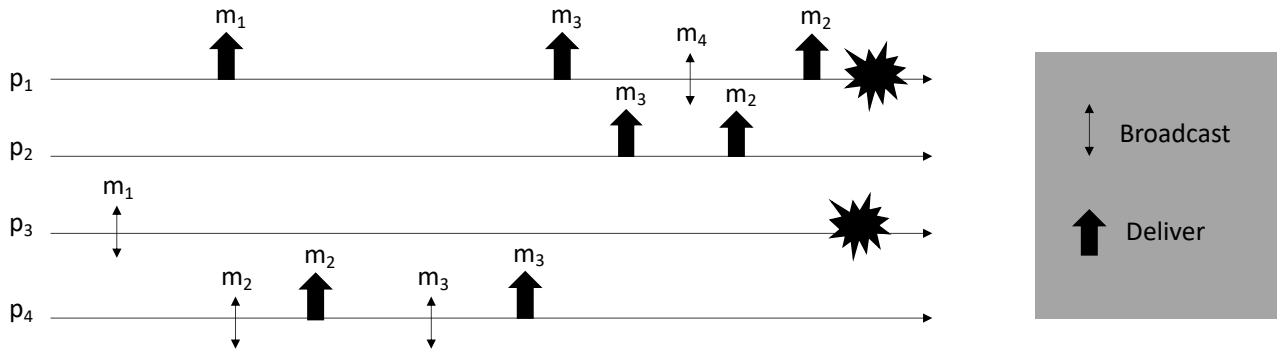
24/01/2023

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.

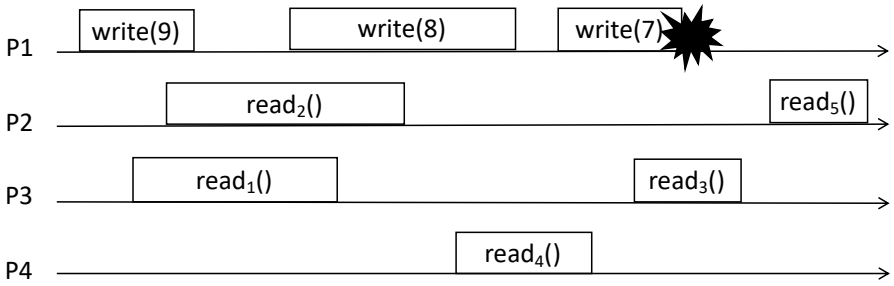
X **Ex 2:** Let us consider the following execution history



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

X **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:

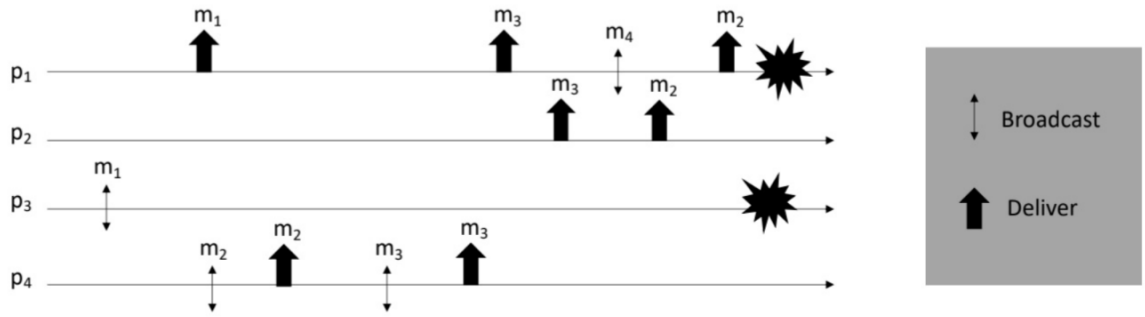
1. Assuming that (i) processes do not fail, (ii) each process is univocally identified by a unique integer identifier, (iii) the system is asynchronous and (iv) every process p_i does not know the overall number of processes in the system, write the pseudo-code of a distributed algorithm implementing 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 2: Let us consider the following execution history



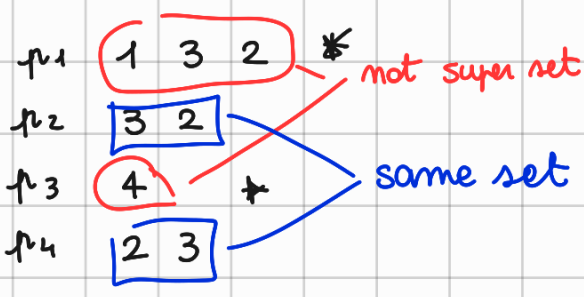
Assess the truthfulness of every statement and provide a motivation for your answer:

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~~

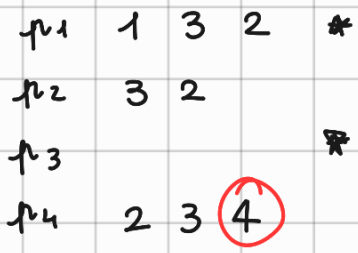
1) T, the correct processes (p_2, p_4) deliver the all the msgs broadcasted by the correct processes, that are m_2 m_3. (validity, no duplication, no creation)

2) F, because the set of the correct processes is not a super set of the faulty one. Indeed, corrects deliver m_3, m_2 and the faulty m_1, m_2, m_3. The uniform agreement is not satisfied.

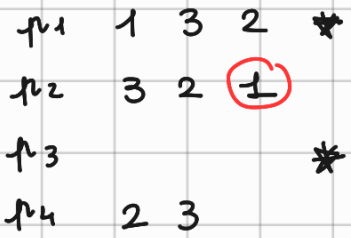
3) T, because the correct processes have the same set so the agreement is satisfied.



4) F, because the correct processes don't have the same set so the agreement is not satisfied.



5) F, because the set of the correct processes is not a super set of the faulty one. The uniform agreement is not satisfied.



6) F, the non uniform agreement is satisfied but not the order in the correct processes ($m_2 \rightarrow m_3$)

7) F, because FIFO impose $m_2 \rightarrow m_3$ and it is not respected in the correct process p_2

8) F, because the corrects have different set and the ordering is not satisfied in faulty, despite m_1 in p_1

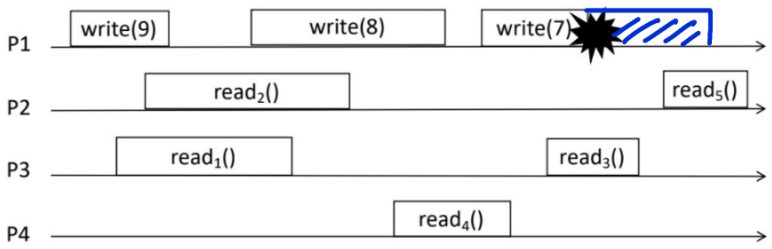
9) F, the agreement is not satisfied cause of m_1

p_1	1	3	2	*
p_2	3	2		*
p_3				*
p_4	2	3		

10) F, the UA is satisfied but the ordering in the correct not (p_2 $m_3 \to m_2$ and p_4 $m_2 \to m_3$)

p_1	3	2		*
p_2	3	2		
p_3				*
p_4	2	3		

Ex 3: Consider the execution depicted in the following figure



1) Regular

2) Atomic

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

2) Atomic: $R_1 = (0,9,8)$ } concurrent $R_4 = (9,8,7)\ \sigma\ (8,7)$
$R_2 = (0,9,8)$ $R_5 = (8,7)\ \sigma\ (7)$
$R_3 = (8,7)\ \sigma\ (7)$

if $R_1/R_2 = 0$ then $R_4 = (9,8,7)$
if $R_1/R_2 = 9$ then $R_4 = (9,8,7)$
if $R_1/R_2 = 8$ then $R_4 = (8,7)$

if $R_4 = 9$ then $R_3 = (8,7)$
if $R_4 = 8$ then $R_3 = (8,7)$
if $R_4 = 7$ then $R_3 = 7$

if $R_3 = 8$ then $R_5 = (8,7)$
if $R_3 = 7$ then $R_5 = 7$

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.

arrival time = 0.2 s

1) service time = 0.05 s

$\lambda = \frac{1}{0.2} = 5$ req/s $\qquad \mu = 20$ req/s

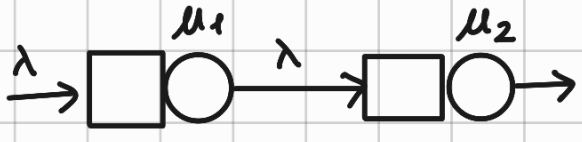
Stability condition $\lambda < \mu \quad 5 < 20$ ✓

Single workload component $R = \frac{1}{\mu - \lambda} = 0{,}06$ s

2) 2 components sequentially

$\mu_1 = 8$ req/s

$\mu_2 = 12$ req/s



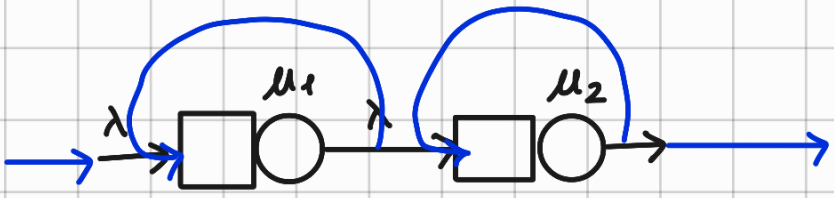
acts as bottleneck

$R_1 = \frac{1}{\mu_1 - \lambda} = 0{,}33$ s $\qquad R_2 = \frac{1}{\mu_2 - \lambda} = 0{,}14$ s

$\lambda < \mu_1$
$\lambda < \mu_2$

$R_{TOT} = 0{,}47\ s$

3) $\mu_1 = 8$ req/s
$\mu_2 = 12$ req/s
$\lambda = 5$ req/s



Nel caso di visite la formula cambia

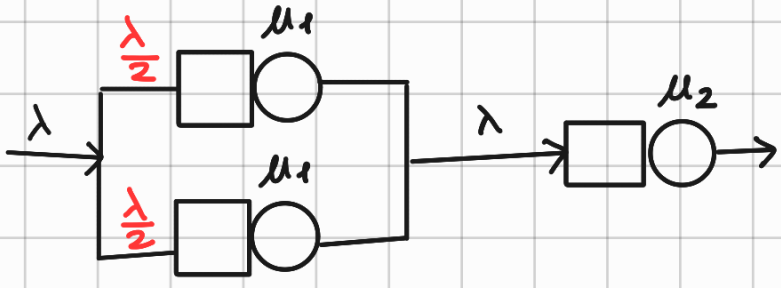
$R_{TOT} = 2\,(R_1 + R_2) = 0{,}94\ s$

4) The first component because have the higher utilization, that acts as a bottleneck

$R_2 = 0{,}14\ s$

$R_1 = K \cdot \dfrac{1}{K\mu_1 - \lambda}$

$= 2 \cdot \dfrac{1}{2 \cdot 8 - 5/2} = 0{,}18\ s$



in $R_1 \rightarrow \frac{\lambda}{2}$

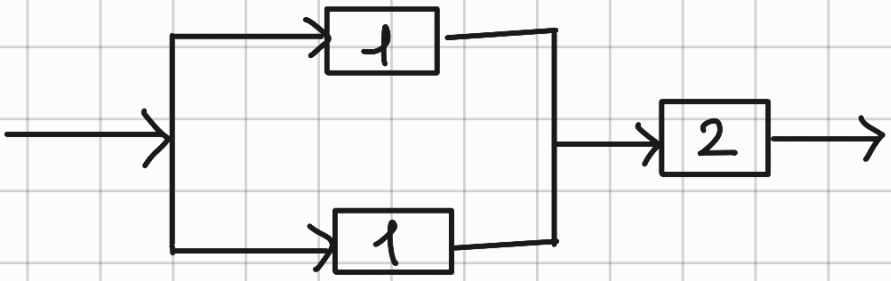
$R_T = R_1 + R_2 = 0{,}32\ s$

[SOLUZIONE: errata (non vedere)
$\mu_1 \cdot 2 = 16$ req/s $\quad R_{TOT} = 0{,}14 + \dfrac{1}{2 \cdot 16 - 5} = 0{,}17\ s$]

5) $MTBF_1 = 10$ days
$MTBF_2 = 5$ days

before failure

repair rates = 0,5 repair/days to repair



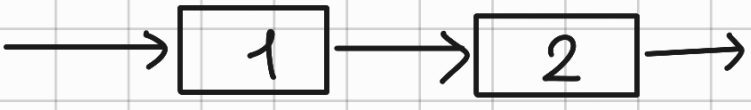
$$A_1 = \frac{MTBF_1}{MTBF_1 + MTTR} = \frac{10}{10 + 0,5} = 0,95$$

$A_2 = 0,90$

$$A_{parallel\,1} = 1 - \Pi(1 - A_1)$$

$$= 1 - (1 - 0,95)^2 = 0,9936$$

$A_{TOT} = \Pi A_i = 0,89$ POINT 4



$A_{TOT} = 0,85$ POINT 2

It is more available the solution at point 4 : $A_{P4} > A_{P2}$, because the A is the fraction of time that a component is operational and indicates a better system reliability.

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:

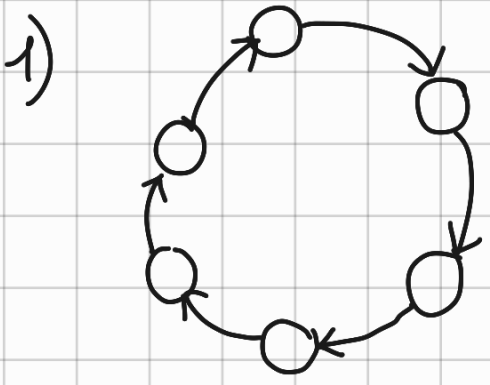
Module
Name: sum Oracle
Events:
Request: ⟨ O_{sum}, get_sum ⟩: invoke the oracle to compute the sum of values stored by correct processes
Indication: ⟨ O_{sum}, sum_return | v ⟩: returns to the calling process the computed sum

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.

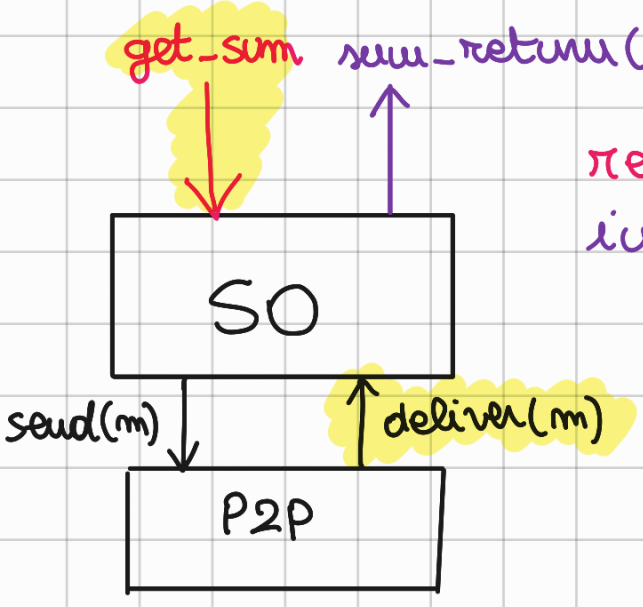
2. Assuming that (i) processes do not fail, (ii) processes are anonymous (i.e., every process has the same integer identifier), (iii) the system is asynchronous and (iv) every process p_i does not know the overall number of processes in the system, discuss if it is possible to implement the sum Oracle.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

1)



next = next process in the ring

myVal



request: get_sum
indication: sum_return(v)

upon event < so, init > do
    next
    myVal

upon event < so, get_sum > do
    trigger < p2p, p2p_send | [myVal, self] > to next

upon event < p2p, p2p_deliver | [myVal, p] > do
    if p == self do 1° in the ring
        trigger < so, sum_return, v >

    else
        val = val + myVal
        trigger < p2p, p2p_send | [val, p] > to next

2) Impossible. Because in the delivery we check if the value is equal to the first process and using the id is impossible to know who is the first / last having all the same id.