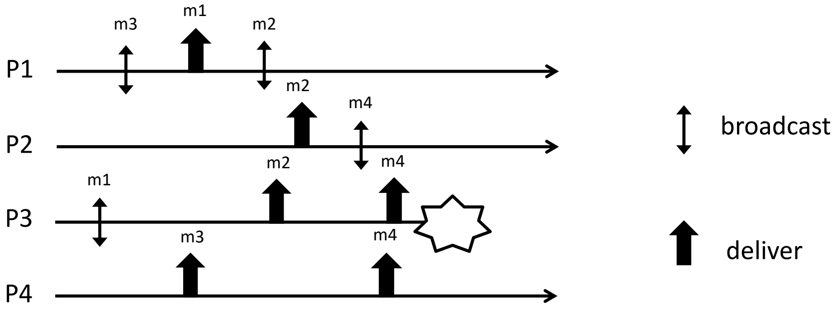
Dependable Distributed Systems
Master of Science in Engineering in Computer Science

AA 2024/2025

Week 7 – Exercises
November 13th, 2024

Ex 1: Consider the partial execution depicted in the Figure

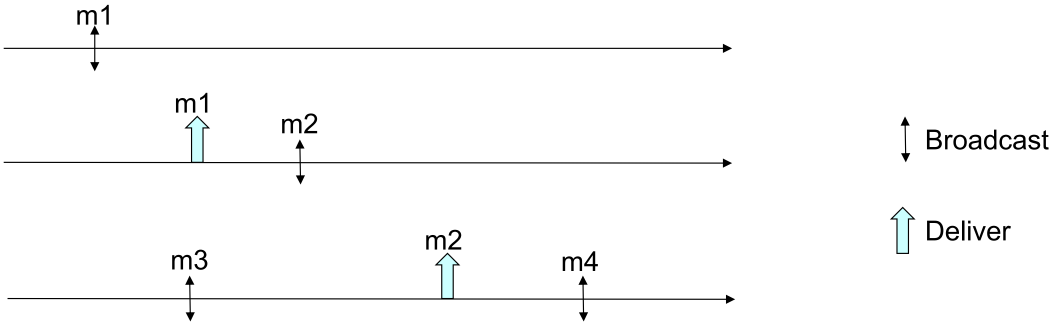


Answer to the following questions:

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: Given the partial execution in Figure, provide all the delivery sequences such that both total order and causal order are satisfied



Ex 3: Let us consider the following algorithm implementing a (1, N) atomic register in synchronous system.

```
13 upon event ⟨ onar, Init ⟩ do
14 (ts, val) := (0, ⊥);
15 correct := Π;
16 writeset := ∅;
17 readval := ⊥;
18 reading := FALSE;

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

21 upon event ⟨ onar, Read ⟩ do
22 reading := TRUE;
23 readval := val;
24 trigger ⟨ beb, Broadcast | [WRITE, ts, val] ⟩;

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

```
1  upon event ⟨ beb, Deliver | p, [WRITE, ts′, v′] ⟩ do
2  if ts′ > ts then
3          (ts, val) := (ts′ , v′ );
4  trigger ⟨ pl, Send | p, [ACK] ⟩;

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

7  upon correct ⊆ writeset do
8  writeset := ∅;
9  if reading = TRUE then
10         reading := FALSE;
11         trigger ⟨ onar, ReadReturn | readval ⟩;
12 else
26     trigger ⟨ onar, WriteReturn ⟩;
```

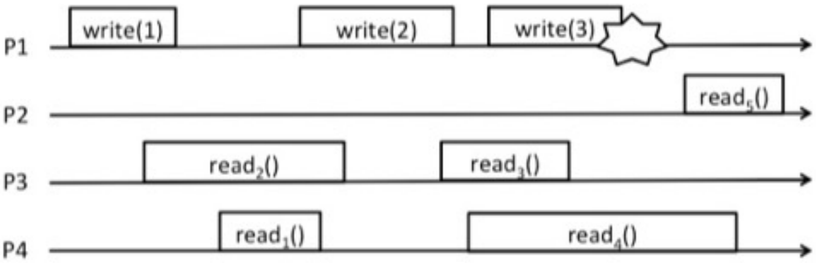
Assuming that messages are sent by using perfect point-to-point links and that the broadcast is best effort answer the following questions:

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 4: Consider a distributed system composed of n processes p1, p2,… pn connected through a ring topology. Initially, each process knows the list of correct processes and maintains locally a *next* variable where it stores the id of the following process in the ring.
Each process can communicate only with its next through FIFO perfect point-to-point channels (i.e. the process whose id is stored in the *next* variable).
Processes may fail by crash and each process has access to a perfect failure detector.

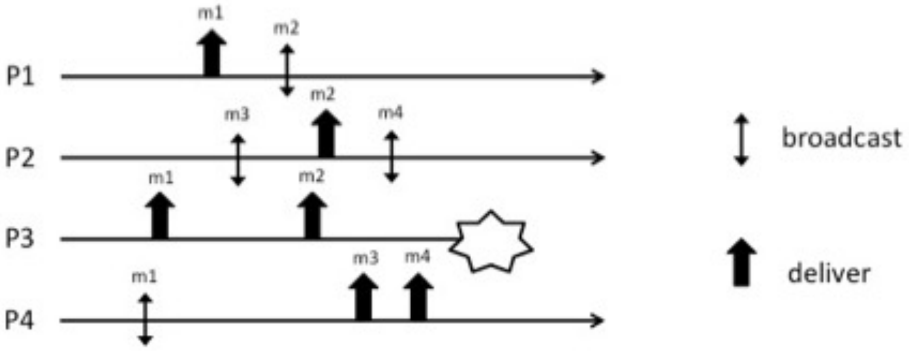
Write the pseudo-code of a distributed algorithm implementing a (1, N) atomic register.

Ex 5: 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.

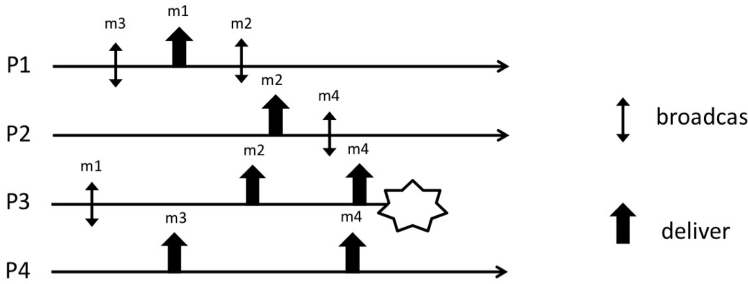
Ex 6: Let us consider the following partial execution



Answer the following points:

1. Provide the list al all the possible delivery sequences that satisfy both Total Order and Causal Order.
2. Complete the history (by adding the missing delivery events) to satisfy Total Order but not Causal Order.
3. Complete the history (by adding the missing delivery events) to satisfy FIFO Order but not Causal Order nor Total Order.

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.

1) CASUAL ORDER = FIFO ORDER + LOCAL ORDER

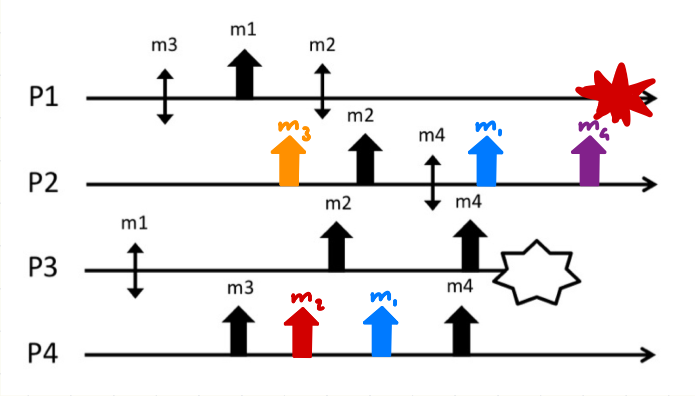
CASUAL ORDER: $m_1 \rightarrow m_2$ LOCAL ORDER
$m_2 \rightarrow m_4$ LOCAL ORDER
$m_3 \rightarrow m_2$ FIFO

TOTAL ORDER. $m_2 \rightarrow m_4$ P_3
$m_3 \rightarrow m_4$ P_4

POSSIBLE SEQUENCES:

m_1, m_3, m_2, m_4
m_3, m_1, m_2, m_4

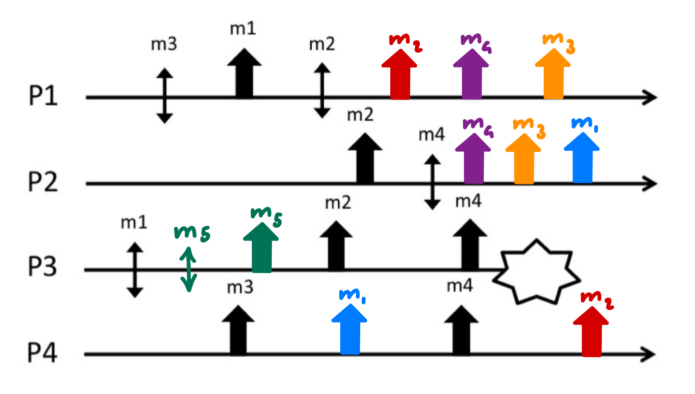
2) ORDER(O): ALL CORRECT PROCESSES DELIVER MESSAGES IN THE SAME ORDER.



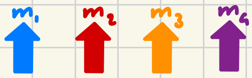
CASUAL ORDER: ~~$m_1 \rightarrow m_2$~~ LOCAL ORDER
~~$m_2 \rightarrow m_4$~~ LOCAL ORDER
$m_3 \rightarrow m_2$ FIFO

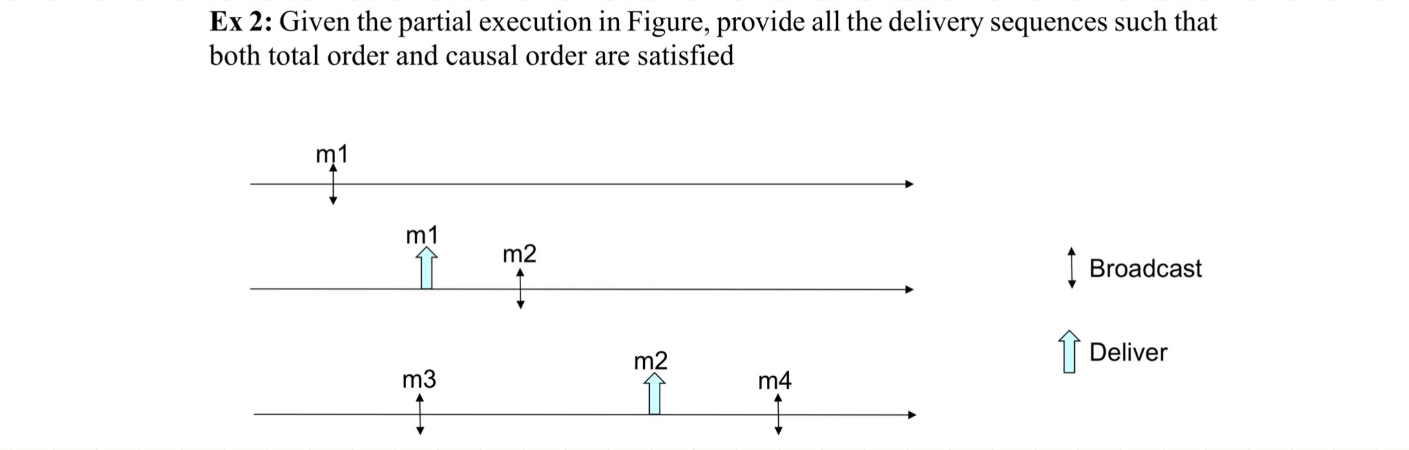
TOTAL ORDER. $m_2 \rightarrow m_4$ P_3
$m_3 \rightarrow m_4$ P_4

3) RB NOT URB, NOT TO



TOTAL ORDER. ~~$m_2 \rightarrow m_4$~~ P_3
~~$m_3 \rightarrow m_4$~~ P_4





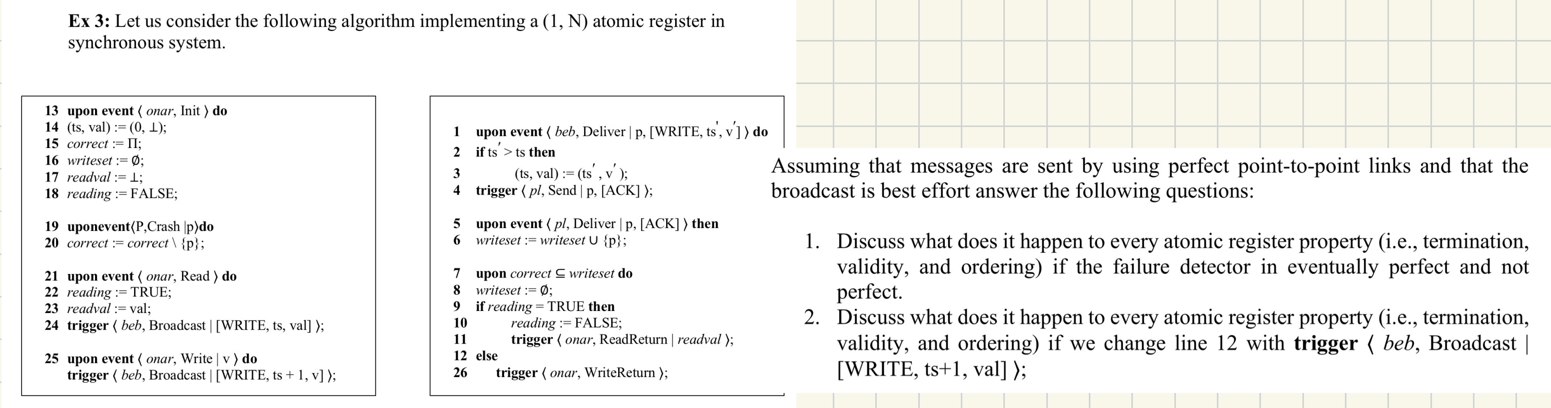
Ex 2: Given the partial execution in Figure, provide all the delivery sequences such that both total order and causal order are satisfied

CASUAL ORDER = FIFO ORDER + LOCAL ORDER

CASUAL ORDER: $m_3 \rightarrow m_4$ FIFO ORDER
$m_1 \rightarrow m_2$ LOCAL ORDER
$m_2 \rightarrow m_4$ LOCAL ORDER

TOTAL ORDER: NOTHING

POSSIBLE SEQUENCES: m_1 , m_2, m_3 , m_4
m_1 , m_3 , m_2, m_4
m_3, m_1 . m_2 , m_4



Ex 3: Let us consider the following algorithm implementing a (1, N) atomic register in synchronous system.

```
13 upon event ⟨ onar, Init ⟩ do
14 (ts, val) := (0, ⊥);
15 correct := Π;
16 writeset := ∅;
17 readval := ⊥;
18 reading := FALSE;

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

21 upon event ⟨ onar, Read ⟩ do
22 reading := TRUE;
23 readval := val;
24 trigger ⟨ beb, Broadcast | [WRITE, ts, val] ⟩;

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

```
1  upon event ⟨ beb, Deliver | p, [WRITE, ts', v'] ⟩ do
2  if ts' > ts then
3          (ts, val) := (ts' , v' );
4  trigger ⟨ pl, Send | p, [ACK] ⟩;

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

7  upon correct ⊆ writeset do
8  writeset := ∅;
9  if reading = TRUE then
10         reading := FALSE;
11         trigger ⟨ onar, ReadReturn | readval ⟩;
12 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] ⟩;

1) AN EVENTUALLY PFD $\Diamond P$ MAY INITIALLY MAKE MISTAKES IN REPORTING CORRECT PROCESSES AS FAILED, BUT ENSURES THAT IT WILL EVENTUALLY ONLY REPORT PROCESSES THAT ACTUALLY FAILED AND RECOGNIZE ALL CORRECT PROCESSES AS SUCH. SO:

TERMINATION: IT REQUIRES ALL READ AND WRITE OPERATIONS TO COMPLETE, EVEN IF THERE ARE FAILURES. WITH A $\Diamond P$, TERMINATION CAN BE DELAYED WHILE THE DETECTOR CORRECTS ITS INFORMATION. HOWEVER, AS THE DETECTOR CONVERGES TO CORRECTNESS, THE ALGORITHM STILL GUARANTEES LONG-TERM TERMINATION.

VALIDITY: IT ENSURES THAT THE READ VALUES WERE ACTUALLY WRITTEN THE ALGORITHM USES TIMESTAMP t_s TO COMPARE VALUES AND SELECT THE MOST RECENT ONE. VALIDITY REMAINS GUARANTEED, SINCE $\Diamond P$ DOESN'T AFFECT THE CONSISTENCY OF THE MESSAGES SENT.

ORDERING: IT REQUIRES THAT ALL READ OPERATIONS RETURN VALUES IN CONSISTENT TIME ORDER. EVEN WITH $\Diamond P$, THE ALGORITHM GUARANTEES THAT t_s AND VAL ARE UPDATED BASED ON CORRECTLY RECEIVED MESSAGES, PRESERVING ORDERING.

2) THIS CHANGE CAUSES THE VALUE VAL TO BE TRANSMITTED WITH AN INCREMENTED TIMESTAMP t_s+1 WITH EACH NEW OPERATION. SO:

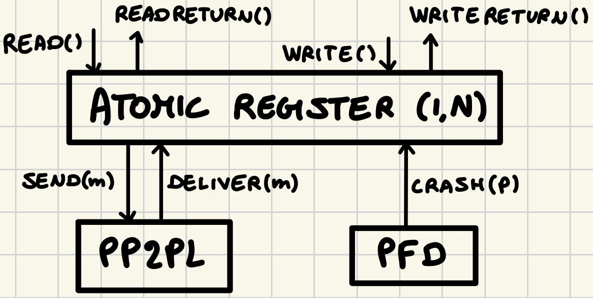
TERMINATION: IT'S NOT AFFECTED BY THE CHANGE, AS THE ALGORITHM CONTINUES TO BROADCAST MESSAGES TO ALL PROCESSES.

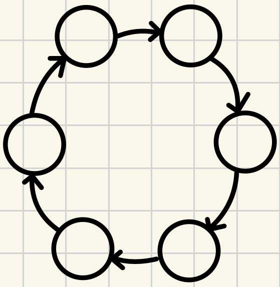
VALIDITY: NOT GUARANTEED. INCREMENTING t_s WITHOUT A NEW VALUE V CREATES INCONSISTENCY BETWEEN THE VALUE ASSOCIATED WITH THE TIMESTAMP AND THE ONE STORED. A PROCESS MIGHT READ A VALUE THAT WAS NEVER WRITTEN.

ORDERING: NOT GUARANTEED BECAUSE TIMESTAMPS NO LONGER UNIQUELY REPRESENT THE SEQUENCE OF ACTUAL WRITES.

Ex 4: Consider a distributed system composed of n processes p1, p2,… pn connected through a ring topology. Initially, each process knows the list of correct processes and maintains locally a *next* variable where it stores the id of the following process in the ring.
Each process can communicate only with its next through FIFO perfect point-to-point channels (i.e. the process whose id is stored in the *next* variable).
Processes may fail by crash and each process has access to a perfect failure detector.

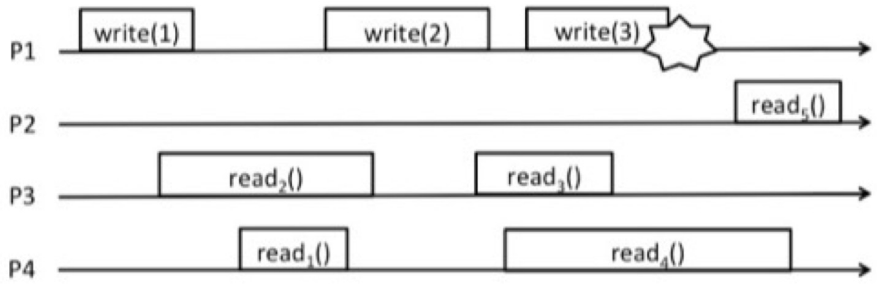
Write the pseudo-code of a distributed algorithm implementing a (1, N) atomic register.





```
INIT
    NEXT= P(i+1) MOD n
    CORRECT: $\Pi$
    READING: FALSE
    ACK = $\emptyset$
    READVAL: $\perp$

UPON EVENT < P, CRASH | p > DO
    CORRECT= CORRECT \ {p}
    NEXT = NEXT + 1

UPON EVENT < ONAR, WRITE | V >
    TRIGGER < PP2PL, SEND | [WRITE, V, ID] > TO NEXT

UPON EVENT < PP2PL, DELIVER | [WRITE, V', ID'] > DO
    IF ID' != SELF
        ACK= ACK U {SELF}
        TRIGGER < PP2PL, SEND | [ACK] > TO NEXT
        TRIGGER < PP2PL, SEND | [WRITE, V', ID'] > TO NEXT

UPON EVENT < PP2PL, DELIVER | [ACK] > DO
    IF CORRECT \ {SELF} $\subseteq$ ACK
        TRIGGER < ONAR, WRITERETURN >
    ELSE
        ACK = ACK U {SELF}
        TRIGGER < PP2PL, SEND | [ACK] > TO NEXT
```

Ex 5: Consider the execution depicted in the following figure and answer the questions

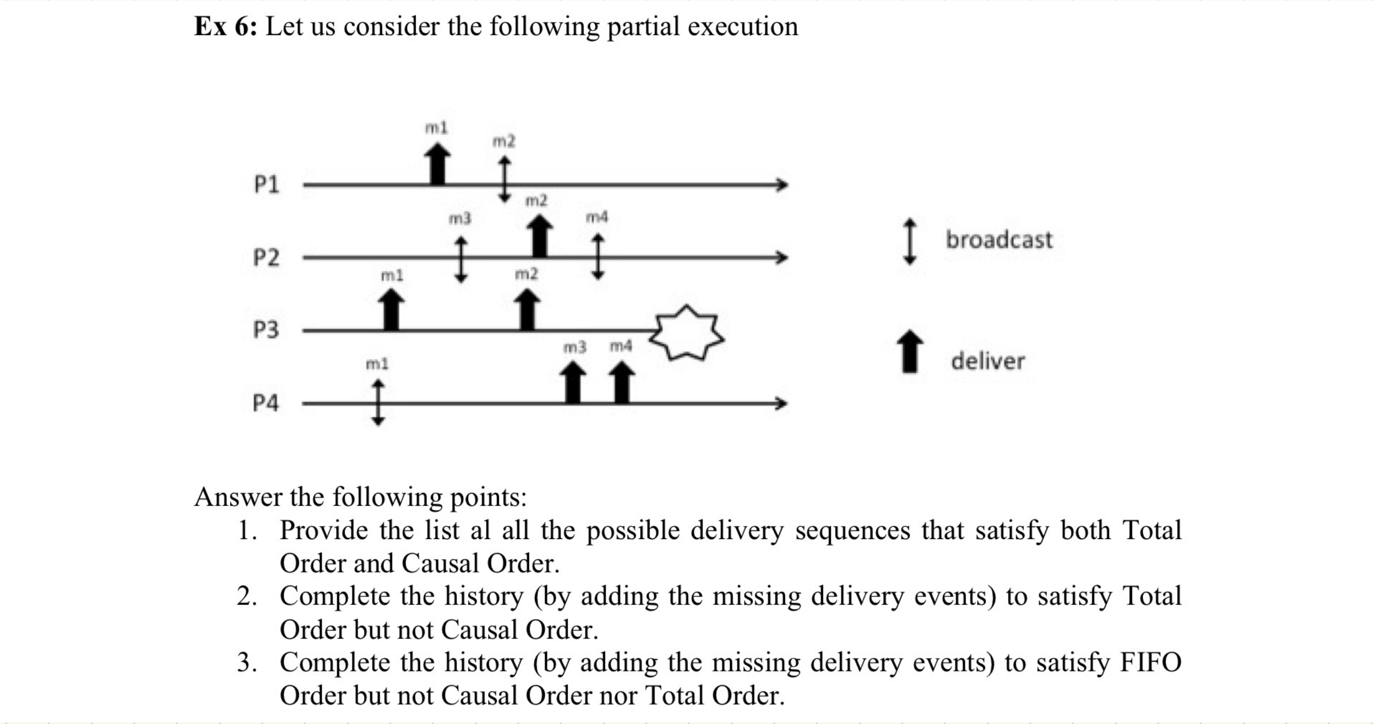


1. Define ALL the values that can be returned by read operations (Rx) assuming the run refers to a regular register.
2. Define ALL the values that can be returned by read operations (Rx) assuming the run refers to an atomic register.

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

2) $R_1()$: 1,2
$R_2()$: 0,1,2
$R_3()$. IF $R_2() \to 2$ THEN $R_3() \to 2,3$
ELSE IF $R_2() \to 1$ THEN $R_3() \to 1,2,3$

IF $R_1() \to 2$ THEN $R_3() \to 2,3$
ELSE IF $R_1() \to 1$ THEN $R_3() \to 1,2,3$
$R_4()$: 2,3
$R_5()$: IF $R_3() \to 3$ THEN $R_5() \to 3$
ELSE IF $R_3() \to 2$ THEN $R_5() \to 2,3$



Ex 6: Let us consider the following partial execution

Answer the following points:

1. Provide the list al all the possible delivery sequences that satisfy both Total Order and Causal Order.
2. Complete the history (by adding the missing delivery events) to satisfy Total Order but not Causal Order.
3. Complete the history (by adding the missing delivery events) to satisfy FIFO Order but not Causal Order nor Total Order.

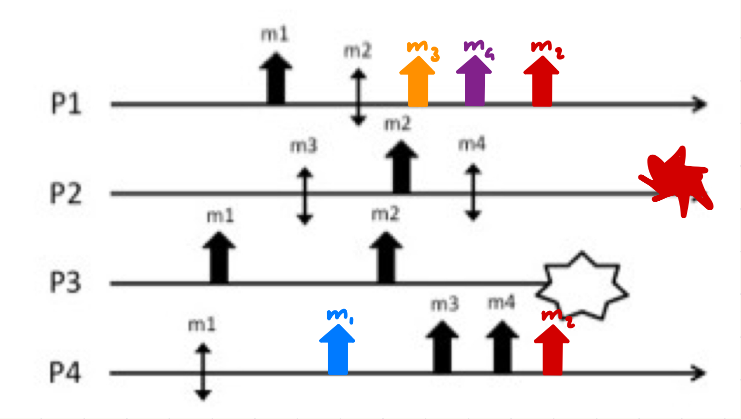
1) CASUAL ORDER = FIFO ORDER + LOCAL ORDER.

CASUAL ORDER. $m_1 \rightarrow m_2$ LOCAL ORDER
$m_2 \rightarrow m_4$ LOCAL ORDER
$m_3 \rightarrow m_4$ FIFO ORDER

TOTAL ORDER: $m_1 \rightarrow m_2$
$m_3 \rightarrow m_4$

POSSIBLE SEQUENCES:
m_1, m_2, m_3, m_4
m_1, m_3, m_2, m_4
m_3, m_1, m_2, m_4

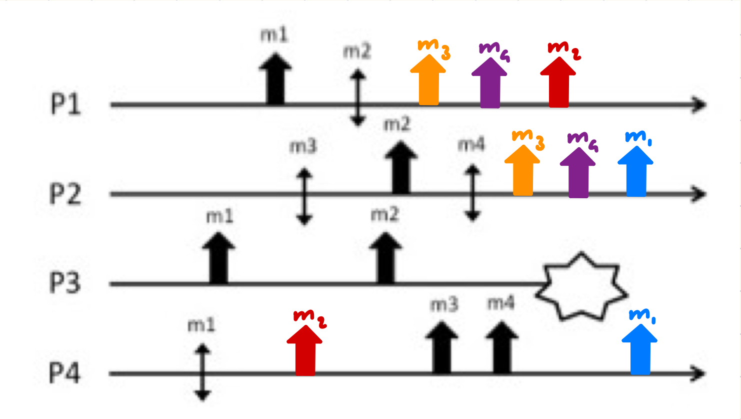
2) TO: ALL CORRECT PROCESSES DELIVER MESSAGES IN THE SAME ORDER



TOTAL ORDER: $m_1 \rightarrow m_2$
$m_3 \rightarrow m_4$

CASUAL ORDER. $m_1 \cancel{\rightarrow m_2}$ LOCAL ORDER
$m_2 \cancel{\rightarrow m_4}$ LOCAL ORDER
$m_3 \rightarrow m_4$ FIFO ORDER

3)



$m_3 \rightarrow m_4$ FIFO ORDER

$m_1 \cancel{\rightarrow m_2}$ LOCAL ORDER
$m_2 \cancel{\rightarrow m_4}$ LOCAL ORDER

m_1 m_2 m_3 m_4