

Distributed Systems

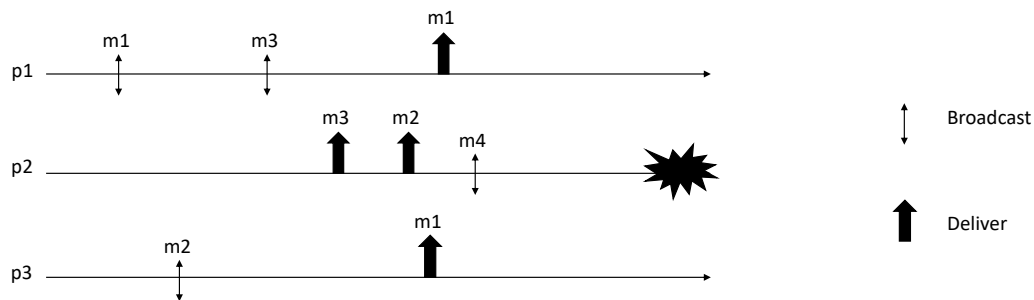
29/01/2021

Exam A

Family Name _____ Name _____ Student ID _____

Ex 1: Provide the specification of the (1, N) Regular Register and describe the majority voting algorithm discussed during the lectures.

Ex 2: Consider the message pattern shown in the Figure

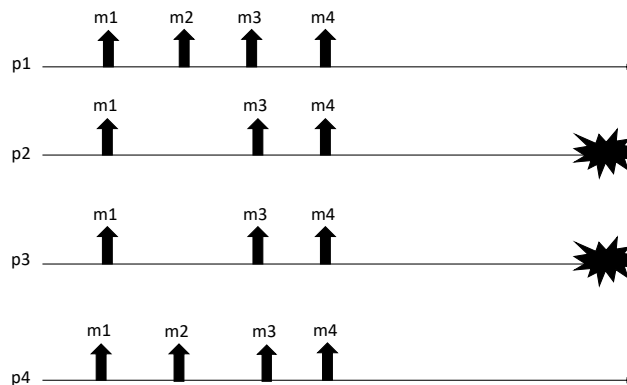


Answer to the following questions:

1. Complete the partial execution in order to obtain a run satisfying Uniform Reliable Broadcast
2. Complete the partial execution in order to obtain a run satisfying Regular Reliable Broadcast but not Uniform Reliable Broadcast
3. Complete the partial execution in order to obtain a run satisfying Best Effort Broadcast but not Regular Reliable Broadcast
4. List **ALL** the possible sequences satisfying both causal order and total order

NOTE: To solve the exercise you can just add deliveries of messages and new broadcast (if needed)

Ex 3: Consider the execution depicted in the Figure



Answer to the following questions:

1. Which is the strongest Total Order specification satisfied by the proposed run? Provide your answer by specifying both the agreement and the ordering property.
2. Modify the run in order to obtain an execution satisfying TO(UA, WNUTO) but not TO(UA, WUTO)

3. Modify the run in order to obtain an execution satisfying TO(NUA, SUTO) but not TO(UA, SUTO).

NOTE: To solve the exercise you can just add deliveries of messages.

Ex 4: Let us consider a distributed system composed of N processes executing the algorithm reported in figure

Algorithm 3.12: Broadcast with Sequence Number

Implements:

FIFOReliableBroadcast, **instance** *frb*.

Uses:

ReliableBroadcast, **instance** *rb*.

upon event $\langle frb, Init \rangle$ **do**

lsn := 0;
pending := \emptyset ;
next := $[1]^N$;

upon event $\langle frb, Broadcast \mid m \rangle$ **do**

lsn := *lsn* + 1;
trigger $\langle rb, Broadcast \mid [DATA, self, m, lsn] \rangle$;

upon event $\langle rb, Deliver \mid p, [DATA, s, m, sn] \rangle$ **do**

pending := *pending* $\cup \{(s, m, sn)\}$;
while exists $(s, m', sn') \in pending$ such that $sn' = next[s]$ **do**
next[*s*] := *next*[*s*] + 1;
pending := *pending* $\setminus \{(s, m', sn')\}$;
trigger $\langle frb, Deliver \mid s, m' \rangle$;

Let us assume that

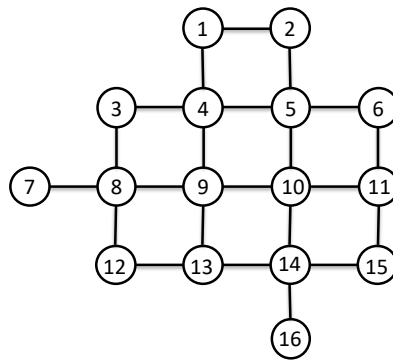
1. up to f processes may be Byzantine faulty and
2. A Byzantine process is not able to compromise the underline Reliable Broadcast primitive (i.e., when a Byzantine process sends a message through the *rbBroadcast* interface, the message will be reliably delivered to every correct process).

For each of the following properties, discuss if it can be guaranteed when $f=1$ and motivate your answer (also by using examples)

- *Validity*: If a correct process p broadcasts a message m , then p eventually delivers m .
- *No duplication*: No message is delivered more than once.
- *No creation*: If a process delivers a message m with sender s , then m was previously broadcast by process s .
- *Agreement*: If a message m is delivered by some correct process, then m is eventually delivered by every correct process.
- *FIFO delivery*: If some process broadcasts message m_1 before it broadcasts message m_2 , then no correct process delivers m_2 unless it has already delivered m_1 .

Ex 5: Consider a distributed system composed by n processes each one having a unique identifier. Processes communicate by exchanging messages through perfect point-to-point links and are connected through a grid (i.e., each process p_i can exchange messages only with processes located at *nord*, *sud*, *east* and *west* when they exist).

An example of such network is provided in the following figure:



Processes are not going to fail, and they initially know only the number of processes in the system N and the identifiers of their neighbors.

Processes in the system must agree on a color assignment satisfying the following specification:

Module

Name: Color assignment

Events:

Request: $\langle \text{ca}, \text{Propose} \mid c \rangle$: Proposes a color to be adopted.

Indication: $\langle \text{ca}, \text{Decide} \mid c \rangle$: Outputs a decided color to be adopted by the process

Properties:

Termination: Every process eventually decides a color.

Validity: If a process decides a color c , then c was proposed by some process.

Integrity: No process decides twice.

Fairness: If a color c is proposed by a process p_i , c will be eventually decided by some process p_j

Color Balance: Let C be the set of proposed colors, every $c_i \in C$ is decided by at most $k = N/|C|$ processes.

Write the pseudo-code of an algorithm implementing the Color assignment primitive.

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 $(1, N)$ Regular Register and describe the majority voting algorithm discussed during the lectures.

A REGULAR REGISTER $(1, N)$ IS A STRUCTURE SHARED BY A WRITER PROCESS AND N READER PROCESSES. ITS MAIN PROPERTIES ARE:

TERMINATION: IF A CORRECT PROCESS INVOKE A READ OR WRITE OPERATION, THE OPERATION WILL COMPLETE SUCCESSFULLY.

VALIDITY: A READING CAN RETURN THE VALUE OF THE LAST COMPLETED WRITE OPERATION, OR A VALUE BEING WRITTEN IF THE READING IS CONCURRENT WITH THE WRITING.

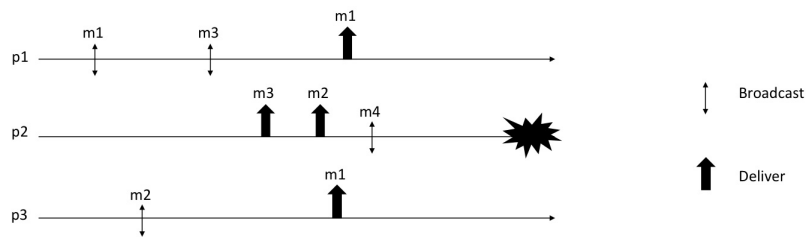
THE MAJORITY VOTING ALG IS USED TO IMPLEMENT A REGULAR REGISTER $(1, N)$ IN A FAIL-SILENT MODEL, WHERE PROCESSES CAN CRASH BUT CRASH CANNOT BE DETECTED.

THE WRITER SENDS $\langle v, \tau_s \rangle$ TO ALL PROCESS VIA bcb. EACH PROCESS RECEIVES THE MESSAGE AND UPDATES ITS LOCAL COPY OF THE REGISTRY IF THE τ_s IS GREATER THAN THE CURRENT ONE. THE WRITER WAITS FOR ACKS FROM A MAJORITY OF PROCESSES TO COMPLETE THE OPERATION.

THE READER SENDS READING REQUESTS TO ALL PROCESSES, WHICH RESPOND WITH $\langle v, \tau_s \rangle$ OF THEIR LOCAL COPY. THE READER COLLECTS THE ANSWERS AND SELECTS THE VALUE WITH THE HIGHEST τ_s .

THE MAJORITY VOTING ALG GUARANTEES TERMINATION AND VALIDITY BY EXPLOITING UNIQUE QUORUM AND TIMESTAMP TO COORDINATE READINGS AND SCRIPTURES.

Ex 2: Consider the message pattern shown in the Figure

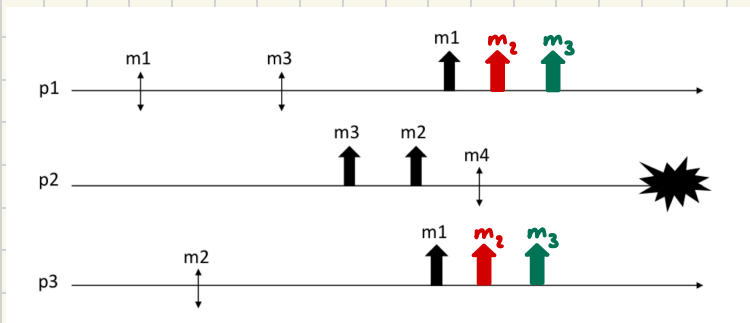


Answer to the following questions:

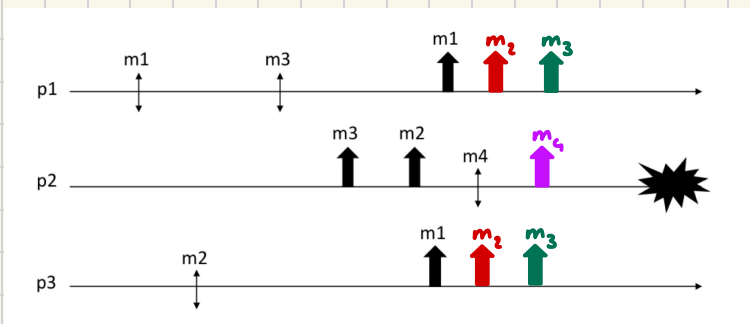
1. Complete the partial execution in order to obtain a run satisfying Uniform Reliable Broadcast
2. Complete the partial execution in order to obtain a run satisfying Regular Reliable Broadcast but not Uniform Reliable Broadcast
3. Complete the partial execution in order to obtain a run satisfying Best Effort Broadcast but not Regular Reliable Broadcast
4. List ALL the possible sequences satisfying both causal order and total order

NOTE: To solve the exercise you can just add deliveries of messages and new broadcast (if needed)

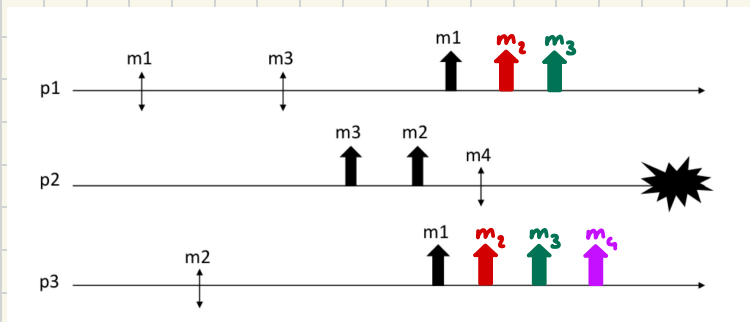
1) URB



2) RB BUT NOT URB



3) BEB BUT NOT RB



4) CASUAL ORDER: FIFO ORDER + LOCAL ORDER

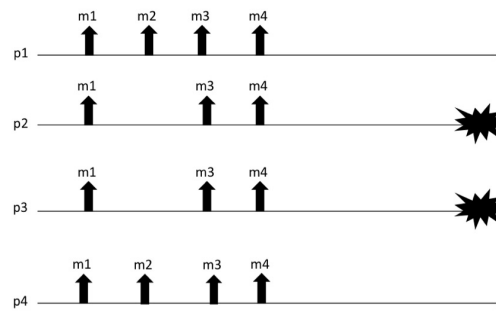
$m_1 \rightarrow m_3$ FIFO
 $m_2 \rightarrow m_4$ LOCAL

$m_3 \rightarrow m_2$ TO

POSSIBLE SEQUENCES:

m_1, m_3, m_2, m_4

Ex 3: Consider the execution depicted in the Figure



Answer to the following questions:

1. Which is the strongest Total Order specification satisfied by the proposed run? Provide your answer by specifying both the agreement and the ordering property.
2. Modify the run in order to obtain an execution satisfying TO(UA, WNUTO) but not TO(UA, WUTO).
3. Modify the run in order to obtain an execution satisfying TO(NUA, SUTO) but not TO(UA, SUTO).

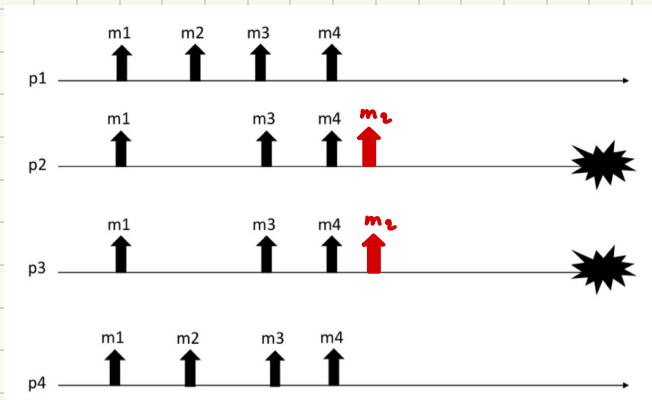
NOTE: To solve the exercise you can just add deliveries of messages.

1) TO (UA, WUTO)

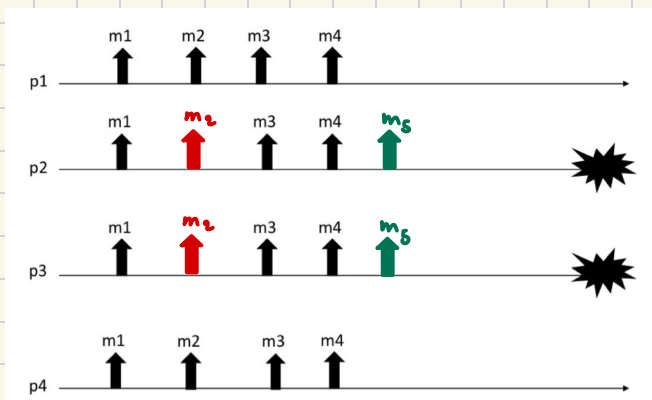
UA: BECAUSE MESSAGES DELIVERED BY FAULTY PROCESSES ARE DELIVERED ALSO BY CORRECT PROCESSES.

WUTO: IF p AND q BOTH DELIVER m AND m' , p DELIVERS m BEFORE m' ONLY IF q DELIVERS m BEFORE m' .

2) TO (UA, WNUTO) BUT NOT TO (UA, WUTO)



3) TO (NUA, SUTO) BUT NOT TO (UA, SUTO)



Ex 4: Let us consider a distributed system composed of N processes executing the algorithm reported in figure

```
Algorithm 3.12: Broadcast with Sequence Number
Implements:
    FIFOReliableBroadcast, instance frb.

Uses:
    ReliableBroadcast, instance rb.

upon event (frb, Init) do
    lsn := 0;
    pending :=  $\emptyset$ ;
    next :=  $[1]^N$ ;

upon event (frb, Broadcast | m) do
    lsn := lsn + 1;
    trigger (rb, Broadcast | [DATA, self, m, lsn]);

upon event (rb, Deliver | p, [DATA, s, m, sn]) do
    pending := pending  $\cup \{(s, m, sn)\}$ ;
    while exists (s, m', sn')  $\in$  pending such that sn' = next[s] do
        next[s] := next[s] + 1;
        pending := pending  $\setminus \{(s, m', sn')\}$ ;
    trigger (frb, Deliver | s, m');
```

Let us assume that

1. up to f processes may be Byzantine faulty and
2. A Byzantine process is not able to compromise the underline Reliable Broadcast primitive (i.e., when a Byzantine process sends a message through the *rbBroadcast* interface, the message will be reliably delivered to every correct process).

For each of the following properties, discuss if it can be guaranteed when $f=1$ and motivate your answer (also by using examples)

- *Validity*: If a correct process p broadcasts a message m , then p eventually delivers m .
- *No duplication*: No message is delivered more than once.
- *No creation*: If a process delivers a message m with sender s , then m was previously broadcast by process s .
- *Agreement*: If a message m is delivered by some correct process, then m is eventually delivered by every correct process.
- *FIFO delivery*: If some process broadcasts message m_1 before it broadcasts message m_2 , then no correct process delivers m_2 unless it has already delivered m_1 .

VALIDITY: SATISFIED THE PROTOCOL WARANTEES THAT, WHEN A CORRECT PROCESS TRANSHITS A MESSAGE m , IT'S SENT VIA RB, AND A BYZANTINE PROCESS, IN THIS CASE, IS NOT ABLE TO COMPROMISE RB.

NO DUPLICATION: SATISFIED m IS STORED IN PENDING, DELIVERED WHEN $\mathcal{I}_s = \text{NEXT}[s]$, AND REMOVED FROM PENDING.

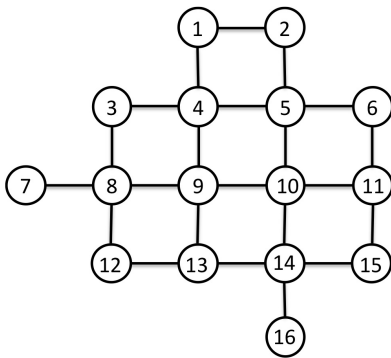
NO CREATION: NOT SATISFIED A BYZANTINE COULD TRANSMIT FALSE MESSAGES WITH THE NAME OF ANOTHER PROCESS, IF THERE ARE NO AUTHENTICATIONS IN THE MESSAGES.

AGREEMENT: SATISFIED USING RB, A TRANSMITTED MESSAGE IS RECEIVED BY ALL CORRECT PROCESSES, ENSURING THAT EACH CORRECT PROCESS WILL DELIVER THE MESSAGE IF ONE HAS ALREADY DONE.

FIFO: SATISFIED A BYZANTINE MAY NOT FOLLOW THE FIFO ORDER, HOWEVER THE CORRECT PROCESSES USE THE *lsn* FIELD TO ENSURE THAT EACH PROECESS TRANSHITS ITS MESSAGES IN INCREASING ORDER AND THAT THE MESSAGES ARE ONLY DELIVERED IF THEIR SEQUENCE NUMBER IS EXACTLY THE EXPECTED ONE.

Ex 5: Consider a distributed system composed by n processes each one having a unique identifier. Processes communicate by exchanging messages through perfect point-to-point links and are connected through a grid (i.e., each process p_i can exchange messages only with processes located at *nord*, *sud*, *east* and *west* when they exist).

An example of such network is provided in the following figure:



Processes are not going to fail, and they initially know only the number of processes in the system N and the identifiers of their neighbors.

Processes in the system must agree on a color assignment satisfying the following specification:

Module

Name: Color assignment

Events:

Request: $\langle ca, \text{Propose} \mid c \rangle$: Proposes a color to be adopted.

Indication: $\langle ca, \text{Decide} \mid c \rangle$: Outputs a decided color to be adopted by the process

Properties:

Termination: Every process eventually decides a color.

Validity: If a process decides a color c , then c was proposed by some process.

Integrity: No process decides twice.

Fairness: If a color c is proposed by a process p_i , c will be eventually decided by some process p_j .

Color Balance: Let C be the set of proposed colors, every $c_i \in C$ is decided by at most $k = N/|C|$ processes.

Write the pseudo-code of an algorithm implementing the Color assignment primitive.

UPON EVENT $\langle ca, \text{INIT} \rangle$ DO

$N = \Pi$

$\text{PROPOSAL.SET} = \emptyset$

$\text{DECISION} = \perp$

$\text{NEIGHBOURS} = \{N, S, E, W\}$

UPON EVENT $\langle ca, \text{PROPOSE} \mid c \rangle$ DO

$\text{PROPOSAL} = c$

$\text{PROPOSAL.SET} = \text{PROPOSAL.SET} \cup \{c\}$

FORALL p_j **IN** NEIGHBOURS DO

$\text{TRIGGER} \langle p2p, \text{SEND}(\text{PROPOSAL}, \text{PROPOSAL.SET}, p_i) \rangle$ **TO** p_j

UPON EVENT $\langle p2p, \text{DELIVER}(\text{PROPOSAL}, \text{PROPOSAL.SET}, p_i) \rangle$ **FROM** p_k

IF $N/|\text{PROPOSAL.SET}| \leq |\text{PROPOSAL.SET}|$ **AND** $\text{DECISION} = \perp$ **DO**

$\text{COLOR} = \text{LIGHTER.COLOR}(\text{PROPOSAL.SET})$

$\text{DECISION} = \text{COLOR}$

$\text{TRIGGER} \langle ca, \text{DECIDE} \mid \text{DECISION} \rangle$

ELSE

$\text{PROPOSAL} = c$

$\text{PROPOSAL.SET} = \text{PROPOSAL.SET} \cup \{c\}$

FORALL p_j **IN** NEIGHBOURS **DO**

$\text{TRIGGER} \langle p2p, \text{SEND}(\text{PROPOSAL}, \text{PROPOSAL.SET}, p_i) \rangle$ **TO** p_j

