Distributed Systems
Master of Science in Engineering in Computer Science

AA 2020/2021

LECTURE 12: TOTAL ORDER BROADCAST

System model

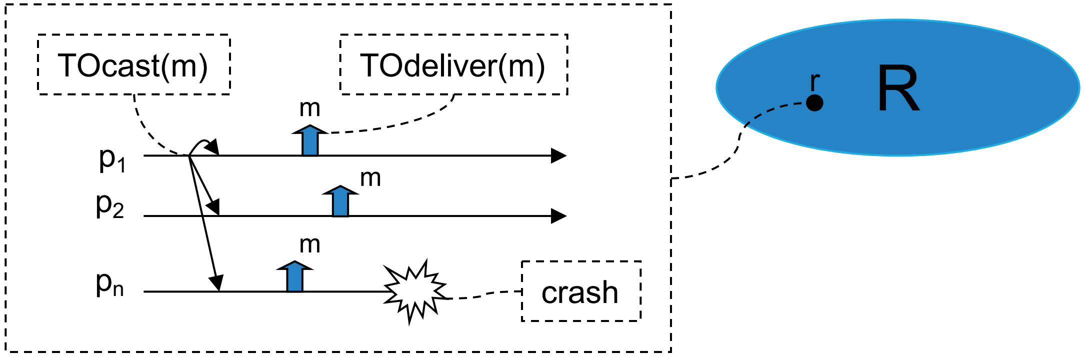
Static set of processes Π = {p_1 … p_n}

Message passing over perfect channels (message exchanging between correct processes is reliable)

Asynchronous

Crash fault model for processes

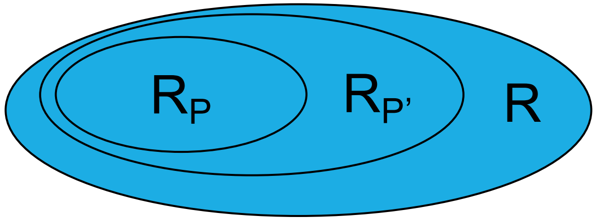
We characterize the system in terms of its possible runs R



A few notation

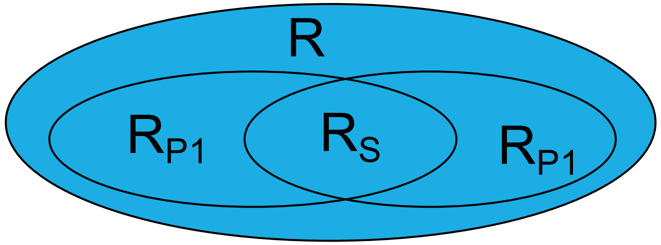
<u>Property</u> P: predicate on the system, identifying a set of runs $R_P \subseteq R$

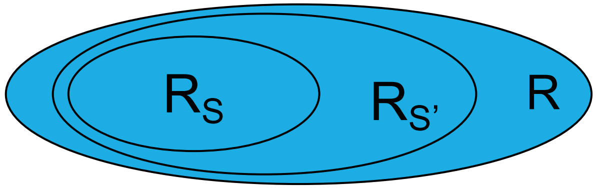
- $P \Rightarrow P'$ *iff* $R_P \subseteq R_{P'}$



<u>Specification</u> $S(P_1,\ldots,P_m)$: logical and of m properties, identifying a set of runs $R_S=R_{P_1} \cap \ldots \cap R_{P_m} \subseteq R$

- $S \rightarrow S'$ *iff* $R_S \subset R_{S'}$





TO specifications

Total order specifications are usually composed by four properties:

- A Validity property guarantees that messages sent by correct processes will eventually be delivered at least by correct processes;
- An Integrity property guarantees that no spurious or duplicate messages are delivered;
- An Agreement property ensures that (at least correct) processes deliver the same set of messages;
- An Order property constrains (at least correct) processes delivering the same messages to deliver them in the same order.

TO specifications

Total Order Broadcast = S(V,I,A,O)

- V = Validity
- I = Integrity

NUV
UI

- ~~A~~ = Agreement
- ~~O~~ = Order

→ TO(A,O)

Distinct specifications arise from distinct formulations of each property

- uniform vs non-uniform
- A uniform property imposes restrictions on the behavior of (at least) correct processes on the basis of events occurred in some process

TO Specifications

Crash failure + Perfect channels ⇒

- **NUV:** if a correct process TOCAST a message m then some *correct* process will eventually deliver m

- **UI**: For any message m, every process p delivers m at most once and only if m was previously TOCAST by some (correct or not) process.

The Agreement property

UNIFORM AGREEMENT (UA)

If a process (correct or not) TODelivers a message m, then all correct processes will eventually TODeliver m

NON-UNIFORM AGREEMENT (NUA)

If a correct process TODelivers a message m, then all correct processes will eventually TODeliver m

CONSTRAINS THE SET OF DELIVERED MESSAGES

Correct processes always deliver the same set of messages M

Each faulty process p delivers a set M_p

UA: $M_p \subseteq M$

NUA: M_p *can be* s.t. $M_p - M \neq \emptyset$

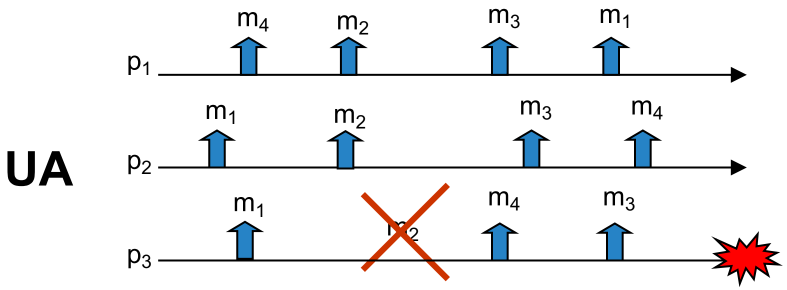
The Agreement property

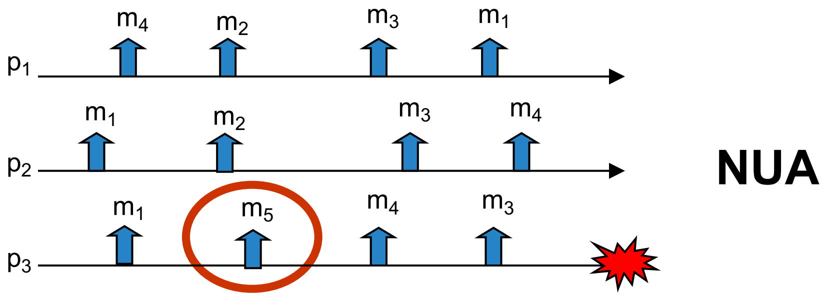
UNIFORM AGREEMENT (UA)

If a process (correct or not) TODelivers a message m, then all correct processes will eventually TODeliver m

NON-UNIFORM AGREEMENT (NUA)

If a correct process TODelivers a message m, then all correct processes will eventually TODeliver m





The Ordering Property

STRONG UNIFORM TOTAL ORDER (SUTO)

If some process TODelivers some message m before message m′, then a process TODelivers m’ only after it has TODelivered m.

- same order
- same prefix of the set of delivered messages
- after an omission, disjoint sets of delivered messages

WEAK UNIFORM TOTAL ORDER (WUTO)

If process p and process q both TODdeliver messages m and m’, then p TODelivers m before m’ if and only if q TODdelivers m before m’.

- no restrictions on the set of delivered messages

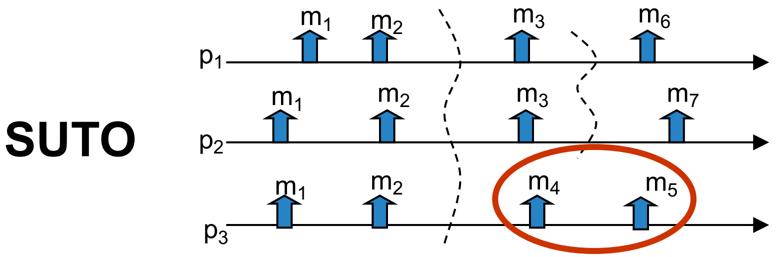
The Order Property

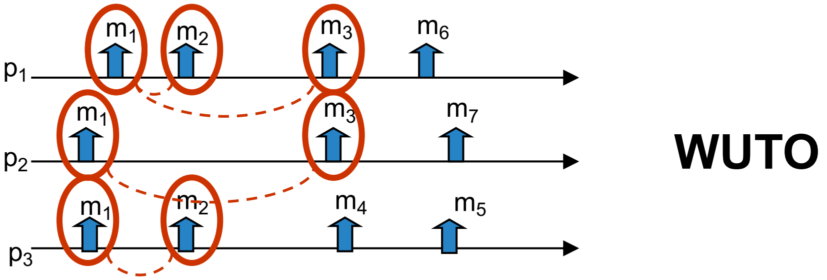
STRONG UNIFORM TOTAL ORDER (SUTO)

If some process TODelivers some message m before message m′, then a process TODelivers m′ only after it has TODelivered m.

WEAK UNIFORM TOTAL ORDER (WUTO)

If process p and process q both TODdeliver messages m and m’, then p TODelivers m before m’ if and only if q TODdelivers m before m’.





The Order Property

SUTO and WUTO are uniform but they both have a non-uniform counterpart

STRONG NON-UNIFORM TOTAL ORDER (SNUTO)

If some correct process TODelivers some message m before message m′, then a correct process TODelivers m′ only after it has TODelivered m.

WEAK NON-UNIFORM TOTAL ORDER (WNUTO)

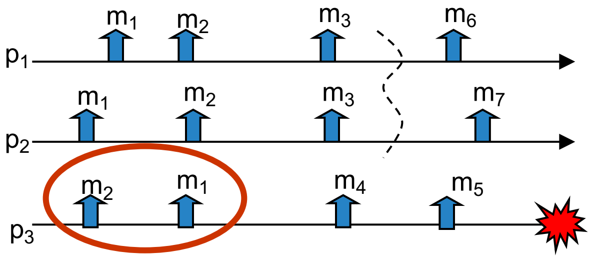
If correct processes p and q both TODeliver messages m and m′, then p TODelivers m before m′ if and only if q TODelivers m before m′.

The Order property (2)

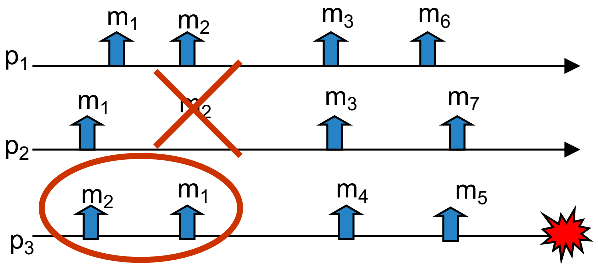
SUTO ⇒ WUTO

SNUTO ⇒ WNUTO

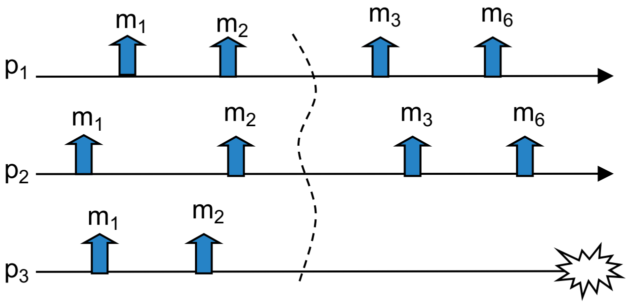
SNUTO



WNUTO



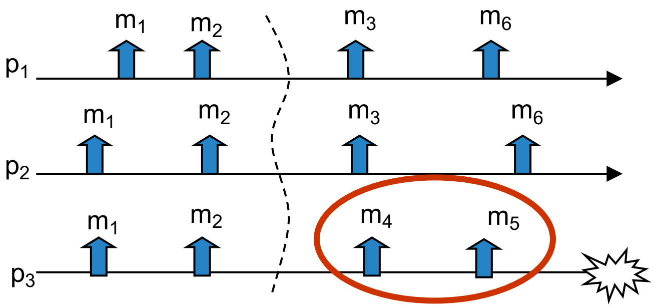
TO specifications

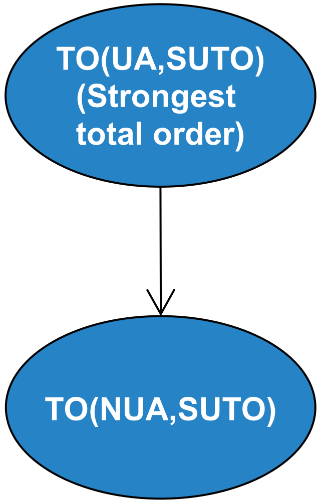


TO(UA,SUTO)

- The strongest TO spec.

TO(NUA,SUTO)



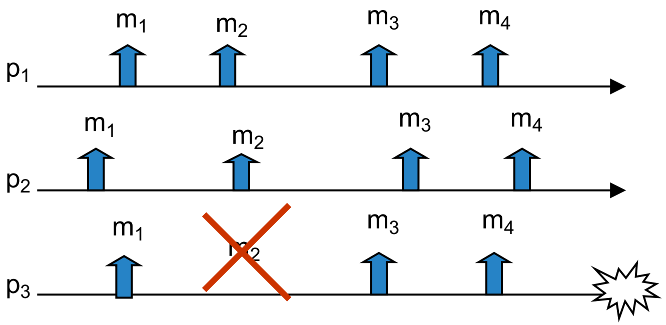


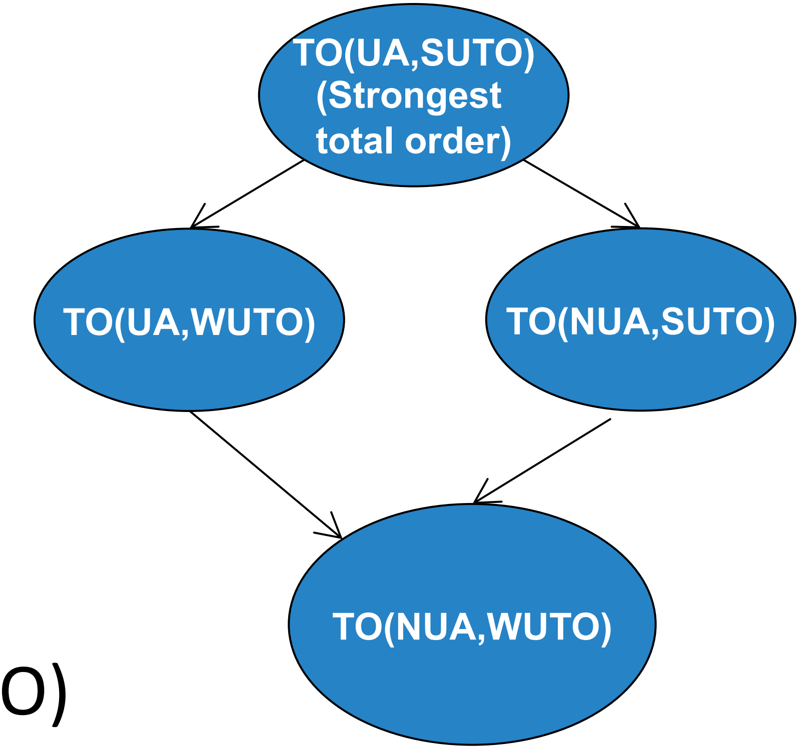
TO(UA,SUTO) (Strongest total order)

TO(NUA,SUTO)

TO specifications (2)

TO(UA,WUTO)

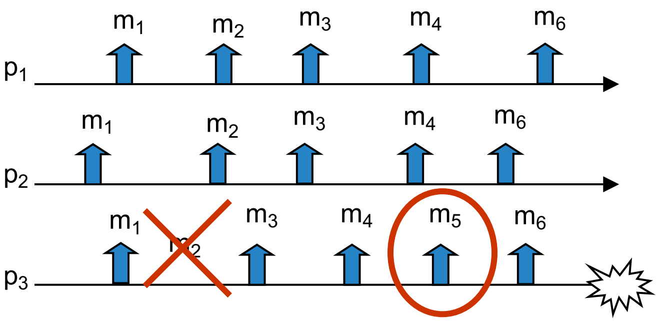



TO(UA,SUTO) (Strongest total order)

TO(UA,WUTO)

TO(NUA,SUTO)

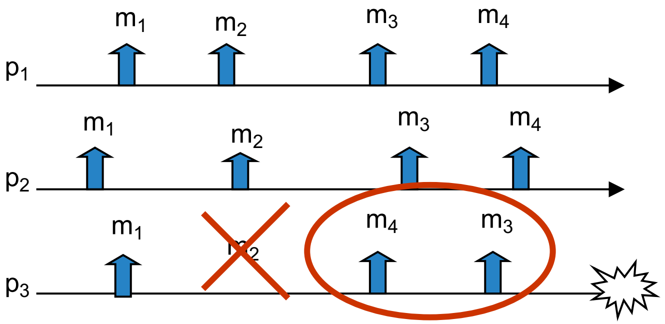
TO(NUA,WUTO)



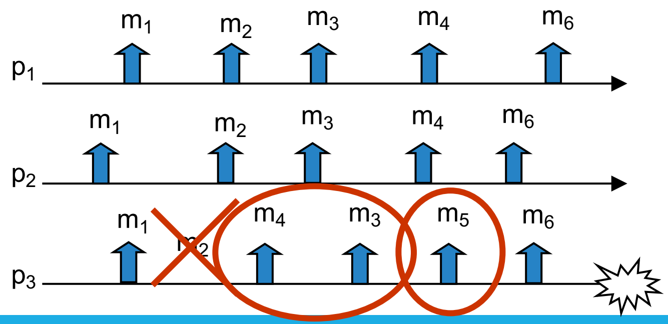
TO(NUA,WUTO)

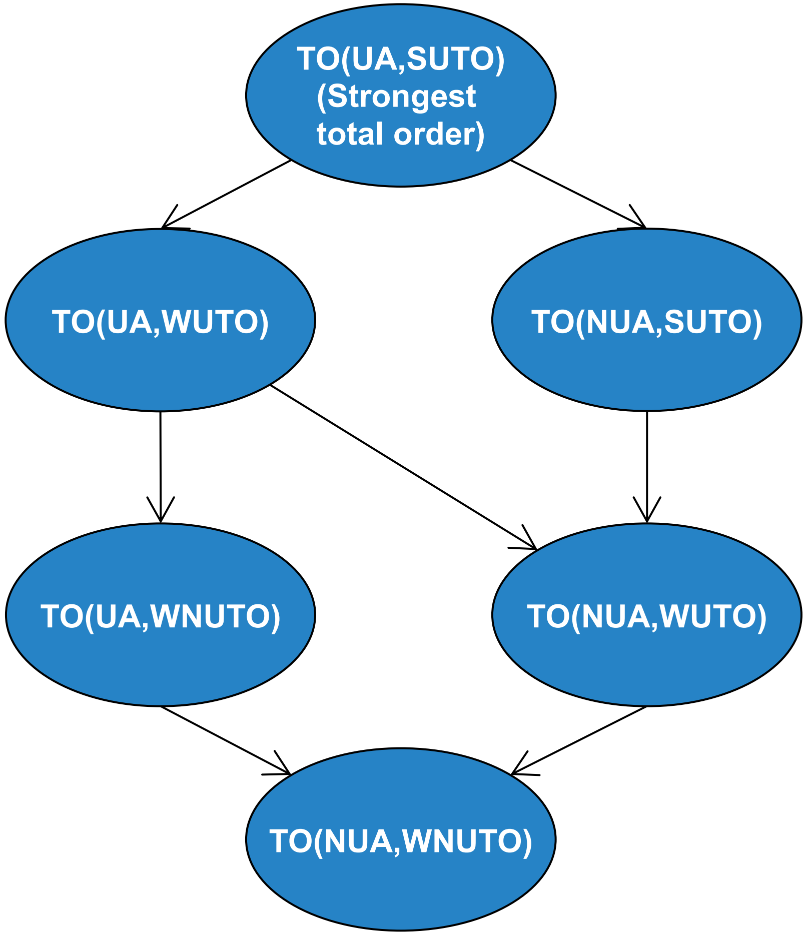
TO specifications (3)

TO(UA,WNUTO)



TO(NUA,WNUTO)





TO(UA,SUTO) (Strongest total order)

TO(UA,WUTO)

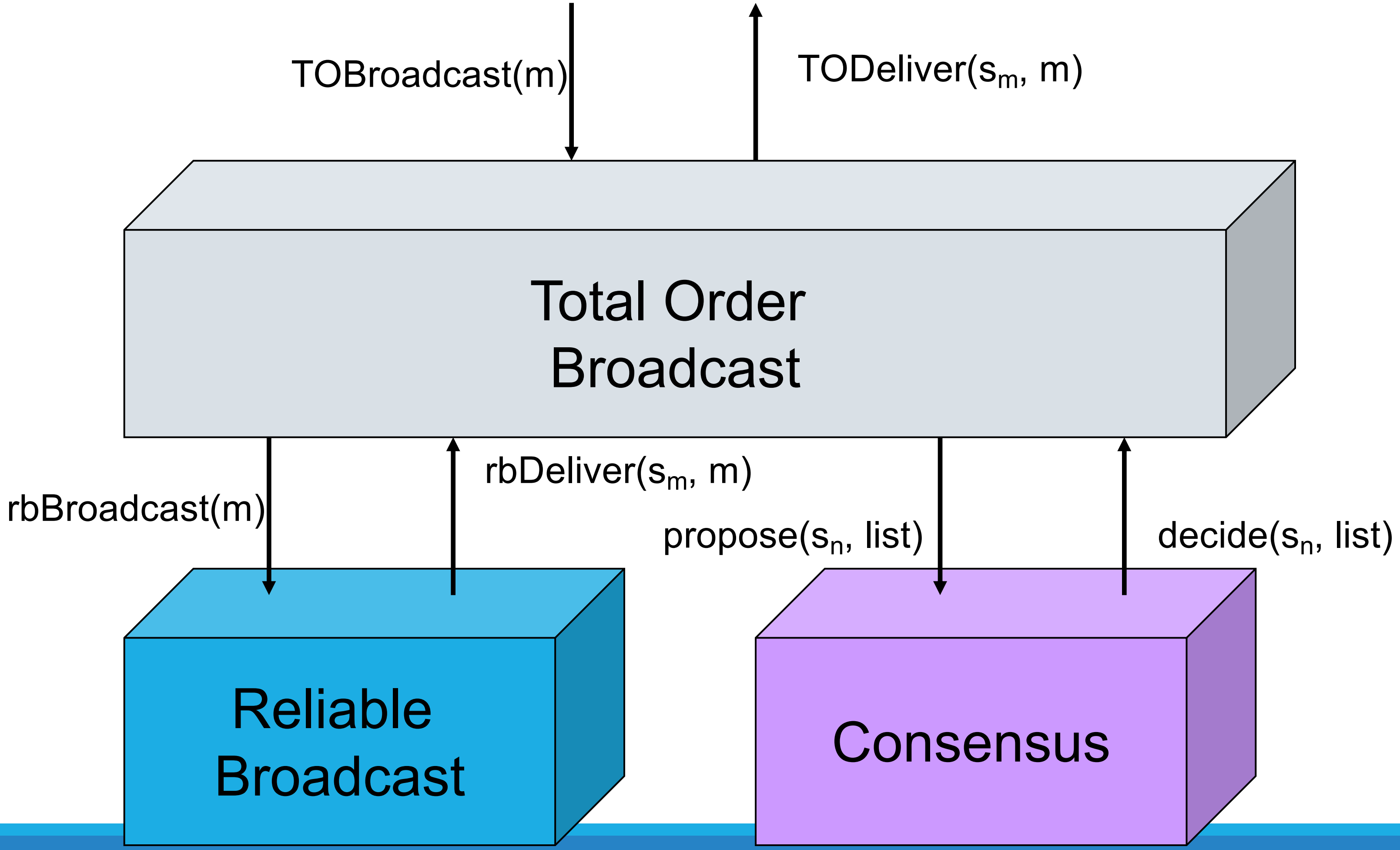
TO(NUA,SUTO)

TO(UA,WNUTO)

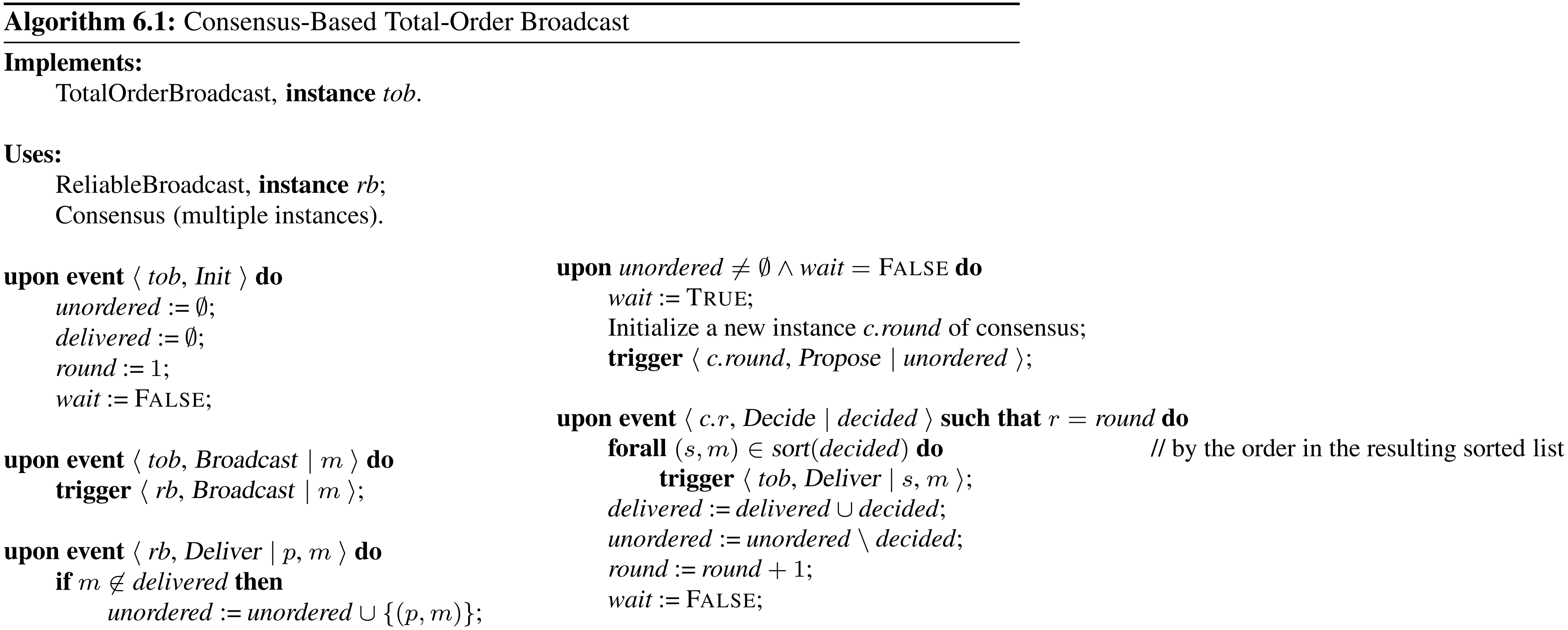
TO(NUA,WUTO)

TO(NUA,WNUTO)

Total Order Implementation



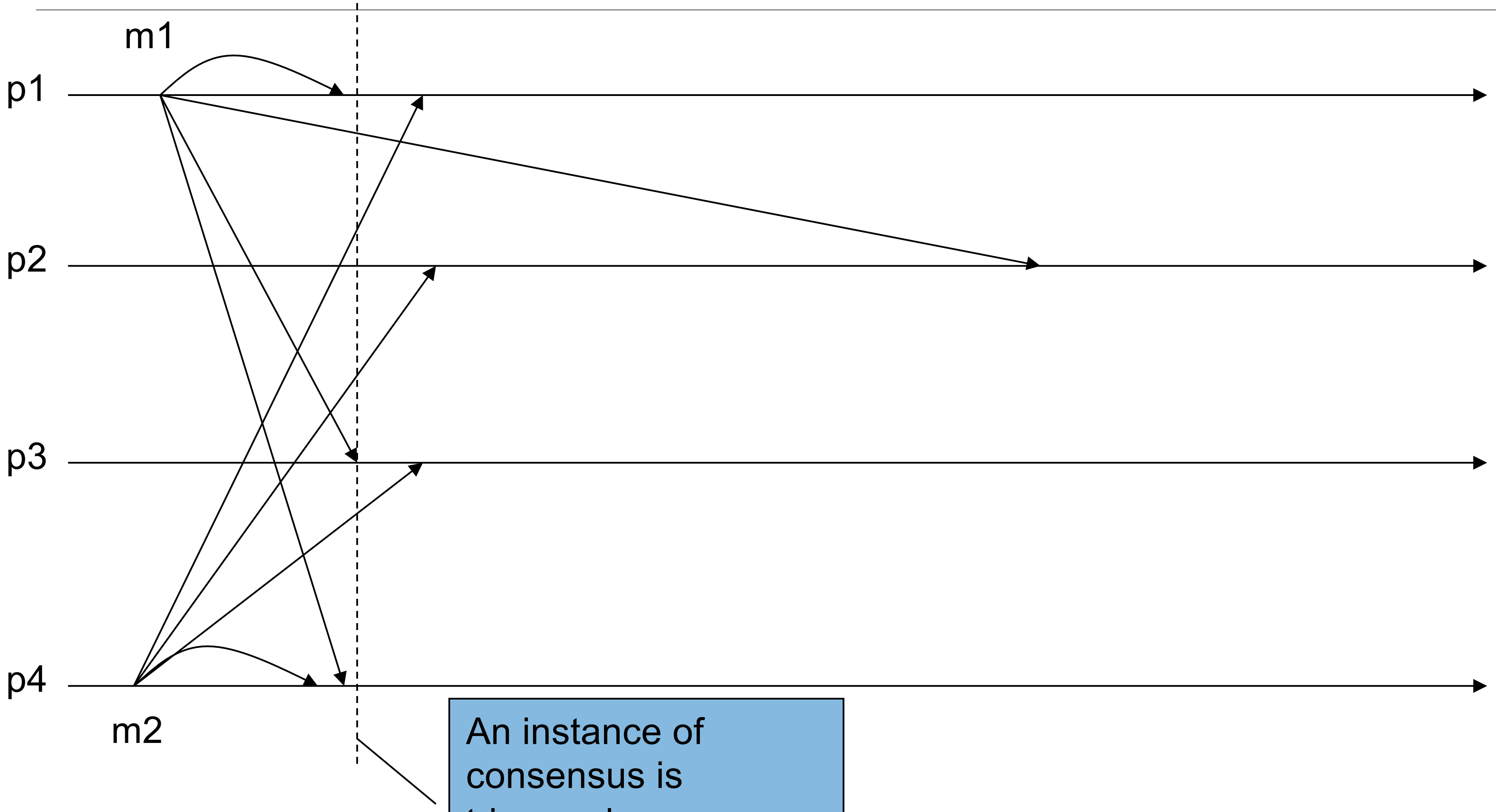
Total Order Algorithm



Algorithm 6.1: Consensus-Based Total-Order Broadcast

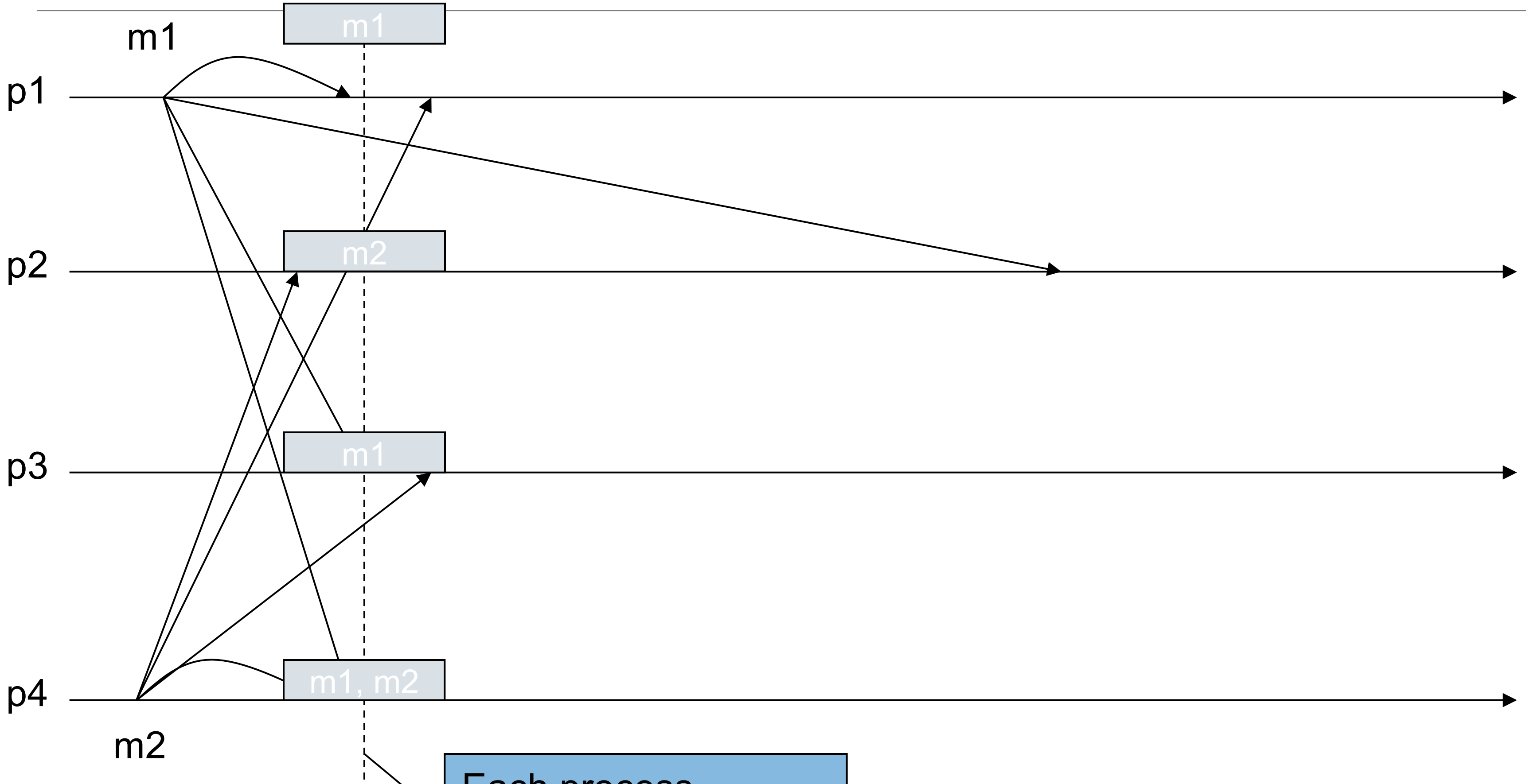
```
Implements:
    TotalOrderBroadcast, instance tob.

Uses:
    ReliableBroadcast, instance rb;
    Consensus (multiple instances).

upon event ⟨ tob, Init ⟩ do
    unordered := $\emptyset$;
    delivered := $\emptyset$;
    round := $1$;
    wait := FALSE;

upon event ⟨ tob, Broadcast | $m$ ⟩ do
    trigger ⟨ rb, Broadcast | $m$ ⟩;

upon event ⟨ rb, Deliver | $p, m$ ⟩ do
    if $m \notin$ delivered then
        unordered := unordered $\cup \{(p, m)\}$;

upon unordered $\neq \emptyset \wedge$ wait $=$ FALSE do
    wait := TRUE;
    Initialize a new instance c.round of consensus;
    trigger ⟨ c.round, Propose | unordered ⟩;

upon event ⟨ c.$r$, Decide | decided ⟩ such that $r =$ round do
    forall $(s, m) \in$ sort(decided) do                // by the order in the resulting sorted list
        trigger ⟨ tob, Deliver | $s, m$ ⟩;
    delivered := delivered $\cup$ decided;
    unordered := unordered $\setminus$ decided;
    round := round $+ 1$;
    wait := FALSE;
```

Example



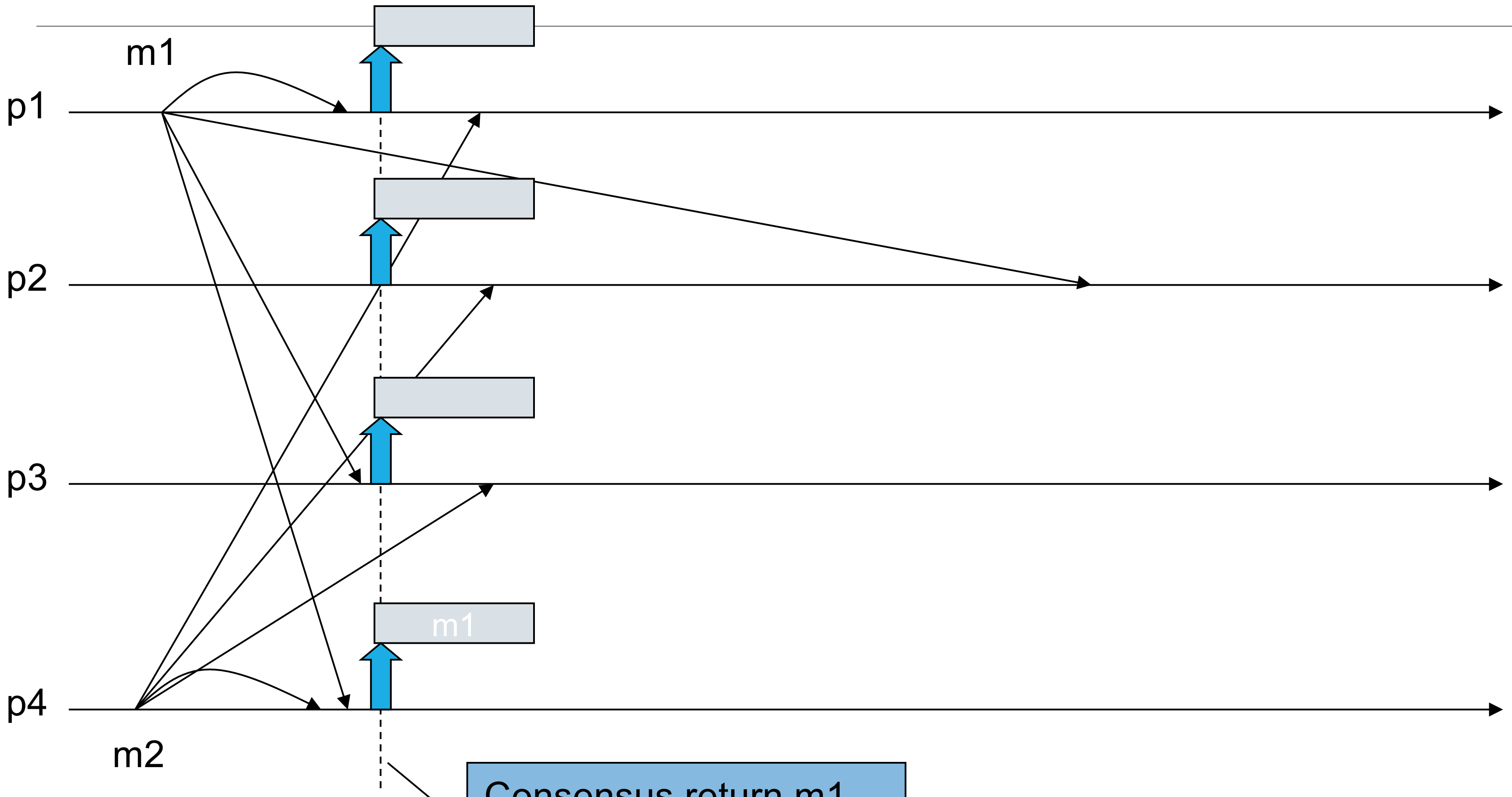
An instance of consensus is triggered

Example



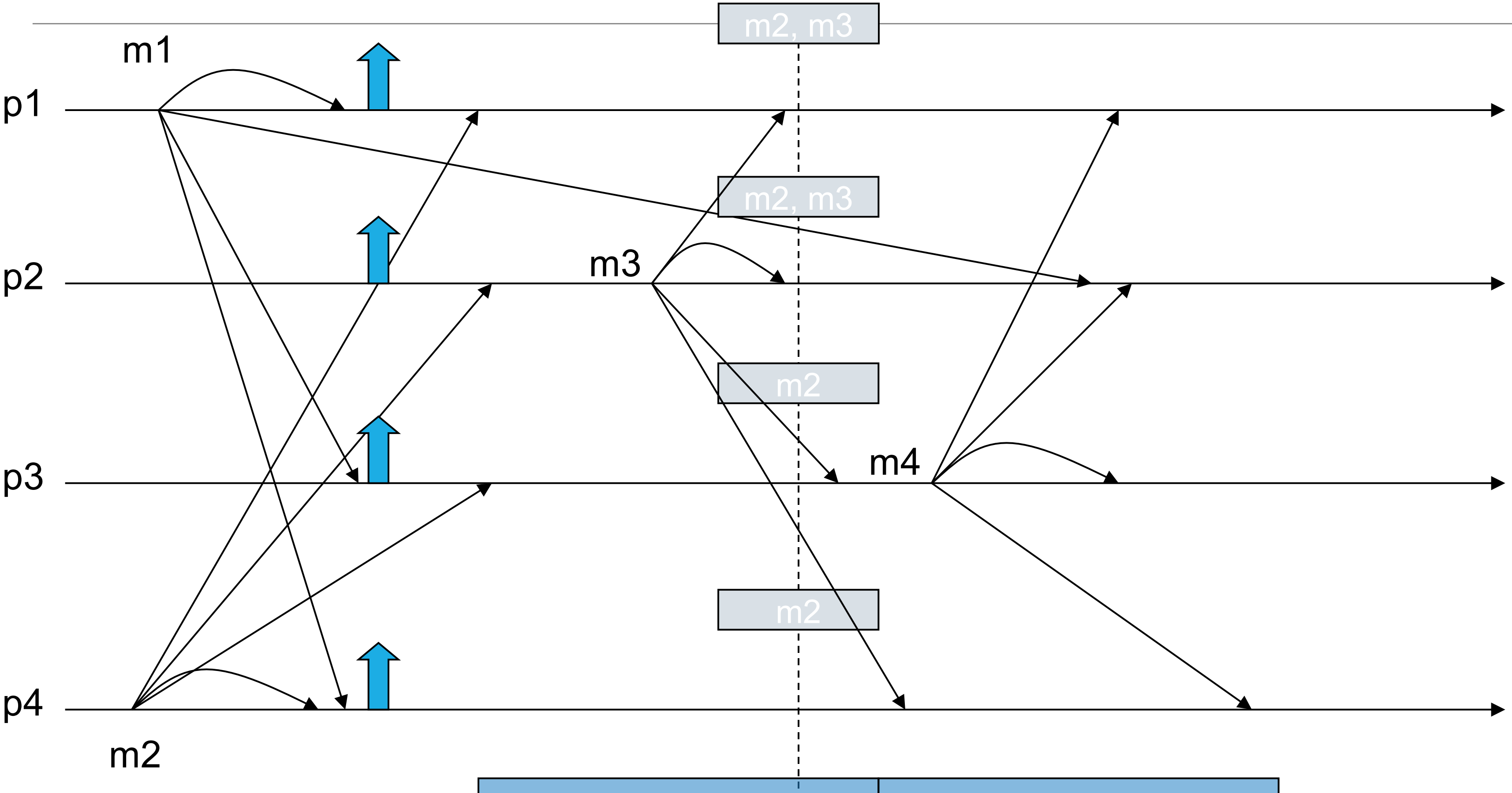
Each process proposes its *unordered* buffer

Example



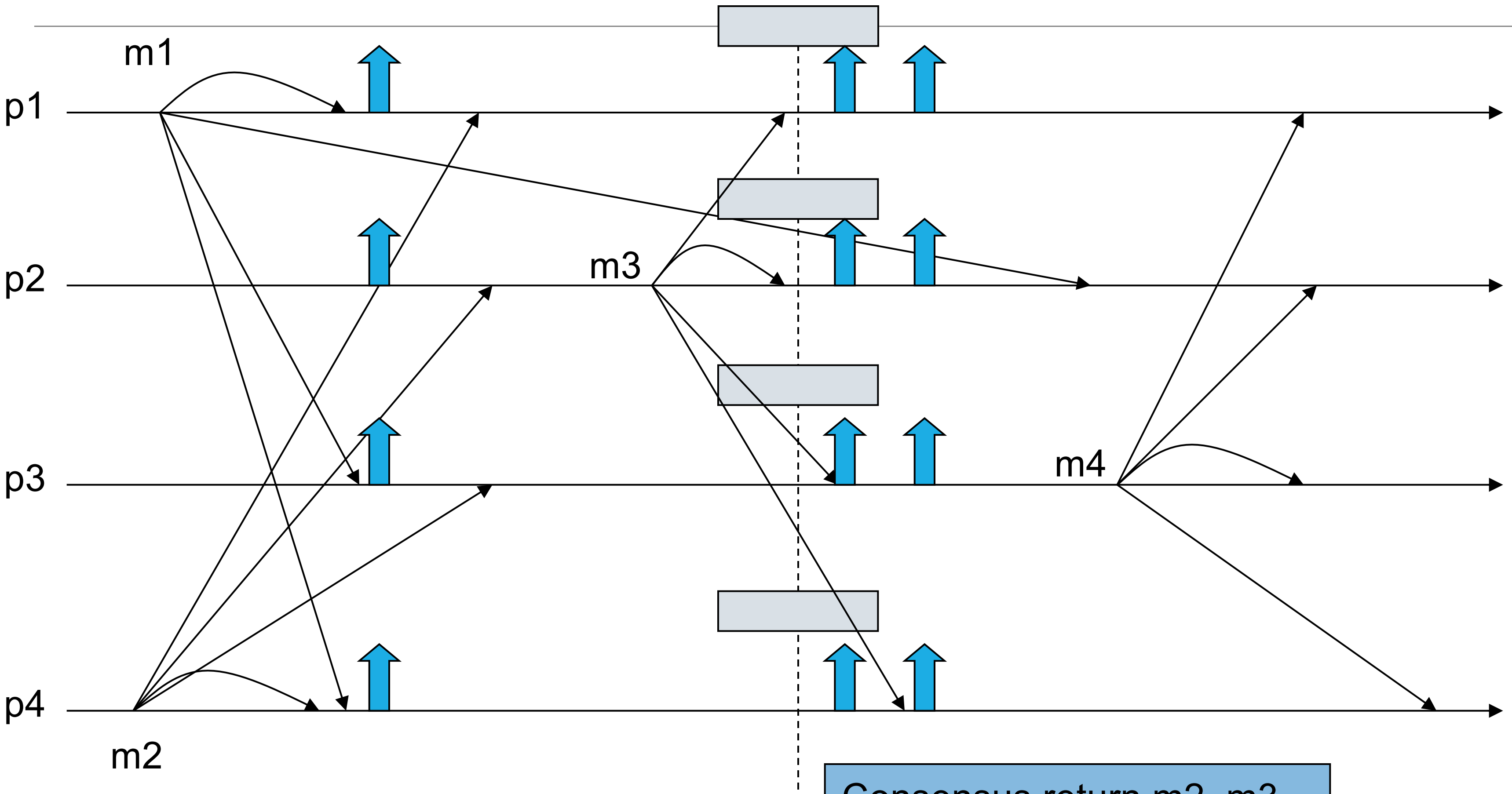
Consensus return m1 and all processes deliver m1

Example



An instance of consensus is triggered	Each process proposes its *unordered* buffer

Example



Consensus return m2, m3 and all processes deliver m2 and then m3

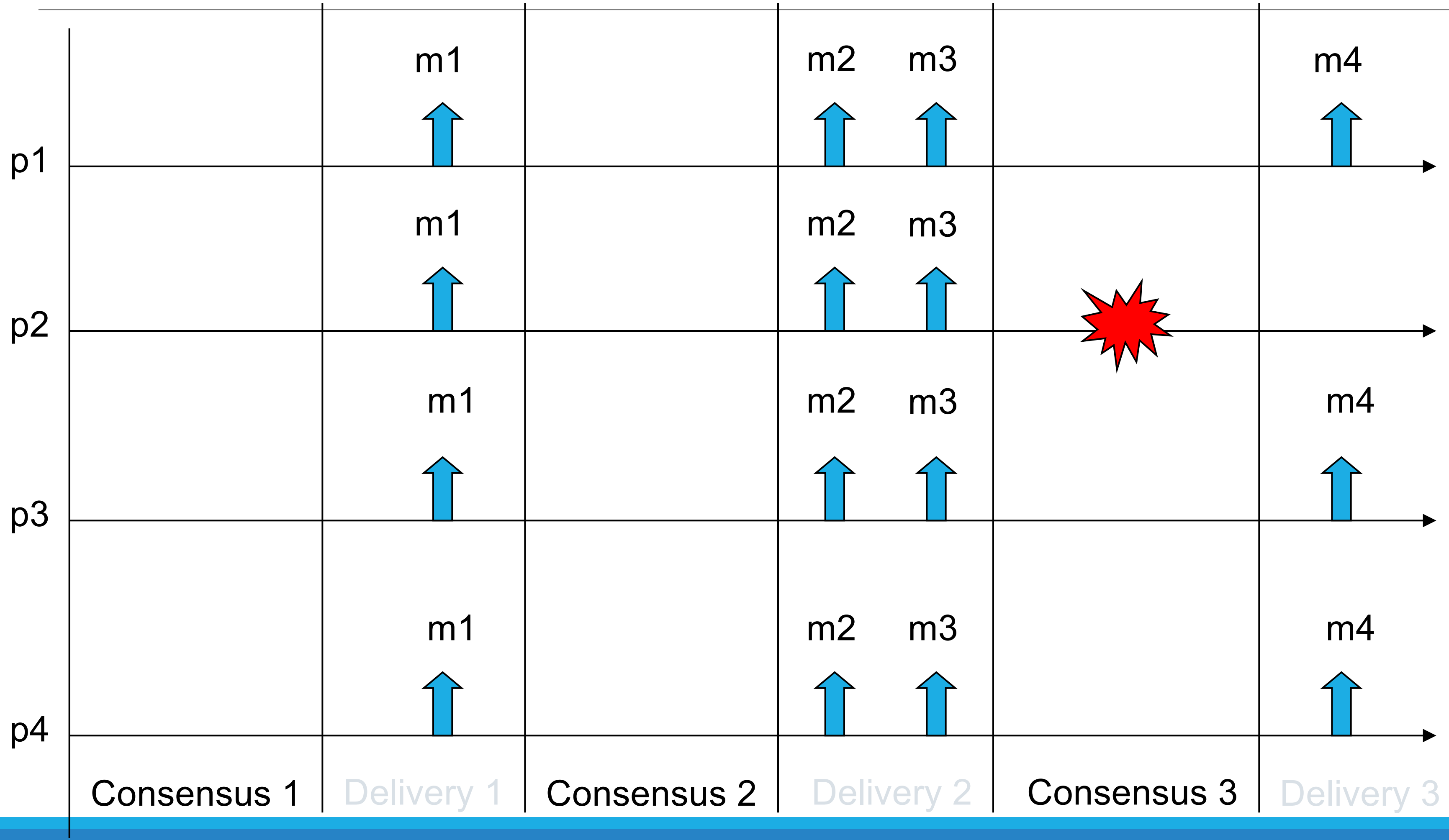
Exercice

Which TO specification is satisfied by this algorithm?

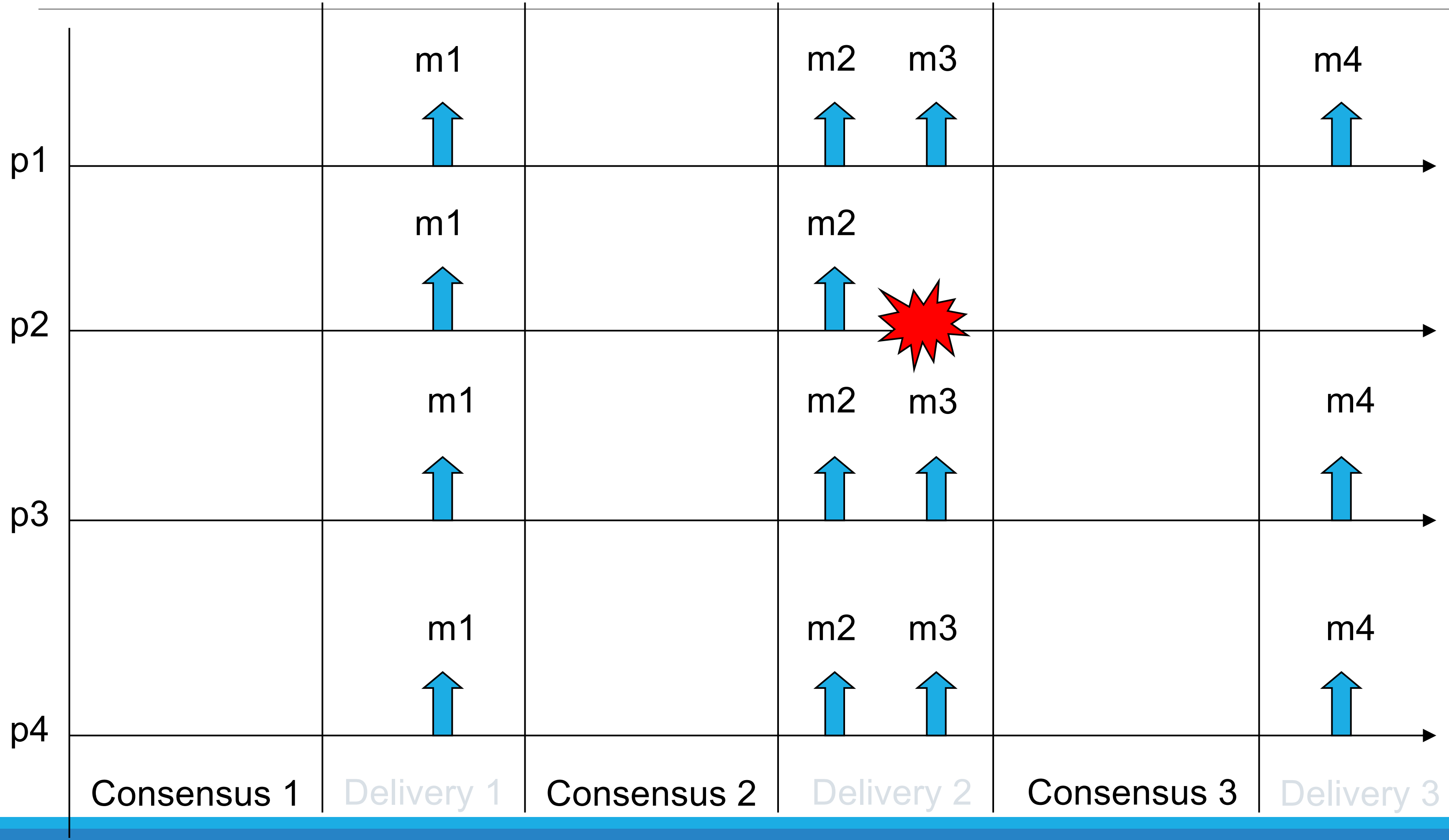
It depends from the assumptions about Reliable Broadcast and Consensus

Consensus Reliable Broadcast	Uniform	Non Uniform
Uniform		
Non Uniform		

Example 1 (UC and URB)



Example 2 (UC and URB)



Uniform Consensus (UC) and Uniform Reliable Broadcast (URB)

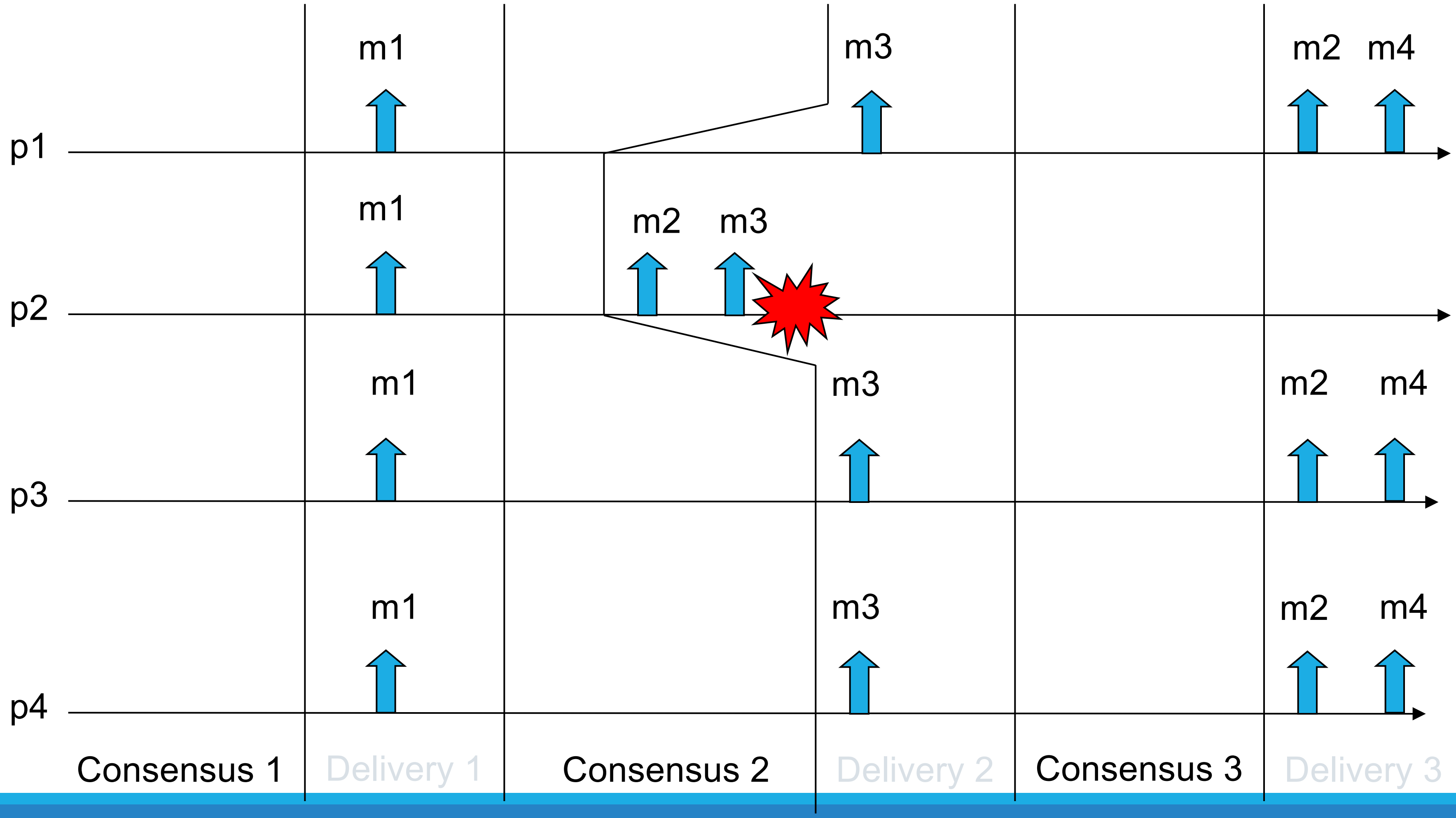
Assuming both Consensus and Reliable Broadcast uniform we have

TO (UA, SUTO)

Proof.

- Due to URB all the processes (even the faults) deliver the same set of messages
- The unordered buffer contains the same set of messages for each process
  - All the processes will deliver the same set of messages (UA)
- Due to UC, all processes (even the faults) decide for the same list of messages
- Messages are sorted by a deterministic rule
  - All processes will deliver the messages in the same order

Example (NUC and URB)



Non Uniform Consensus (NUC) and Uniform Reliable Broadcast (URB)

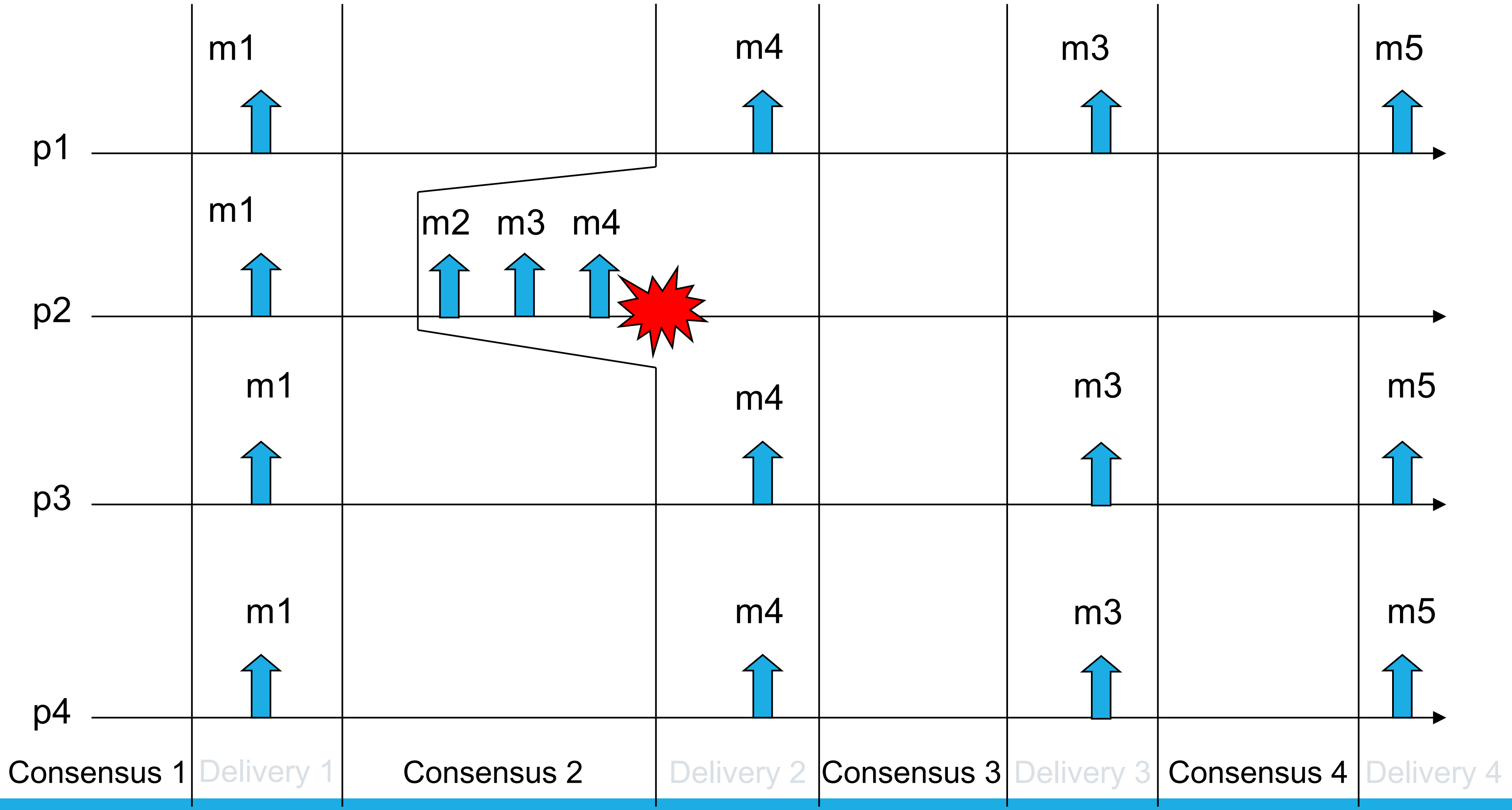
Assuming both Consensus and Reliable Broadcast uniform we have

TO (UA, WNUTO)

Proof.

- Due to URB all the processes (even the faults) deliver the same set of messages
- The unordered buffer contains the same set of messages for each process
  - All the processes will deliver the same set of messages (UA)
- Due to NUC, all correct processes decide for the same list of messages
- Faulty processes can decide differently
  - All correct processes will deliver the messages in the same order
  - Faulty processes will deliver, just before a crash, a different sequence of messages

Example (NUC and NURB)



Non Uniform Consensus (NUC) and Non Uniform Reliable Broadcast (NURB)

Assuming both Consensus and Reliable Broadcast uniform we have

TO (NUA, WNUTO)

Proof.

- Due to NURB correct processes deliver the same set of messages
- Faulty processes can deliver other messages
  - Only correct processes will deliver the same set of messages (NUA)
- Due to NUC, all correct processes decide for the same list of messages
- Faulty processes can decide differently
  - All correct processes will deliver the messages in the same order
  - Faulty processes will deliver, just before a crash, a different sequence of messages

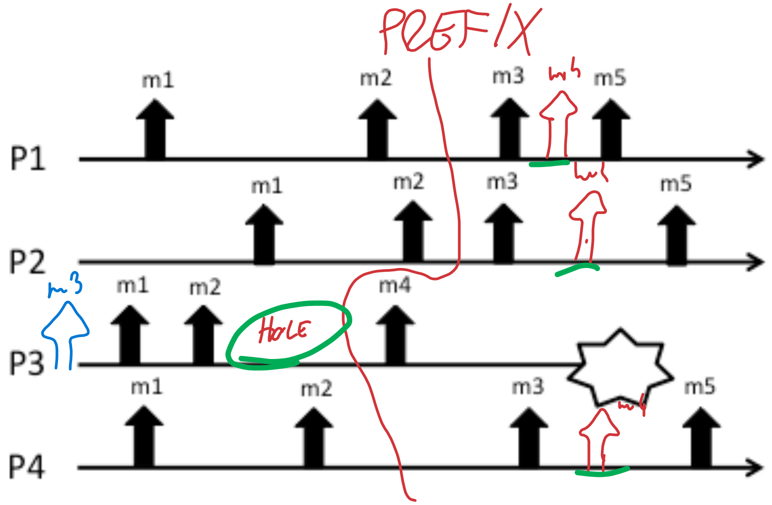
Consensus Reliable Broadcast	Uniform	Non Uniform
Uniform	UA SUTO	UA WNUTO
Non Uniform		NUA WNUTO

Exercice

Which specification is satisfied assuming UC and NURB?

Exercice

Consider the run depicted in the figure:



1. TO(NUA, SUTO)

2. Adding m4 to all (UA)

Add m4 after m3 (WUTO)

3. Put m3 before m1

1.Which type of total ordering is satisfied by the run? Specify both the agreement and the ordering properties.
2.Modify the run in order to satisfy TO(UA, WUTO) but not TO (UA SUTO)
3.Modify the run in order to satisfy TO(NUA, WNUTO) but not TO(NUA, WUTO)

References

C. Cachin, R. Guerraoui and L. Rodrigues. Introduction to Reliable and Secure Distributed Programming, Springer, 2011

- Chapter 6 – Section 6.1

Stefano Cimmino, Carlo Marchetti, Roberto Baldoni ”*A Guided Tour on Total Order Specifications*” WORDS Fall 2003: 187-194