

# Sistemi Distribuiti

## Total Order Broadcast Implementation

**Dott. Ing. Silvia Bonomi**

**bonomi@dis.uniroma1.it**

# Total Order Specification

- Validity
- Integrity
- Agreement
  - **Uniform (UA)** If a process (correct or not) delivers a message  $m$  then all correct processes will eventually deliver  $m$
  - **Non Uniform (NUA)** If a correct process delivers a message  $m$  then all correct processes will eventually deliver

## ■ Order

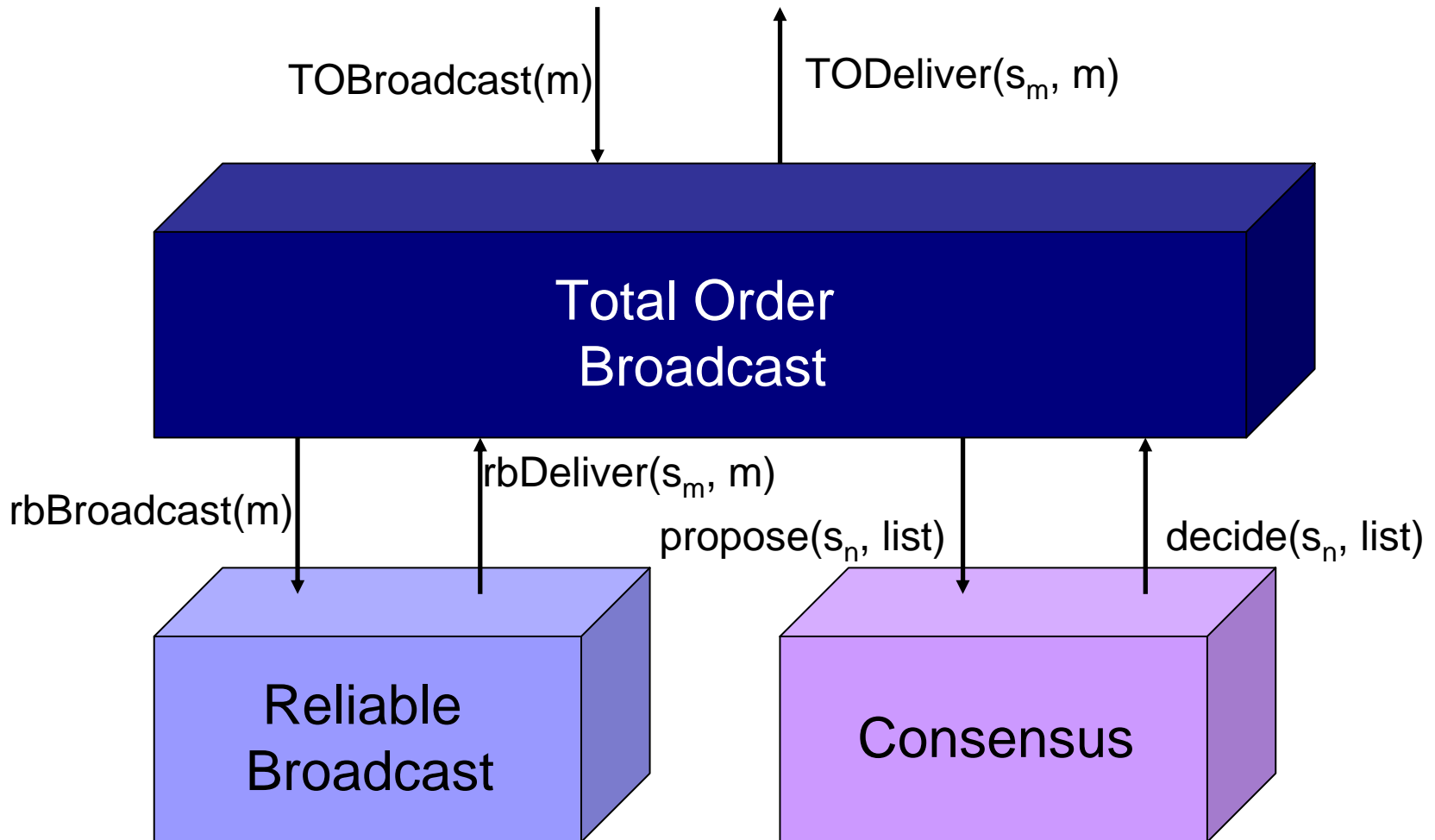
### □ Strong

- **Uniform (SUTO)** If some process delivers some message  $m$  before  $m'$  then every correct process delivers  $m'$  only after it has delivered  $m$ .
- **Non Uniform (SNUTO)** If some correct process delivers some message  $m$  before  $m'$  then every correct process delivers  $m'$  only after it has delivered  $m$ .

### □ Weak

- **Uniform (WUTO)** If both processes  $p$  and  $q$  deliver messages  $m$  and  $m'$  then  $p$  delivers  $m$  before  $m'$  if and only if  $q$  delivers  $m$  before  $m'$ .
- **Non Uniform (WNUTO)** If correct processes  $p$  and  $q$  deliver messages  $m$  and  $m'$  then  $p$  delivers  $m$  before  $m'$  if and only if  $q$  delivers  $m$  before  $m'$ .

# Total Order Implementation



# Total Order Algorithm

**upon event** <Init> **do**

unordered := delivered :=  $\emptyset$ ;  
sn := 1;  
wait := false;

**upon event** <toBroadcast | m> **do**

**trigger** <rbBroadcast | m> ;

**upon event** <rbDeliver |  $s_m, m$ > **do**

**if**  $m \notin$  delivered **then**  
unordered := unordered  $\cup$   $\{(s_m, m)\}$ ;

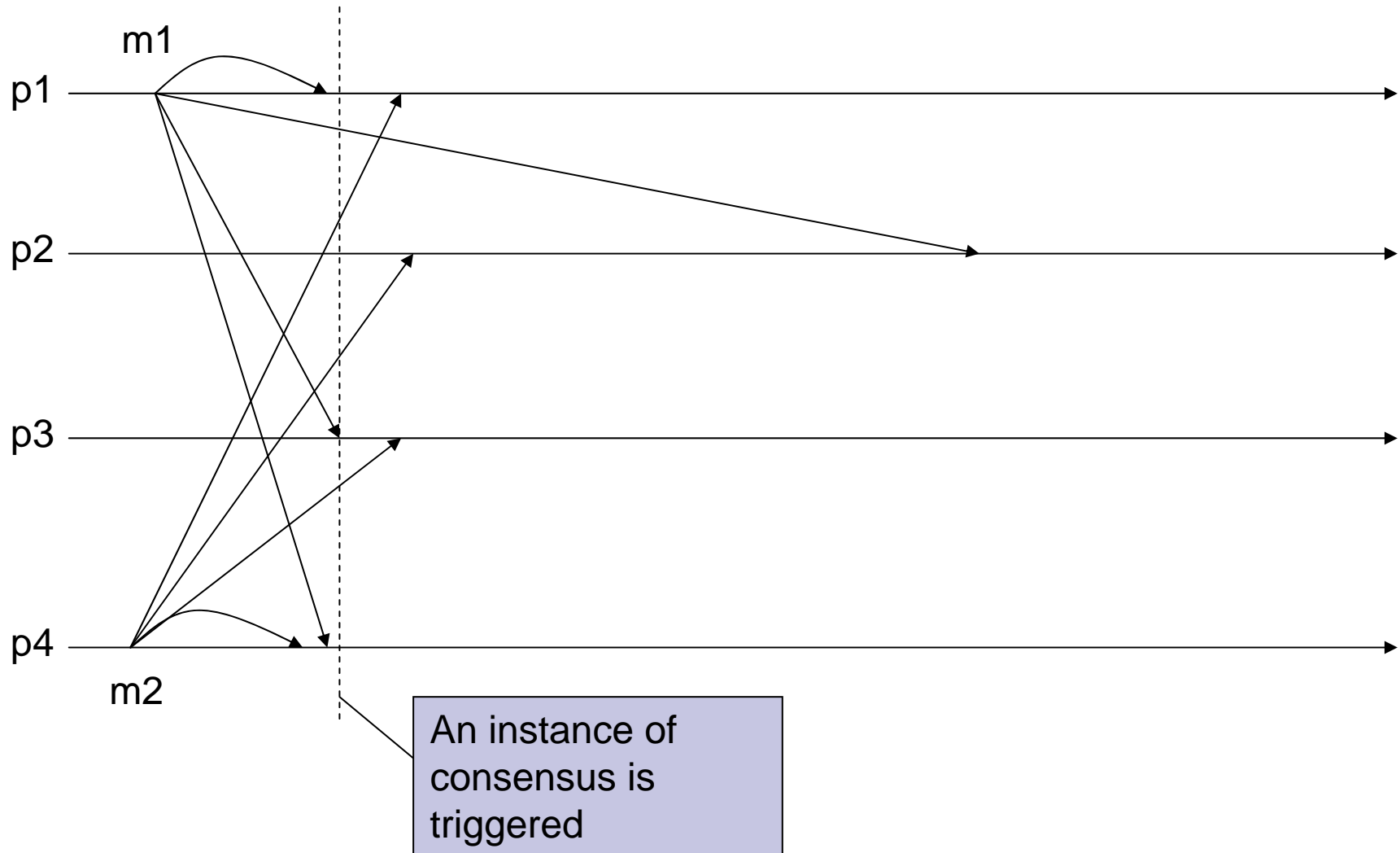
**upon** (unordered =  $\emptyset$ )  $\wedge$  (wait = false) **do**

wait := true;  
**trigger** <propose |  $s_n$ , unordered> ;

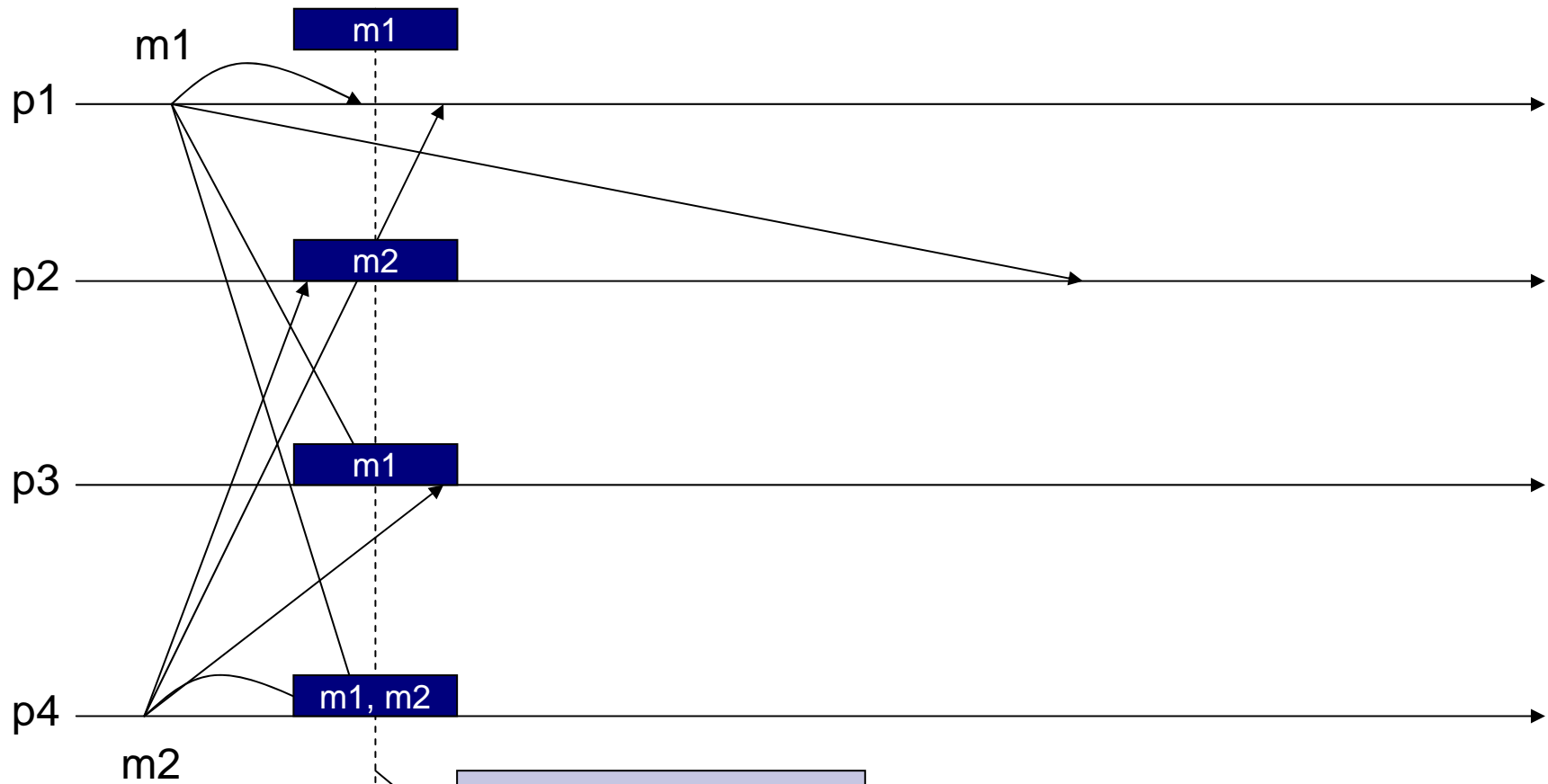
**upon event** <decide |  $s_n$ , decided > **do**

delivered := delivered  $\cup$  decided;  
unordered := unordered  $\setminus$  decided;  
decided := sort (decided);  
% some deterministic order;  
**forall** ( $s_m, m$ )  $\in$  decided **do**  
    **trigger** <toDeliver |  $s_m, m$  >;  
    % following the deterministic order  
 $s_n := s_n + 1$ ;  
wait := false;

# Example

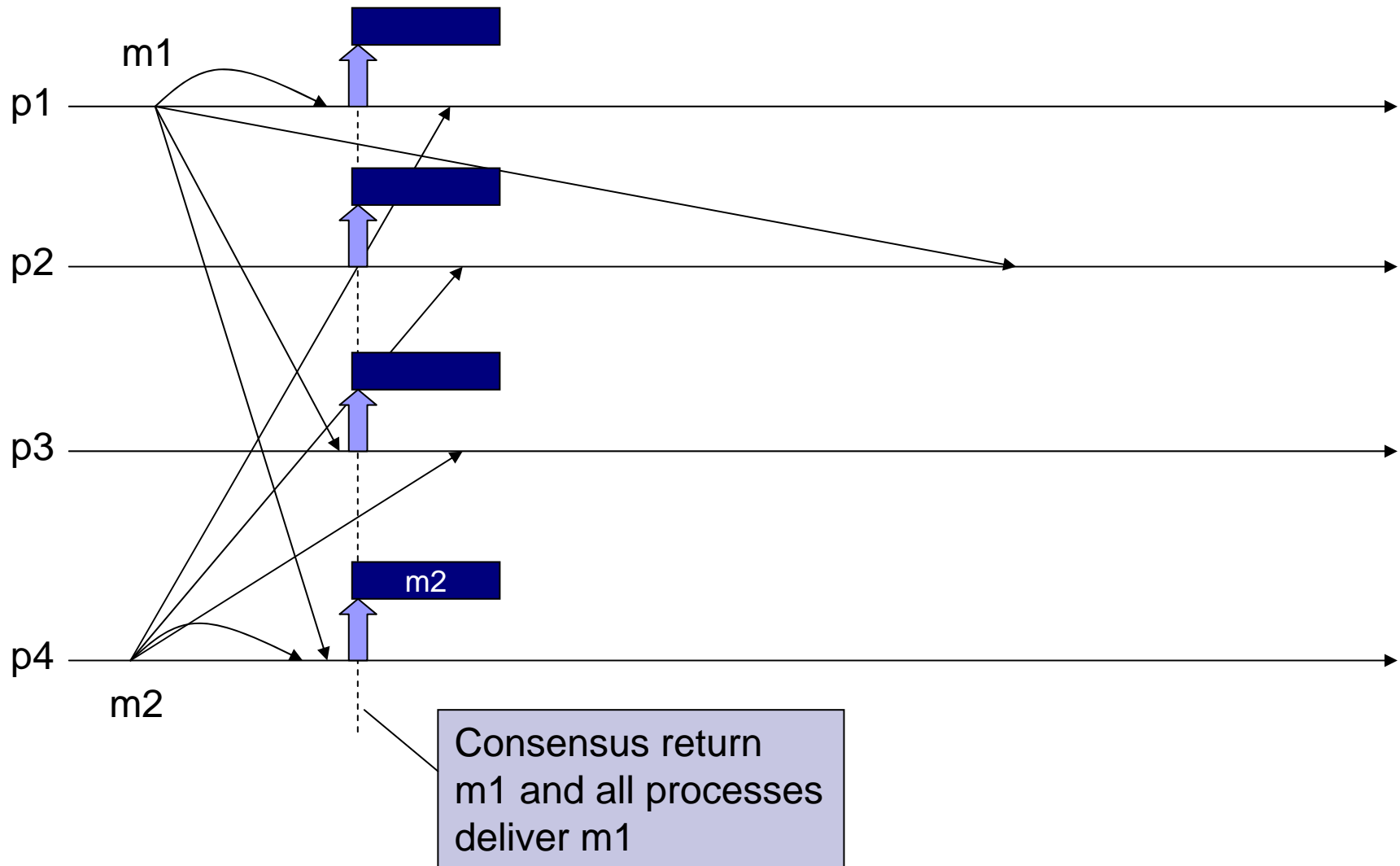


# Example

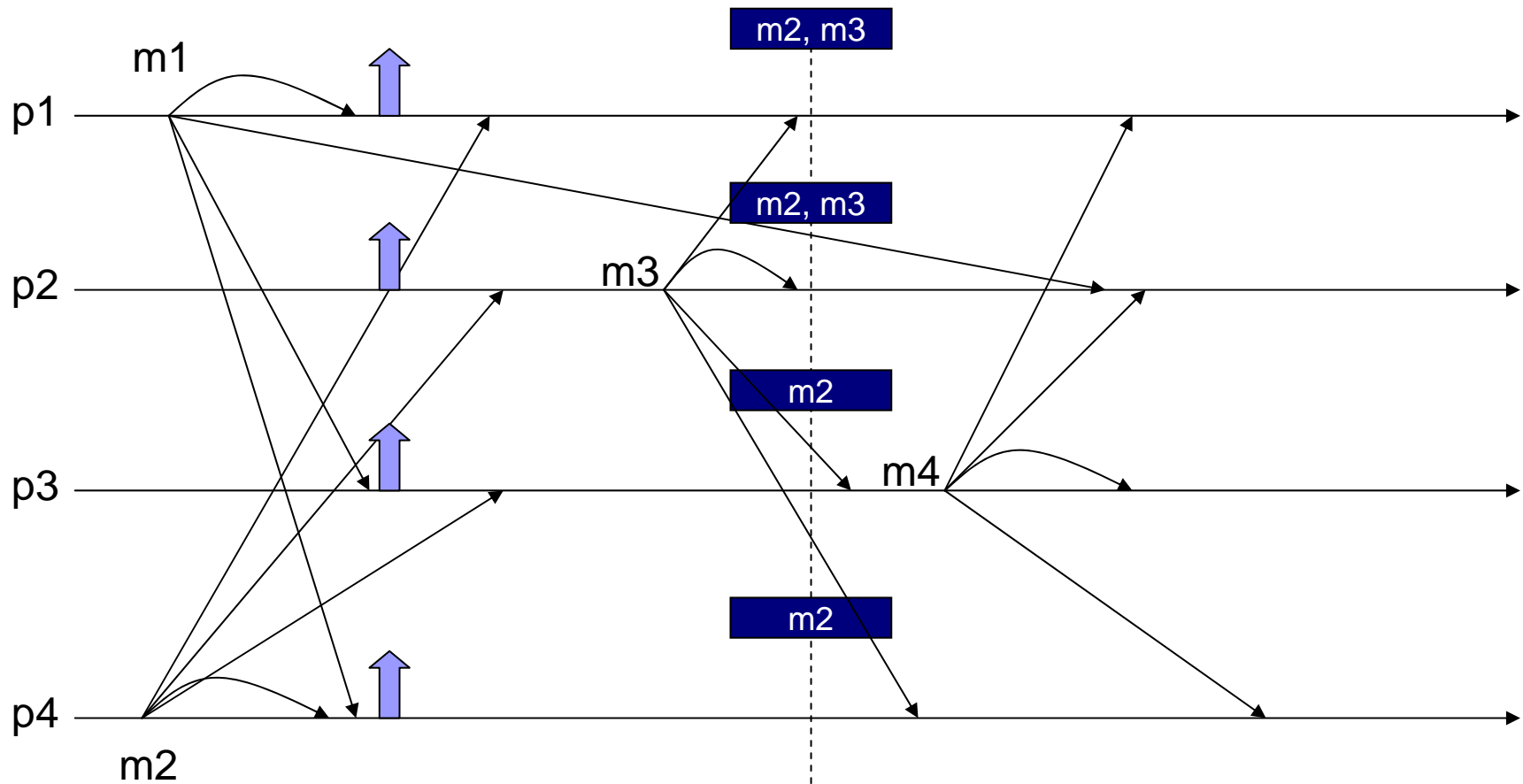


Each process proposes its *unordered* buffer

# Example



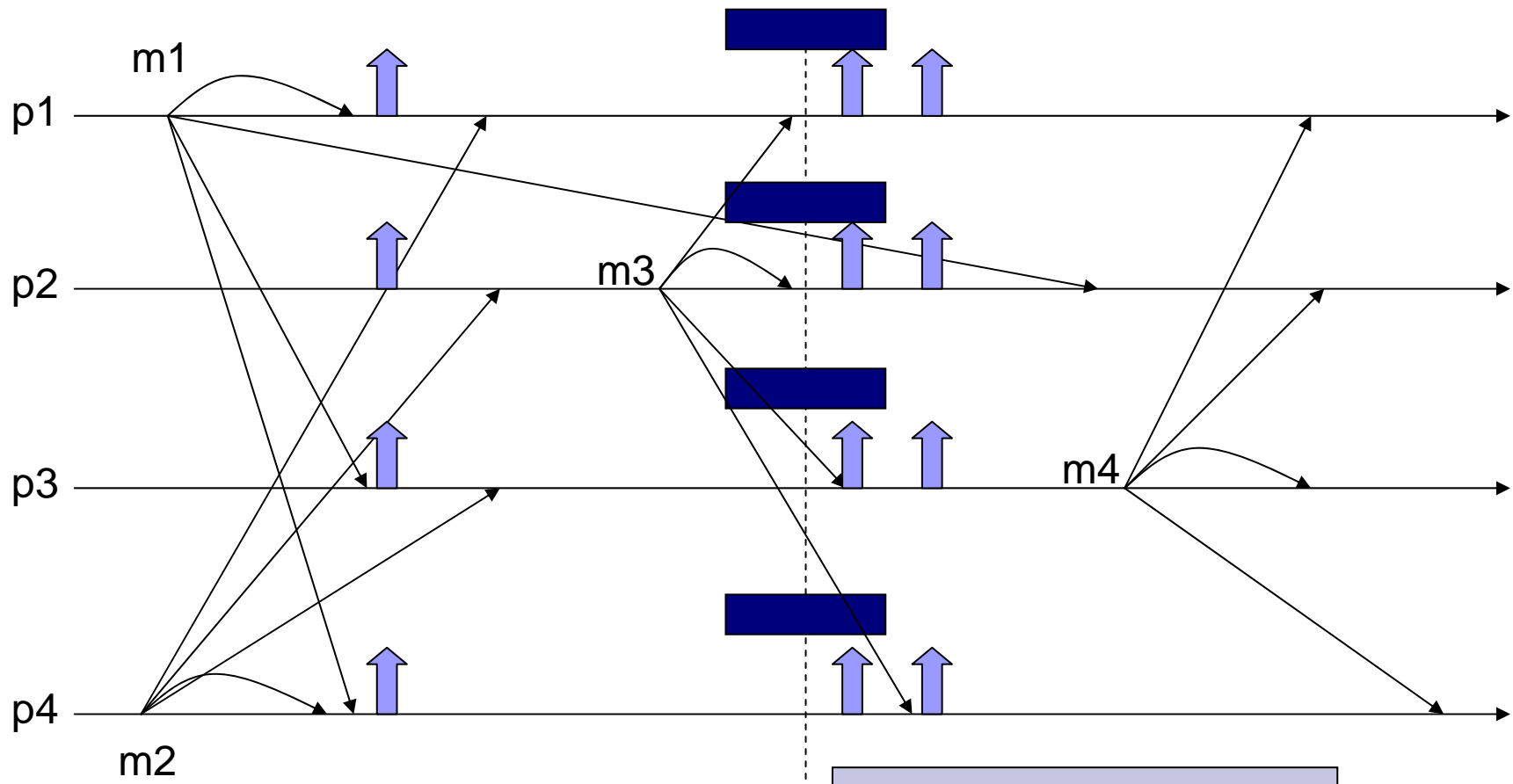
# Example



An instance of  
consensus is  
triggered

Each process  
proposes its  
*unordered* buffer

# Example



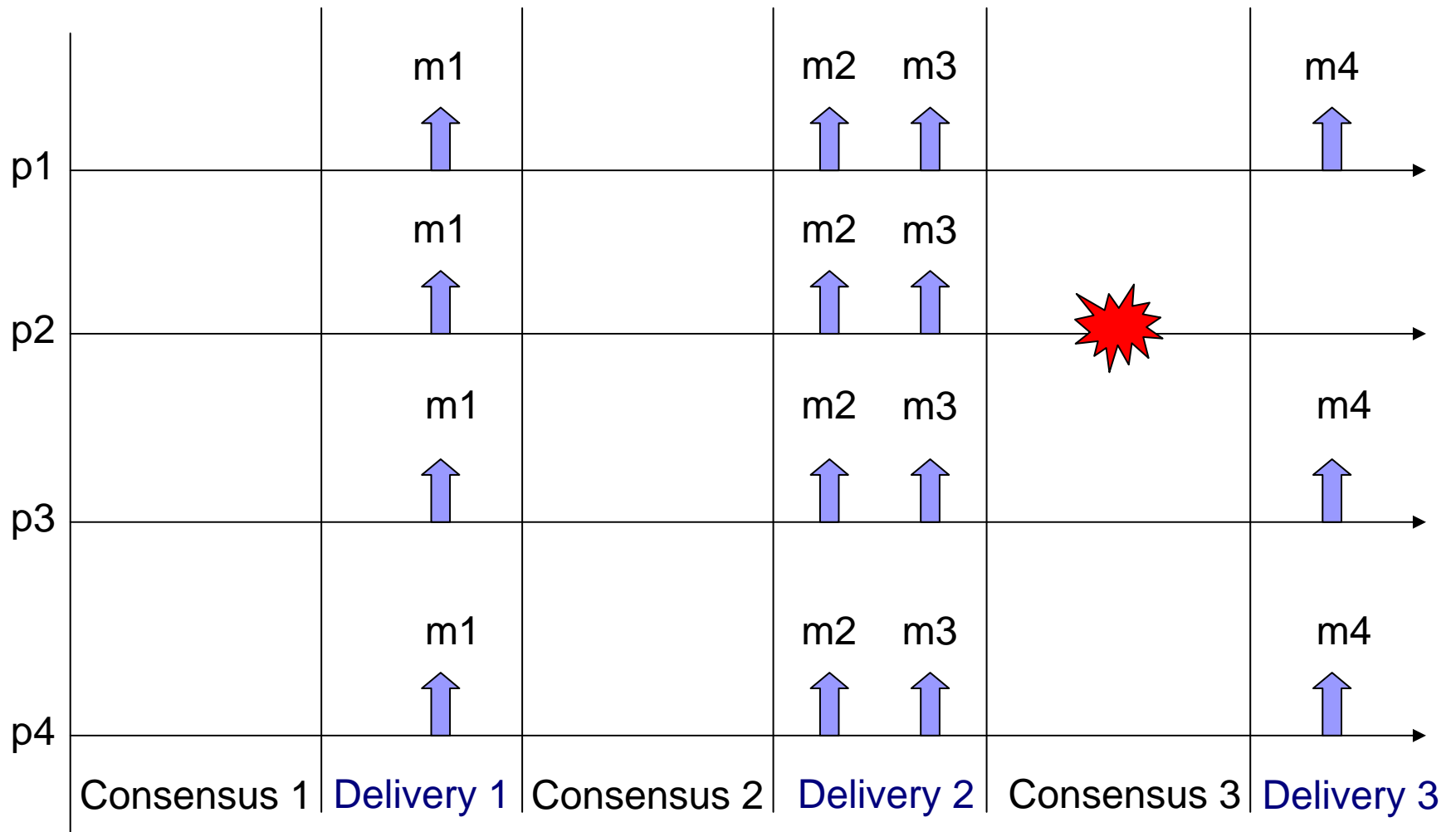
Consensus return m2, m3 and all processes deliver m2 and then m3

# Exercice

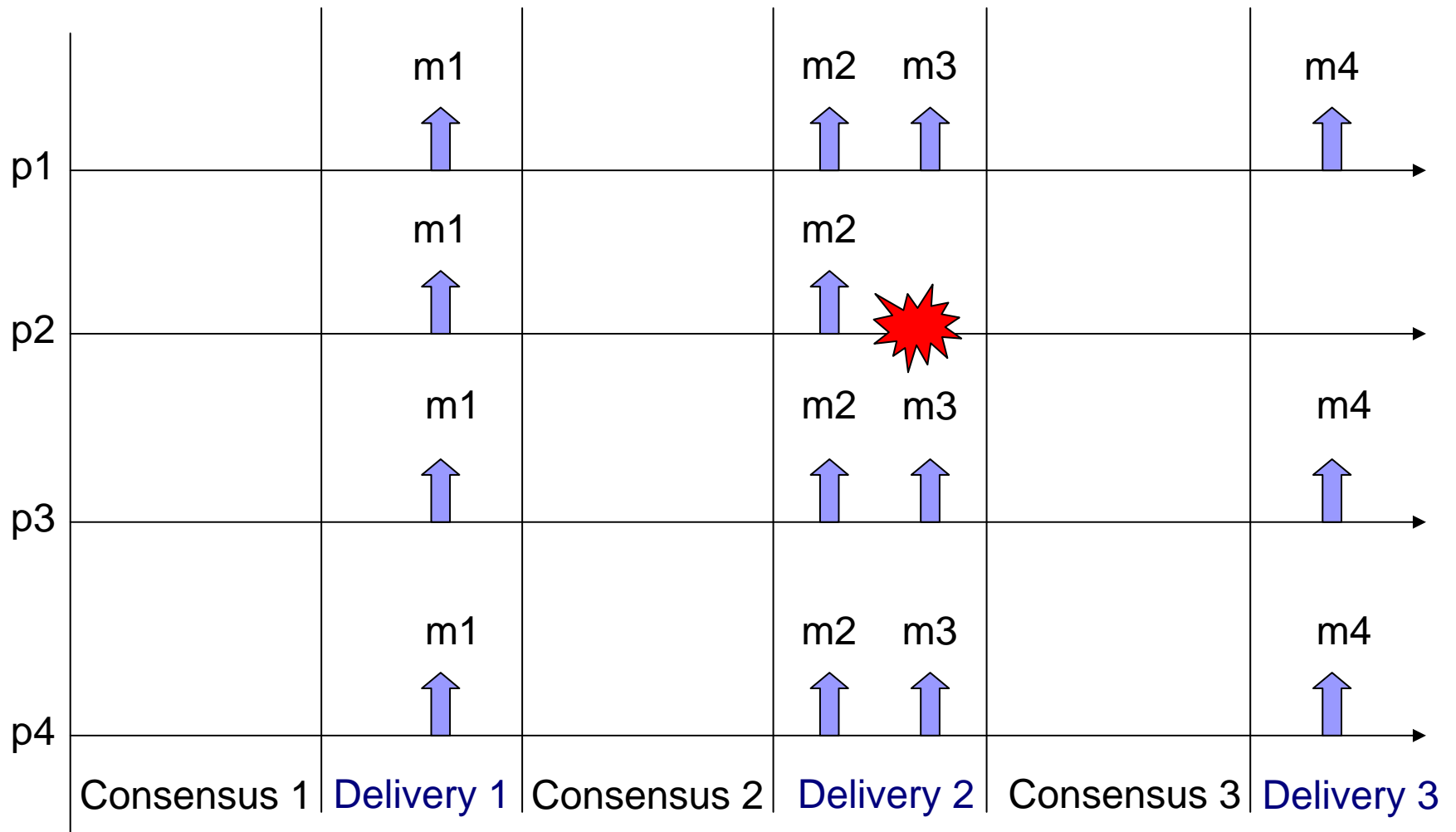
- Which TO specification is satisfied by this algorithm?
- It depends from the assumptions about Reliable Broadcast and Consensus

Consensus	Uniform	Non Uniform
Reliable Broadcast		
Uniform		
Non Uniform		

# Example 1 (UC and URB)



# Example 2 (UC and URB)



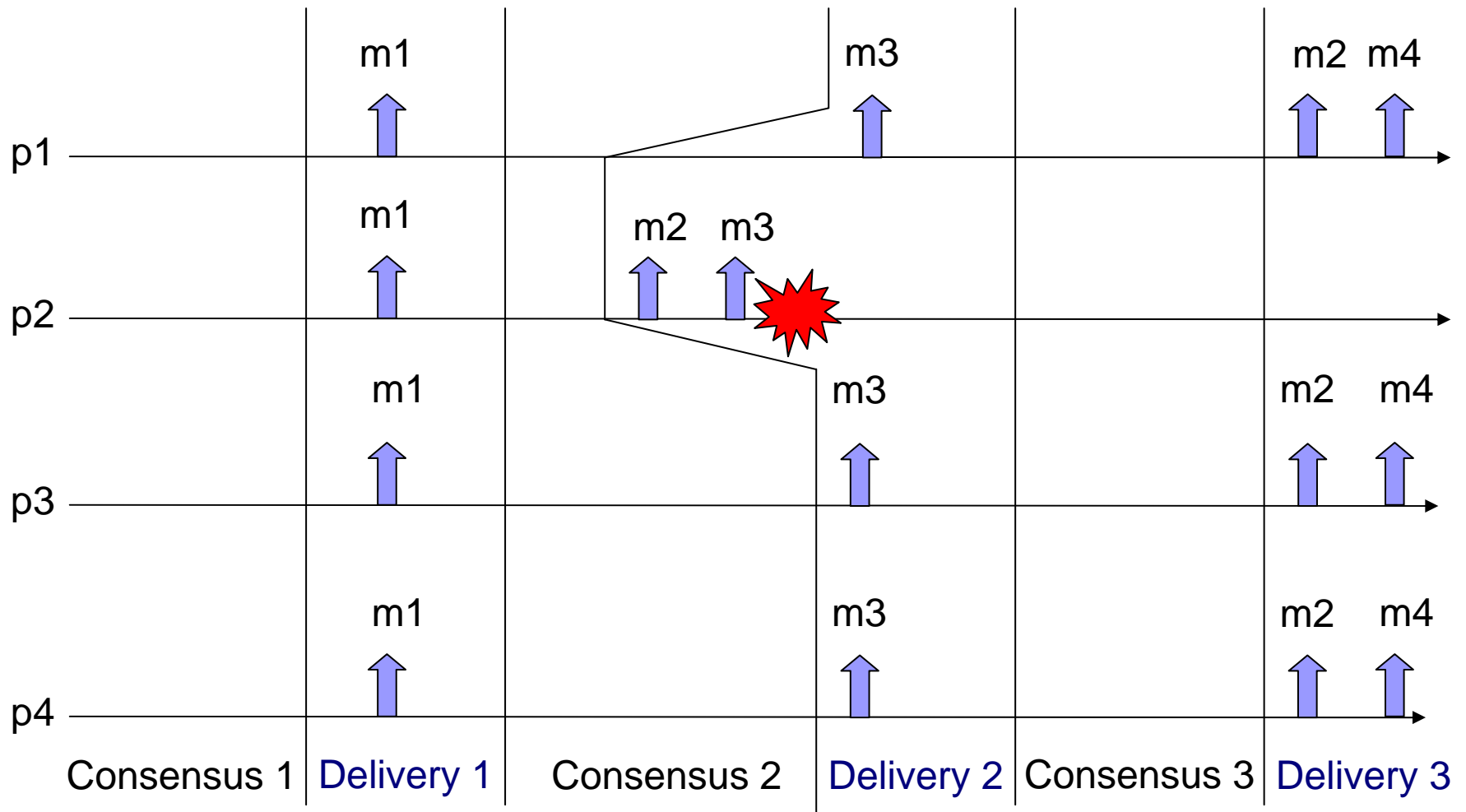
# Uniform Consensus (UC) and Uniform Reliable Broadcast (URB)

- 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

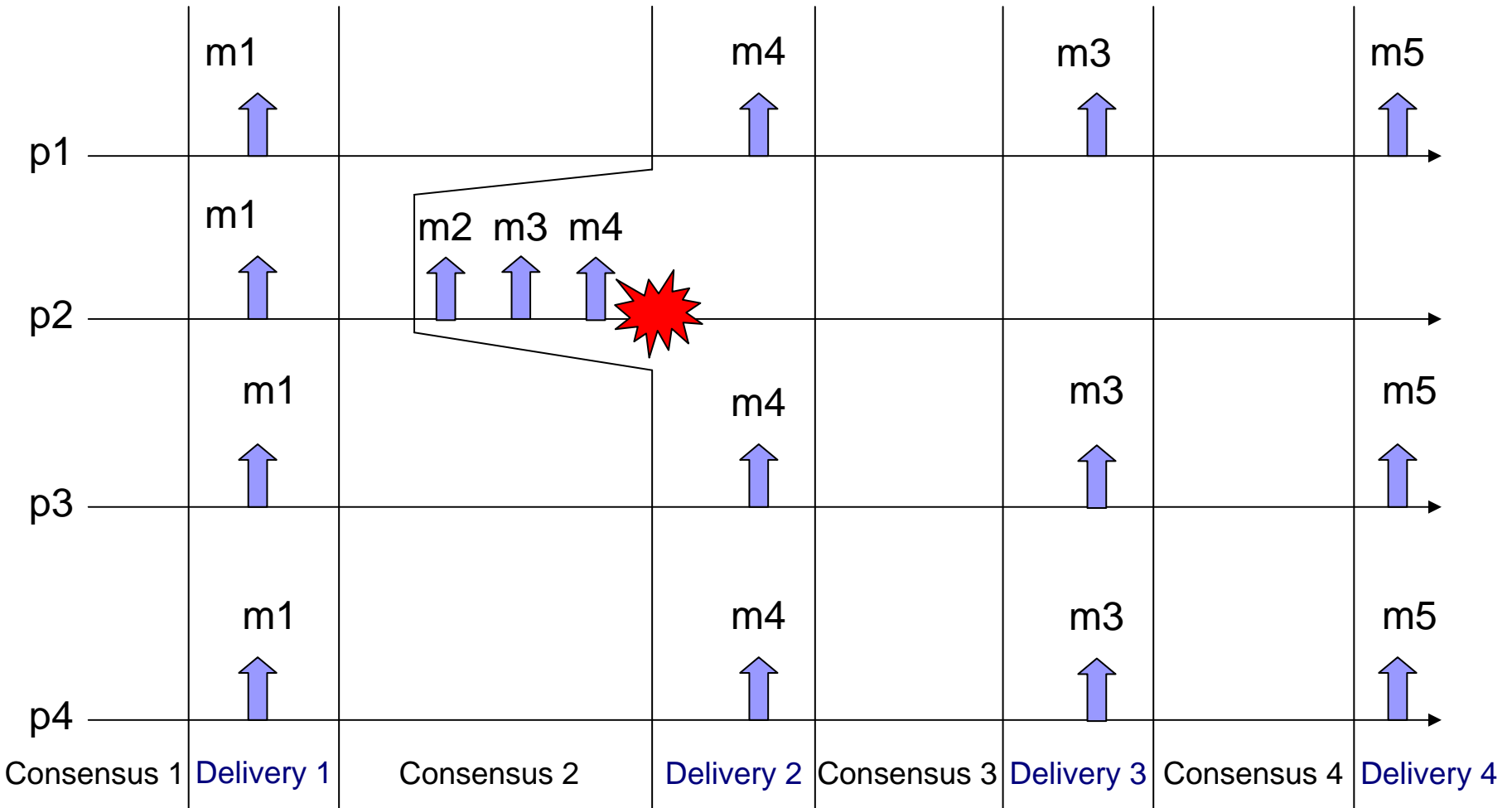
# Example (NUC and URB)



# 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

## Exercise

Which specification is satisfied assuming UC and NURB?