Embedded Operating System
Distributed global state

Graziano Pravadelli

The need of a global state

Problems of traditional OS valid also for distributed systems

- Mutual exclusion
- Starvation
- Deadlock

Complication

- There is no a global state (no global clock, no shared memory)

What is a global state?

Assume to simultaneously interrupt the distributed computation of all processes

The global state is given by

- status of each process
- content of each communication channel

Observation of the global state of a distributed system is simple for an external observer but difficult for an internal node

Global state – Examples of its relevance

Knowledge of the global status is important for

- Checkpoint for rollback in case of failures
- Detection of stable properties
  - “The computation has terminated”
  - “The system is in deadlock”
  - “All tokens in a token-ring network have disappeared”

Global state – Difficulties

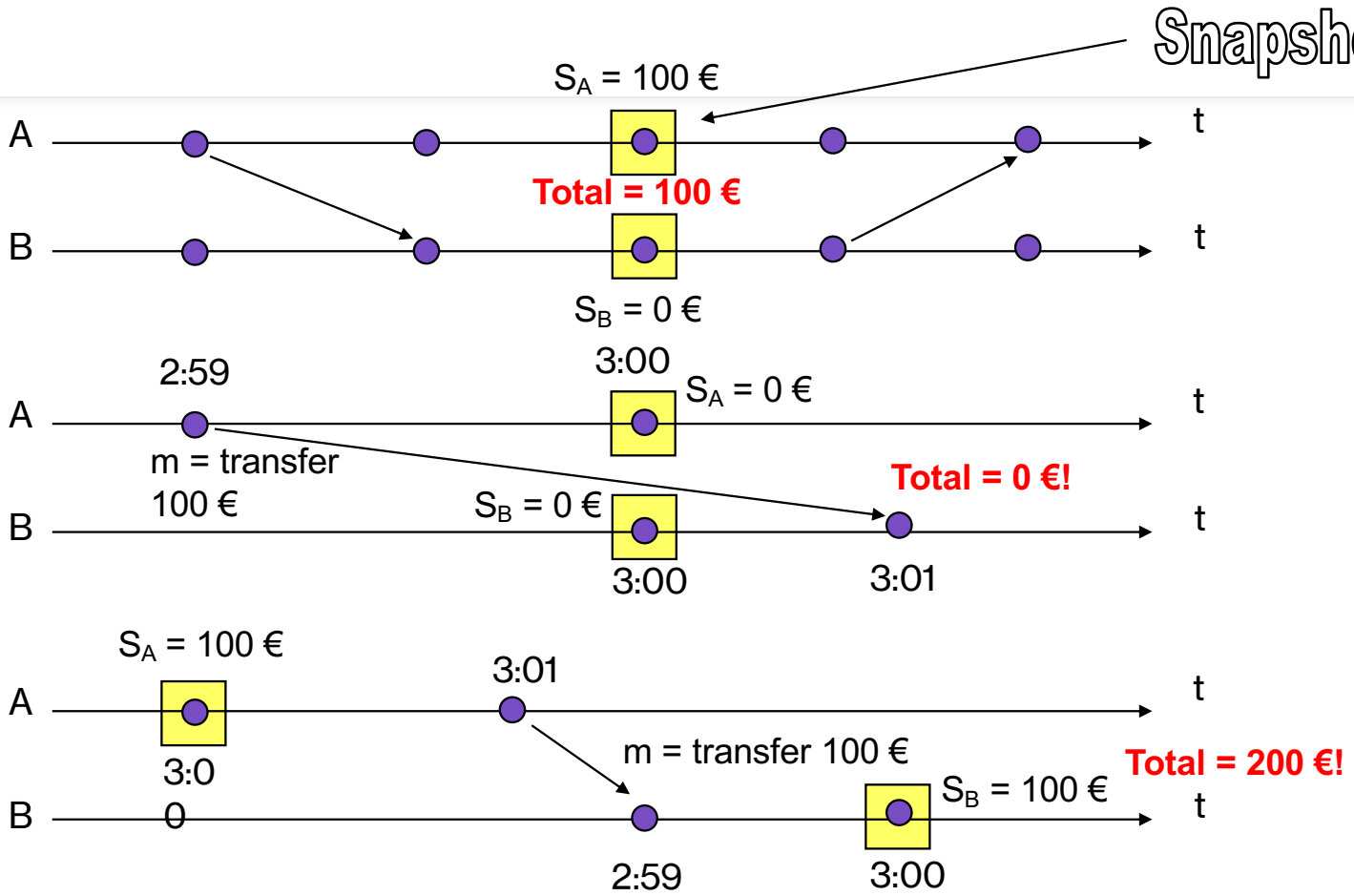
It is necessary to collect information on different machines but we have only local knowledge (no shared memory)

- A process does not know the status of another process

Instant recording is impossible since

- local status recording can not be synchronized based on time (no global clock)
- we know the local status of other processes with a delay (e.g., distant stars)

Global status – What can go wrong?



Definitions – Local state

Channel

- Between P1 and P2 there is a channel if P1 and P2 exchange messages
- It is unidirectional

Local state

- List of *sends* and *receives* in progress
- Defined for each local time (local process)

Snapshot

- It records the status of a process
- It contains records of all messages sent and received on all channels since the last snapshot

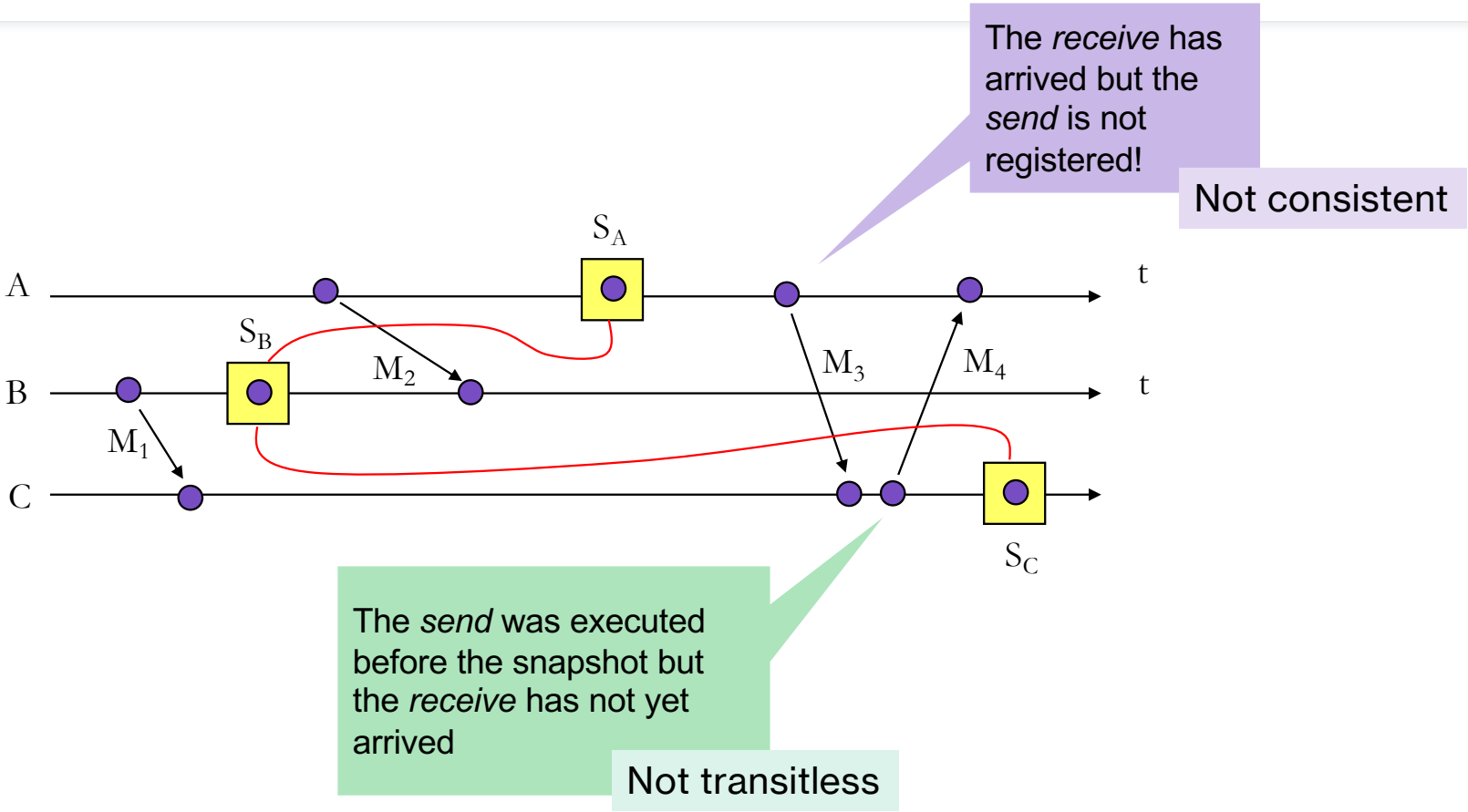
Definitions – Global state

Global state

- Local state of each process + status of communication channels at the time of the snapshot

	Consistent	Transitless	Strongly consistent
Causality between *sends* ans *receives*	• For each received message the corresponding *send* has been recorded	• All *sends* have been received	• Consistent + transitless

Global state – Example

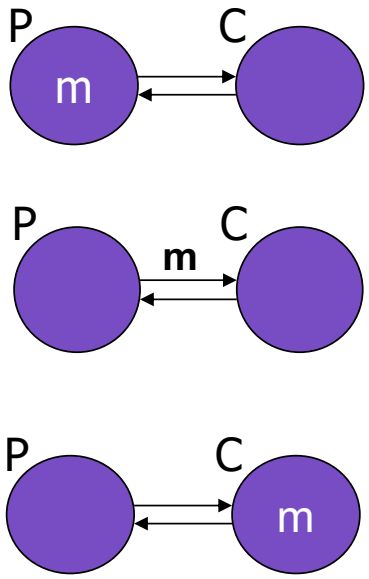


Computation of the global state

- Specific algorithm is required
- Intuitive algorithm
  - Processes record their status in an arbitrary instant
  - A designated process collects various states
- Simple, but not correct
  - Essential to maintain the status of messages and of the channel

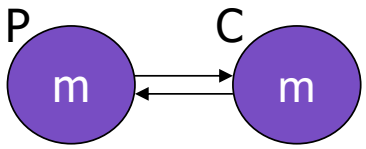
Why the intuitive algorithm does not work?

- Producer-consumer
  - P records its status
  - The message transits on the channel
  - C receives the message and records its status

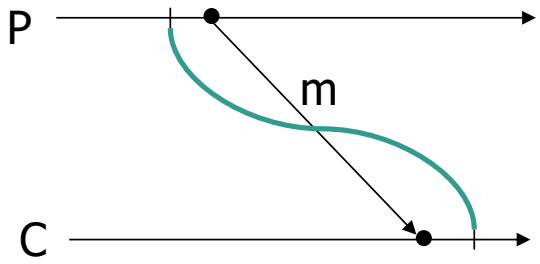


Why the intuitive algorithm does not work?

- Computed global state



- What went wrong?
  - P records its state
  - P does not record the send *sending* for m
  - C records the *receive* for m



Distributed snapshot algorithm - 1
[Chandy&Lamport 85]

Idea: use a special message (*marker*) as a synchronization token

Assumptions

- FIFO channels (implies relations of precedence)
- No loss of messages

by TCP

Algorithm started by an initiator

- It records its local state
- It sends the marker on all its outgoing channels

Atomically

No messages received or sent during these operations

Distributed snapshot algorithm - 2

Behaviour of a generic process

- At the first reception of the marker on a channel
  - It records its local state
  - It labels as *empty* the channel from which the marker was received
  - It propagates the marker on all its outgoing channels
  - It starts listening its ingoing channels
- At successive receptions of the marker on a channel
  - It records the state of the channel
    {sequence of messages received since the first reception of the marker}

Atomically

Distributed snapshot algorithm - 3

- The algorithm ends when all processes received the marker on all ingoing channels
- The collection of the global status requires a further process by the initiator
- It guarantees a consistent global state

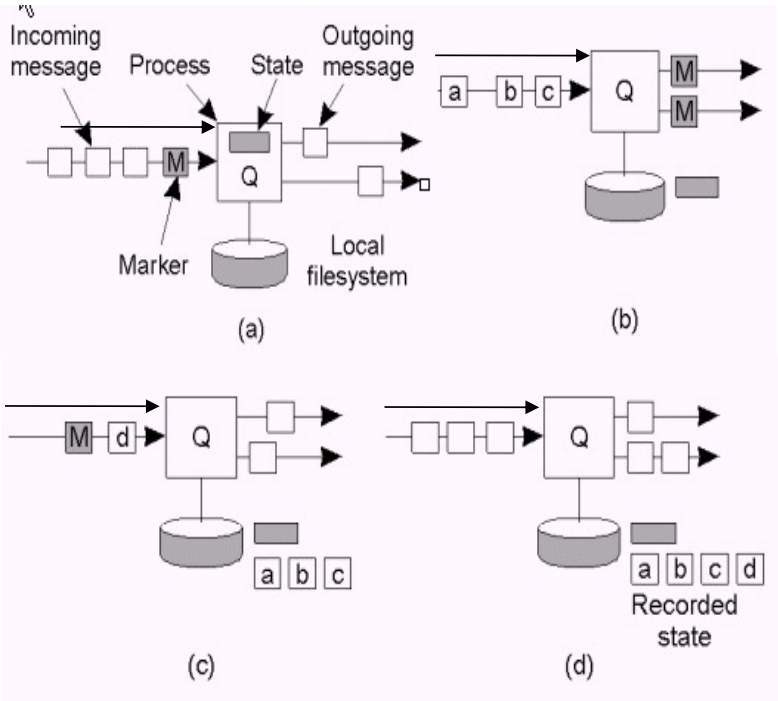
Distributed snapshot

a) First reception of the marker

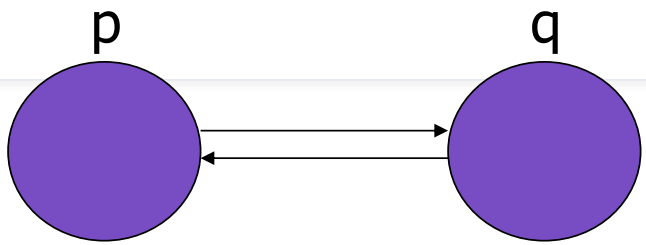
b) Q starts recording its state and sends the marker on all its outgoing channels

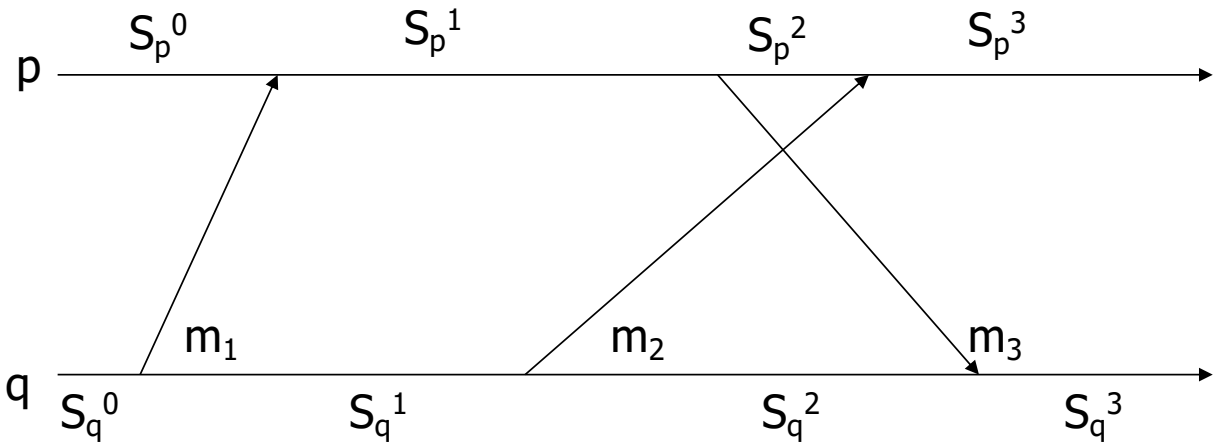
c) Q receives a marker

d) Q sends its state
S = {a,b,c,d}



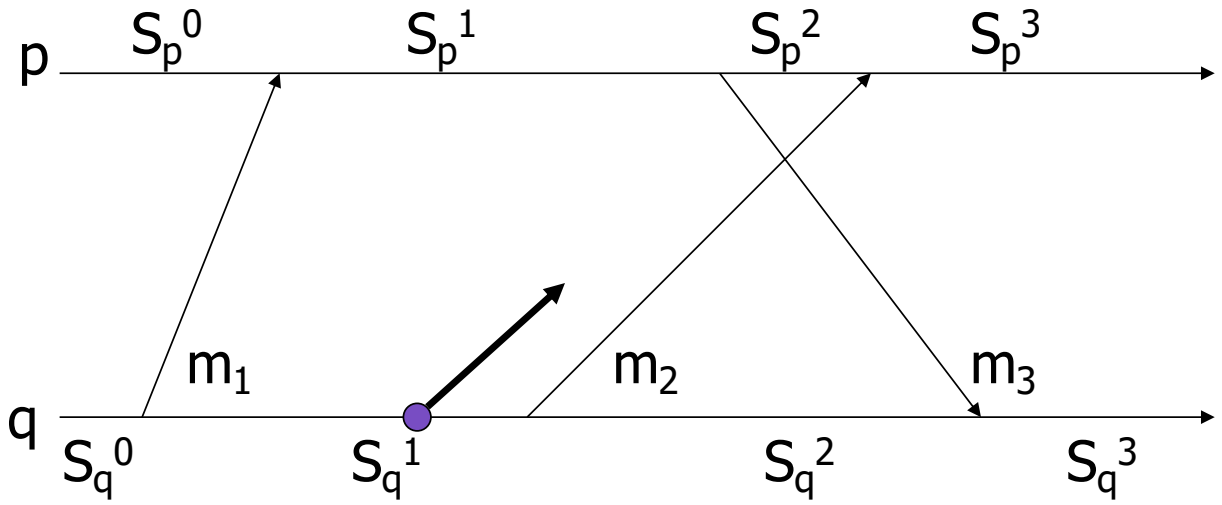
Example of execution – 1





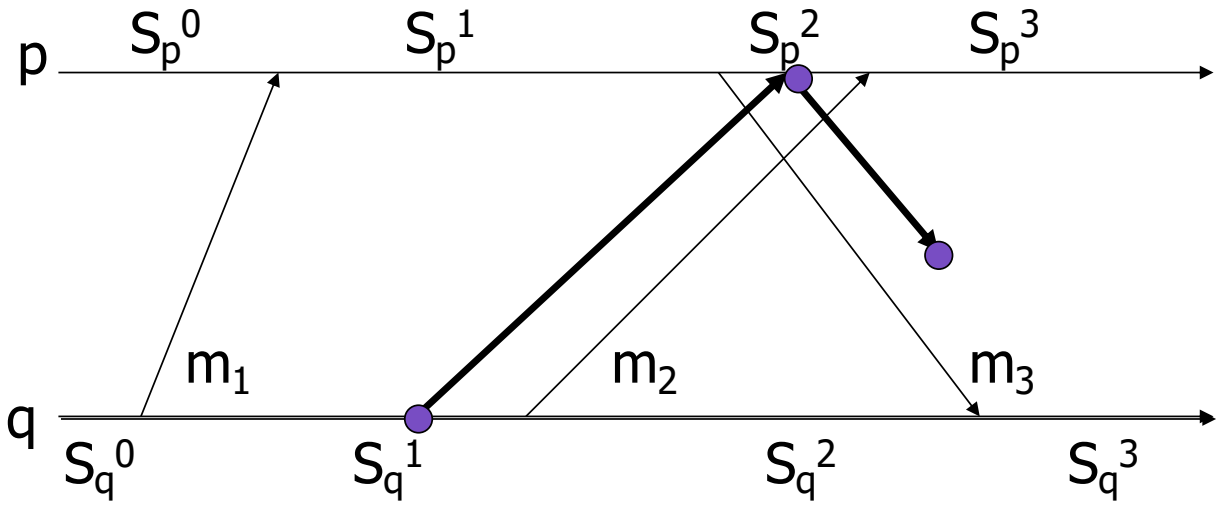
Example of execution – 2

- q starts the algorithm and records the status S_q^1
- q sends the marker to p



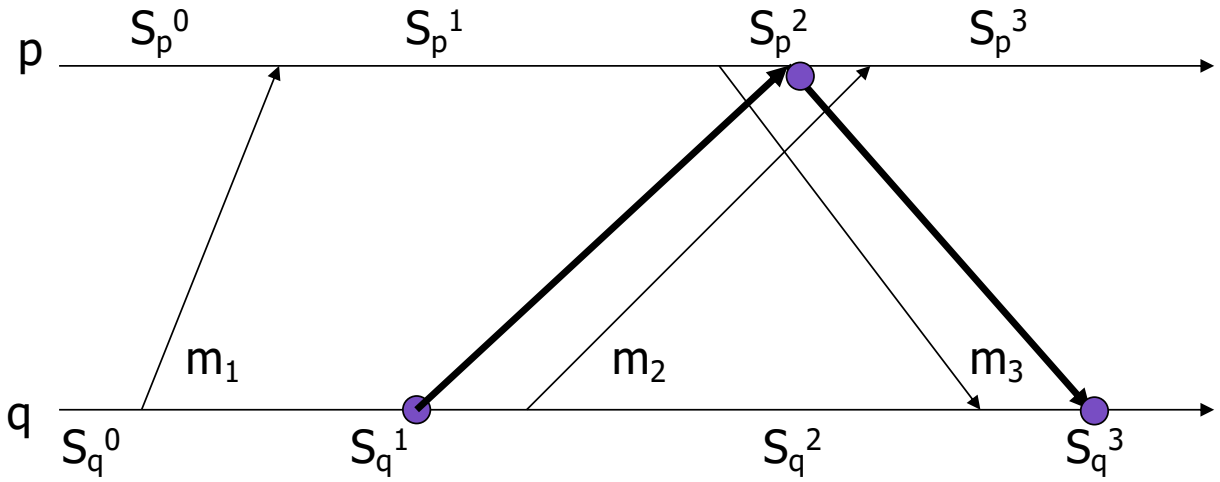
Example of execution – 3

- p receives the marker and records its state as S_p^2
- State of the channel = {}
- p sends back the marker on the output channel
- in the meantime message m_3 is received by q



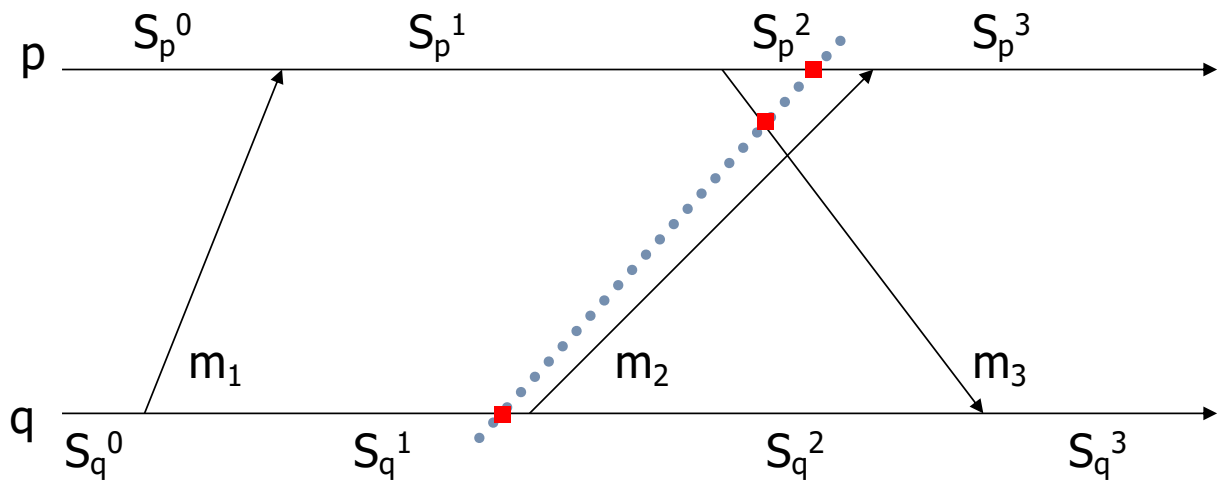
Example of execution – 4

- q receives the marker for the second time
- q records the state of the channel = $\{m_3\}$

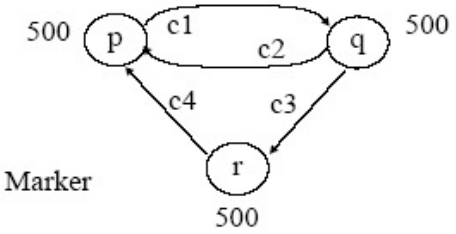


Example of execution (5)

- GLOBAL STATUS = {(S_p^2, S_q^1), (0, m_3)}
- Consistent
  - if receive(m) has been recorded, the same is true for send(m)

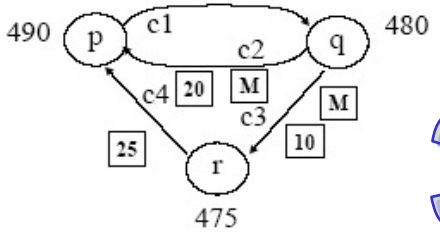


Another example – 1



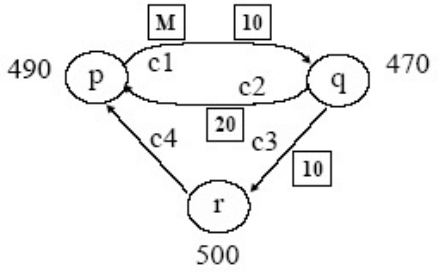
1

Node	Recorded state				
		c1	c2	c3	c4
p			{}		{}
q		{}			
r				{}	



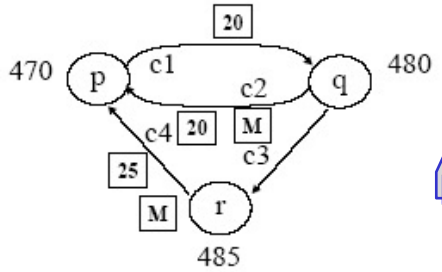
3

Node	Recorded state				
	state	c1	c2	c3	c4
p	490		{}		{}
q	480	{}			
r				{}	



2

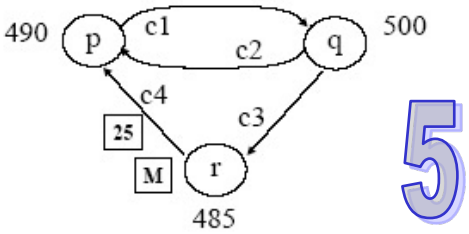
Node	Recorded state				
	state	c1	c2	c3	c4
p	490		{}		{}
q		{}			
r				{}	



4

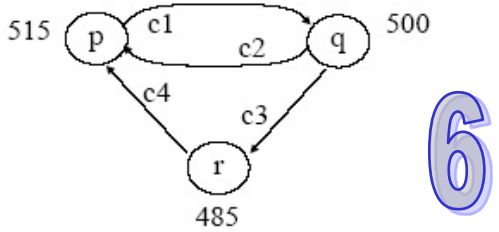
Node	Recorded state				
	state	c1	c2	c3	c4
p	490		{}		{}
q	480	{}			
r	485			{}	

Another example – 2



5

Node	Recorded state				
	state	c1	c2	c3	c4
p	490		{20}		{}
q	480	{}			
r	485			{}	



6

Node	Recorded state				
	state	c1	c2	c3	c4
p	490		{20}		{25}
q	480	{}			
r	485			{}	

Distributed snapshot – Property

The distributed snapshot algorithm guarantees that			
if S_i is the global state at the beginning of the algorithm	and S_j it the global state at the end of the algorithm	and S^* is the global state recorded by the algorithm	then S^* is reachable from S_i and S_j is reachable from S^*