Global States and Distributed Snapshot Algorithms

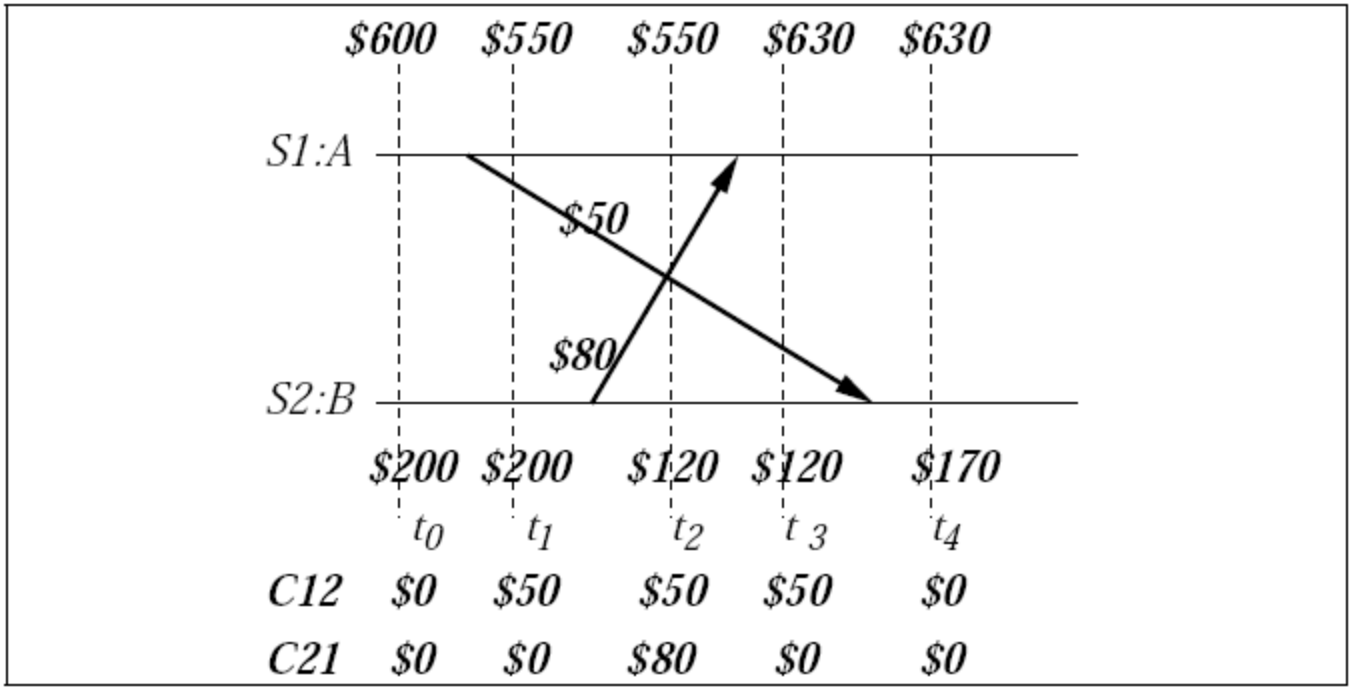
Pradipta De
pradipta.de@sunykorea.ac.kr

Global State

- State of a process: local memory and history of activities
- State of a channel: sequence of messages sent along the channel, excluding messages received along the channel

- *The global state of a distributed computation is the set of local states of all individual processes involved in the computation plus the state of the communication channels*

Example



What happens if we record the states as follows?
t0: record state of A
t2: record state of B, and channels C12, C21

Property of Consistent Global State

A global cut is consistent iff it satisfies two conditions:

C1: $\text{send}(m_{ij}) \in LS_i : m_{ij} \in SC_{ij} \oplus \text{rec}(m_{ij}) \in LS_j$. ($\oplus$ is Ex-OR operator.)

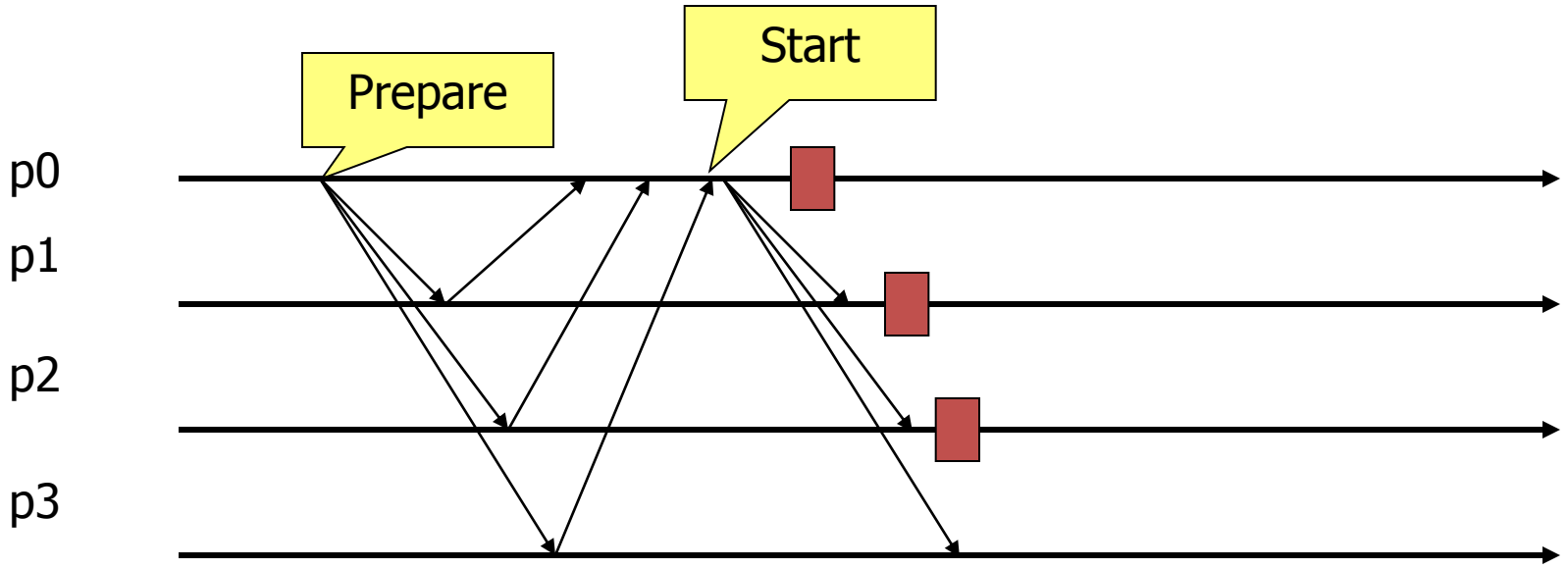
C2: $\text{send}(m_{ij}) \notin LS_i : m_{ij} \notin SC_{ij} \wedge \text{rec}(m_{ij}) \notin LS_j$.

Condition C1 ensures that "conservation of messages" ➔ every message that is send by process p_i must be recorded either in transit, or received by process p_j

Condition C2 ensures that for every effect the cause is present.

Snapshot of Global State

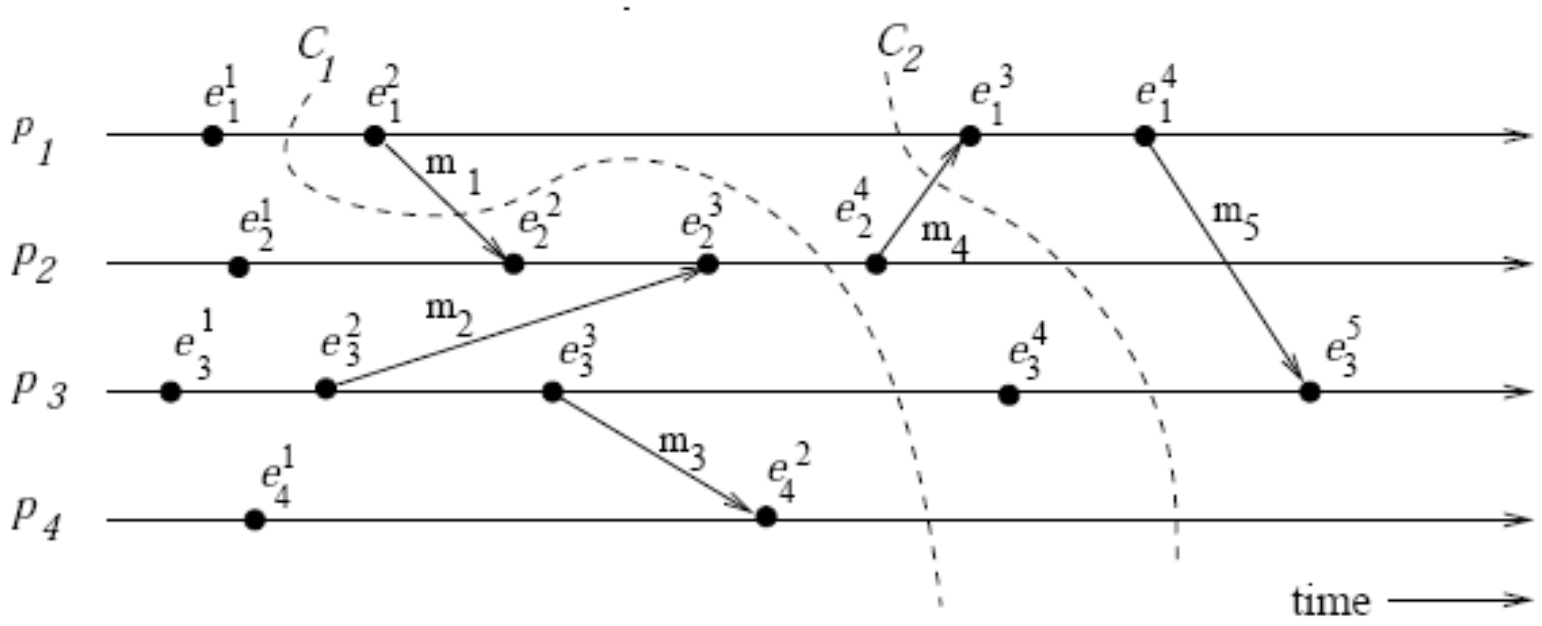
Coordinate the process across all process
Send a Start message; stop all events in each process; take a snapshot



Coordinated Blocking of all processes
Latency of snapshot technique is high
Is blocking unavoidable ?

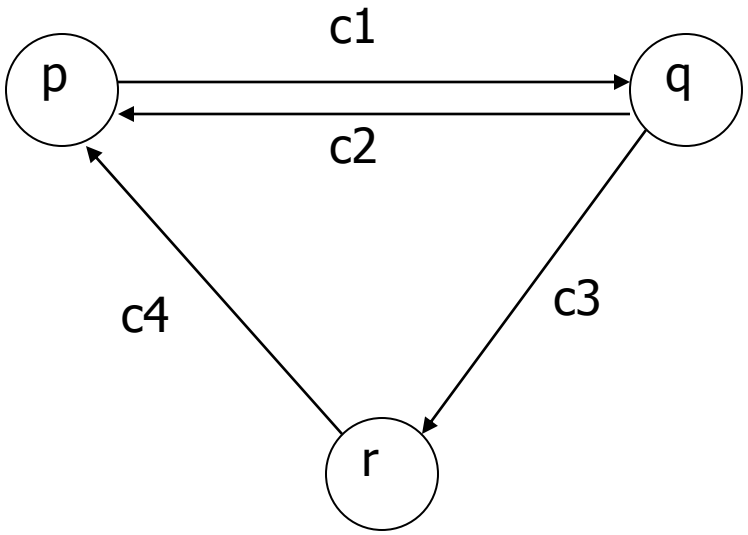
Consistent Cut

- A cut in a space-time diagram is a line joining an arbitrary point on each process line that slices the space-time diagram into a PAST and a FUTURE.
- A consistent global state corresponds to a cut in which every message received in the PAST of the cut was sent in the PAST of that cut.



System Model

- Processes: p, q, r
- Channels: c1, c2, c3, c4 (FIFO, directed, reliable)



A process is defined by the set of states; transitions from one state to another state when an event e occurs

Illustration: Deterministic Case

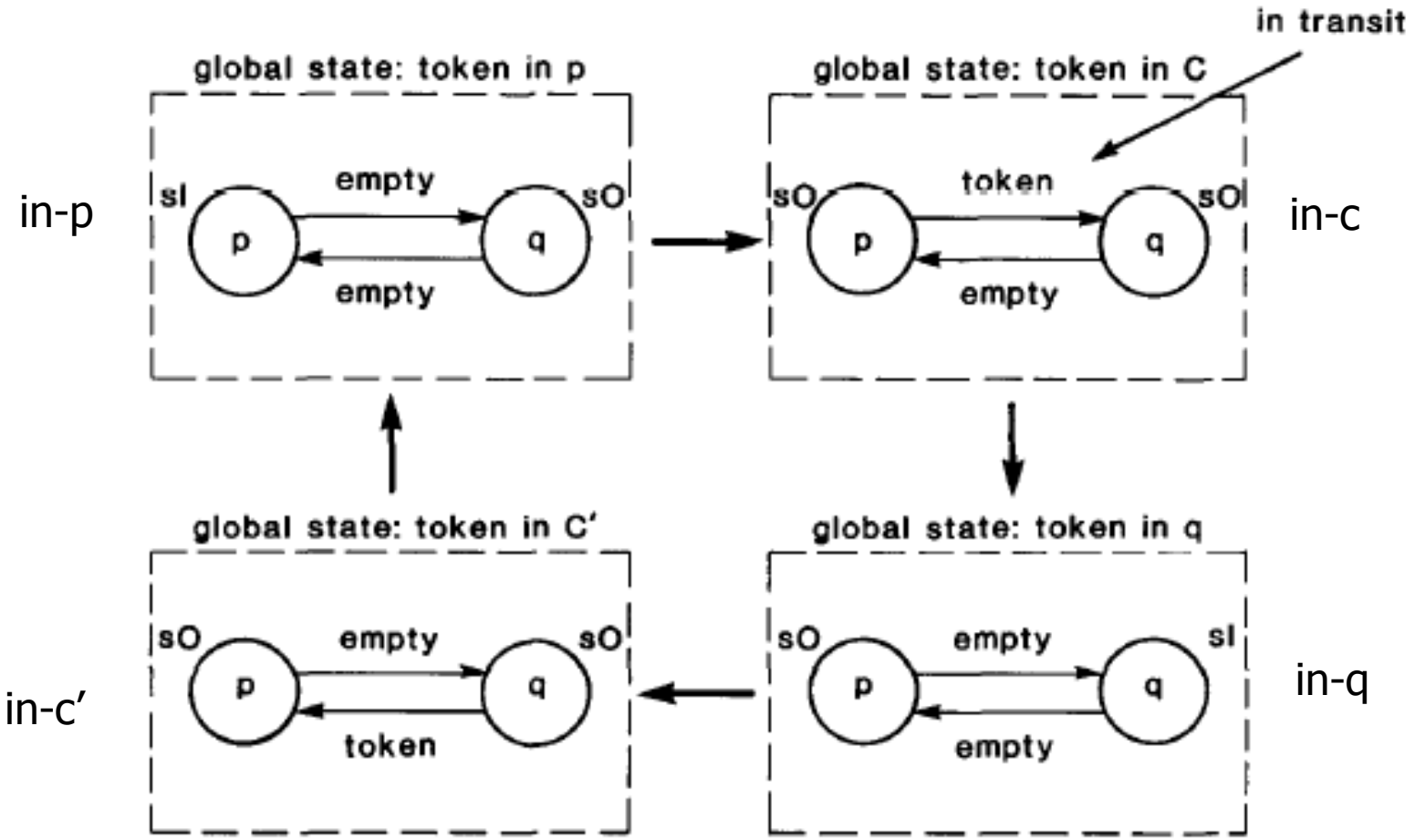
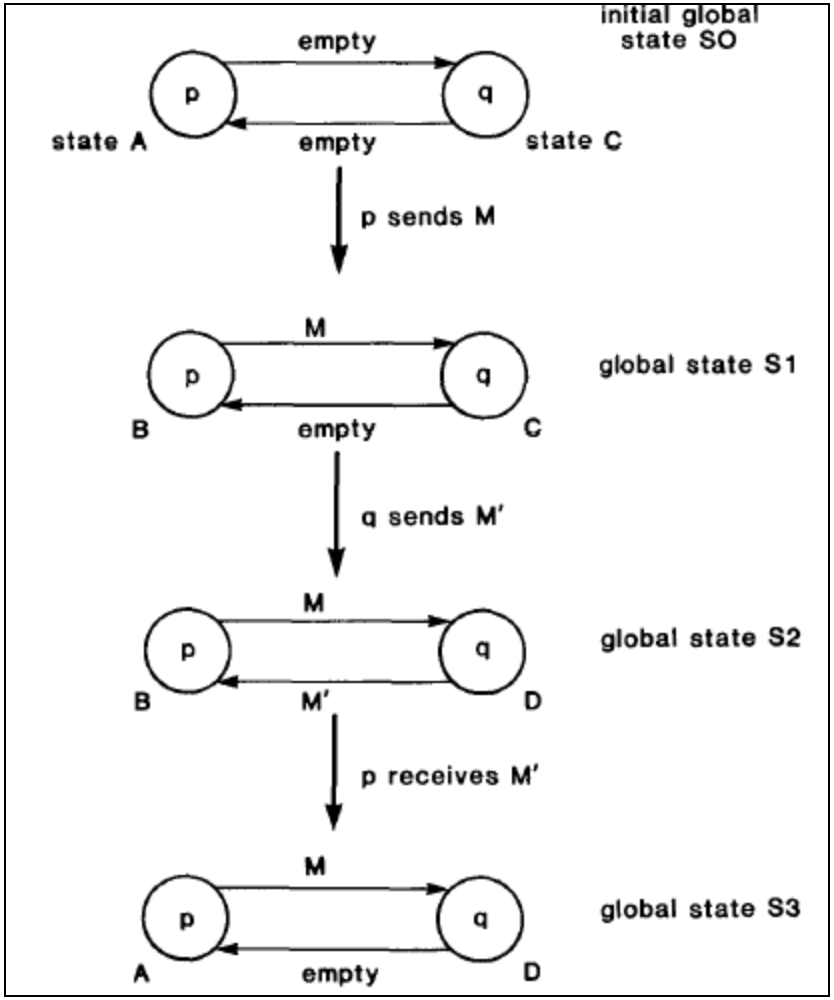


Fig. 4. Global states and transitions of the single-token conservation system.

Non-deterministic Case



- Multiple transitions possible from a global state
  - "P sends M" or "Q sends M"

Intuition: When to snapshot ?

Scenario: Snapshot global state by recording process state before sending a message, and channel state after sending a message

- Record state of process p in in-p ➔ token in p
- Next "p sends a token" ➔ move to state in-c
- Snapshot: one token in p, and one in c
- Inconsistent since state of p recorded "**before** p sent a message and state of c recorded **after** p sent the message"

Intuition: When to snapshot ?

Scenario: Snapshot global state by recording process state before sending a message, and channel state after sending a message

- Record state of channel c in in-p → no token in c
- "p sends a token" → moves to state in-c … record global states of p, q, c' → no token in p,q,c'
- Snapshot: No token in global state

- Inconsistent since state of c recorded "**before** p sent a message and state of p recorded **after** p sent the message"

Global Snapshot : Challenges

Without a global clock, how to decide when to snapshot ?

I1: How to distinguish between the messages to be recorded in the snapshot (either in a channel state or a process state) from those not to be recorded. The answer to this comes from conditions **C1** and **C2** as follows:

Any message that is sent by a process before recording its snapshot, must be recorded in the global snapshot (from **C1**).

Any message that is sent by a process after recording its snapshot, must not be recorded in the global snapshot (from **C2**).

I2: How to determine the instant when a process takes its snapshot. The answer to this comes from condition **C2** is as follows:

A process p_j must record its snapshot before processing a message m_{ij} that was sent by process p_i after recording its snapshot.

Solution Criterion

- In every state, the number of messages received along a channel cannot exceed number of messages sent along a channel

- The state of channel c that is recorded must be the sequence of messages sent along channel before sender's state is recorded, excluding the seq of messages received along the channel before the receiver's state is recorded

Chandy-Lamport Algorithm

- **Marker Sending Rule for process i**
  - Process i records its state
  - For each outgoing channel C on which a marker has not been sent, i sends a marker along C before i sends further messages along C.

- **Marker Receiving Rule for process j**

  On receiving a marker along channel C,

  if j has not recorded its state then

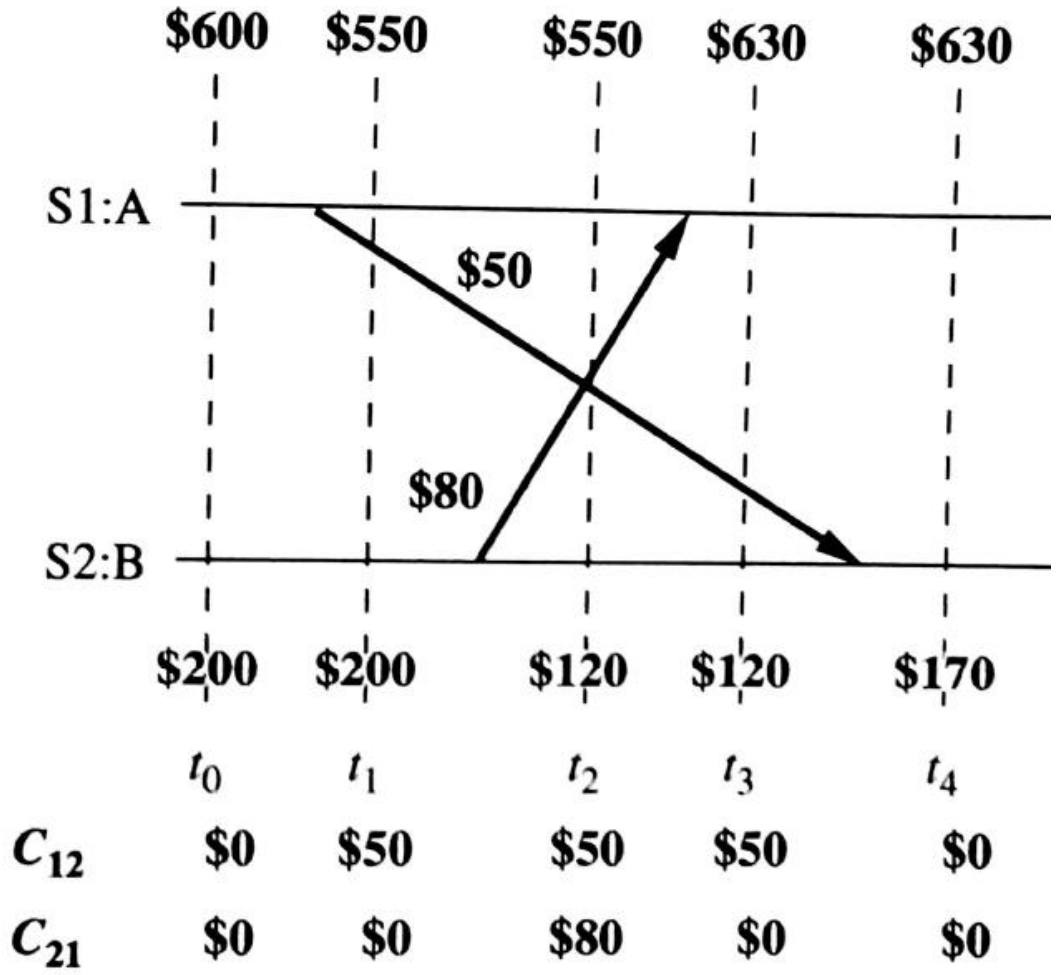
  Record the state of C as the empty set

  Follow the "Marker Sending Rule"

  else

  Record the state of C as the set of messages received along C after j 's state was recorded and before j received the marker along C

Example



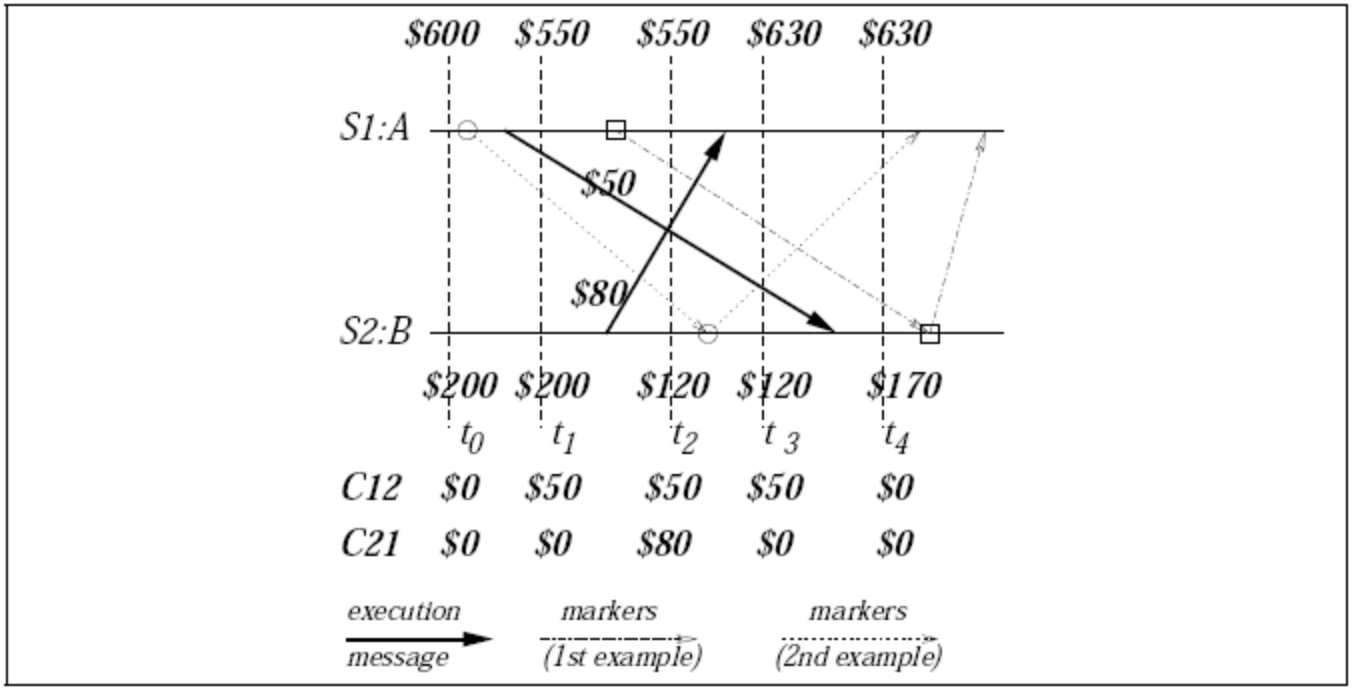
t_0: record A
t_2: record B, C_{12}, C_{21}
Consistent Snapshot?

t_1: record A
t_2: record B, C_{12}, C_{21}
Consistent Snapshot?

t_2: record A
t_4: record B, C_{12}, C_{21}
Consistent Snapshot?

Recorded Global State: Properties

Recorded global state may not correspond to any of the global states that occurred during computation



Recorded Global State: Properties

- If S_i and S_j are the global state when Lamport's algorithm started and finished respectively and S^* is the state recorded by the algorithm then,
  - S^* is reachable from S_i
  - S_j is reachable from S^*

Other Techniques

- Non-FIFO channel
  - Lai-Yang Algorithm
- Causal Order Channel
  - causal message delivery property provides a built-in message synchronization to control and computation messages, leading to simplified algorithms
    - Acharya-Badrinath algo
    - Alagar-Venkatesean algo