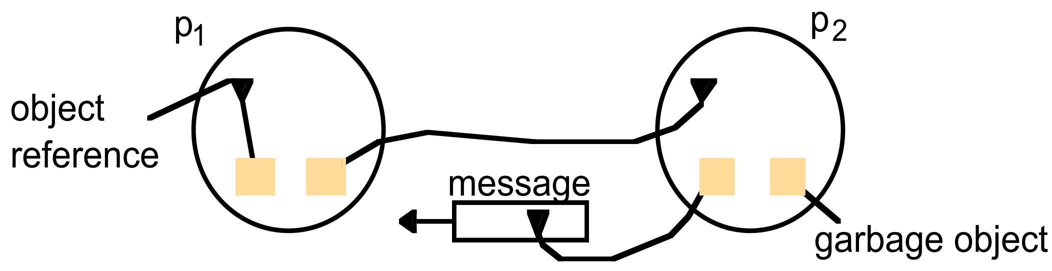
Snapshot Algorithm

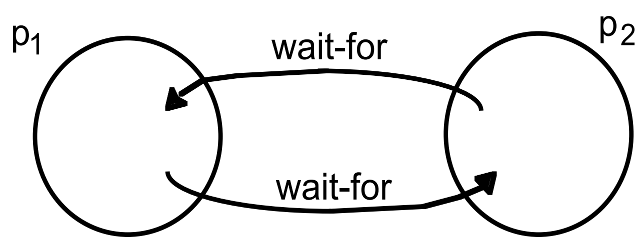
Yao Liu

Detecting global properties

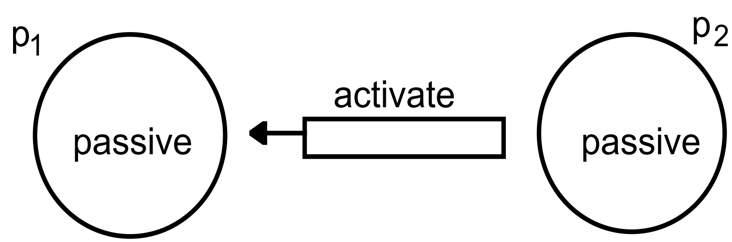
a. Garbage collection



b. Deadlock



c. Termination



Global snapshot

- Individual state of each **process** in the distributed system +
- Individual state of each **communication channel** in the distributed system

- Capture the instantaneous state of each process
- as well as the instantaneous state of each communication channel
  - i.e., messages in transit on the channels

If we have perfectly synchronized clocks

- Synchronize clocks of all processes
- Ask all processes to record their states at future time *t*

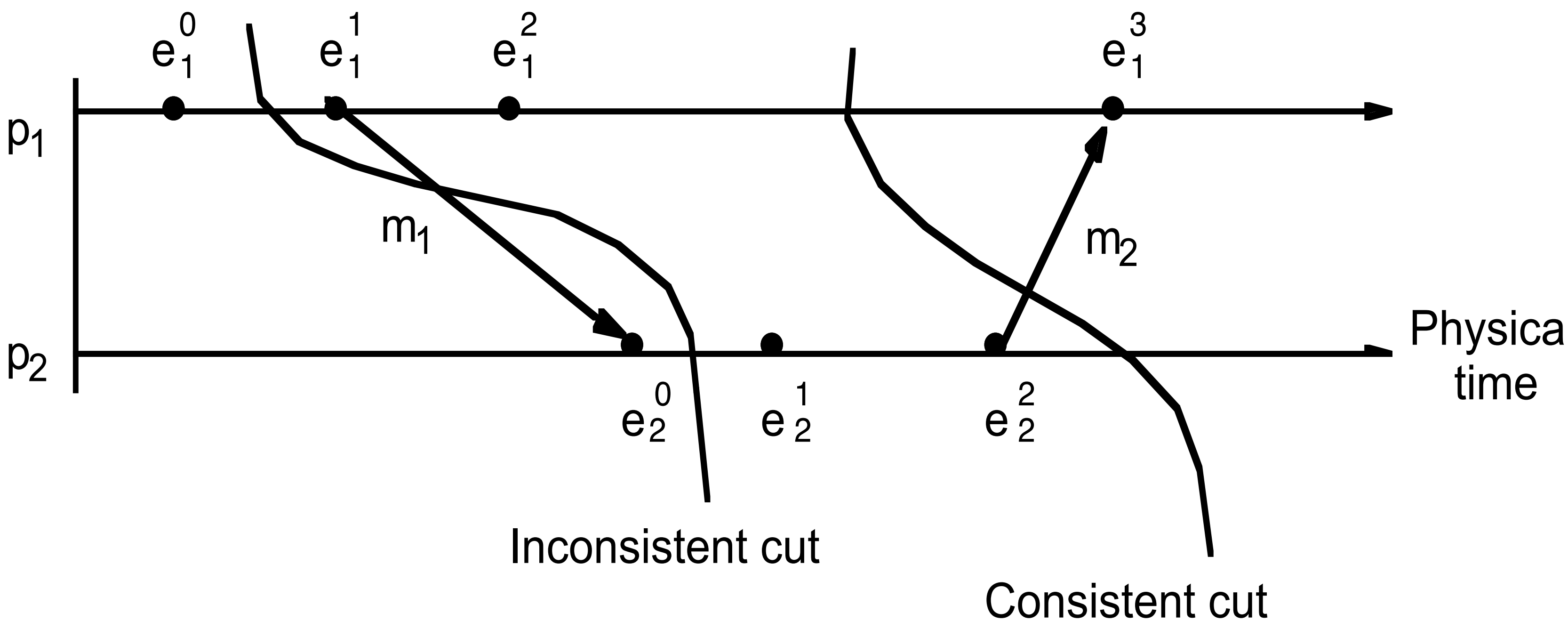
- However …
  - There are no perfectly synchronized clocks
  - Messages in transit are not recorded

- In fact, synchronized is not required. Causality is enough.

Global states and consistent cuts

- How to capture a meaningful global state from local states recorded at different real times?
- Each process records events that correspond to internal actions, e.g., updating a variable, and the sending or receipt of a message
- A cut is a subset of the system's global history that is a union of prefixes of process histories
- A cut is **consistent** if for each event it contains, it also contains all events that happened before that event

Consistent and inconsistent cuts



Chandy and Lamport's snapshot algorithm

- Goal:
  - record a set of process and channel states for a set of processes
  - such that even if the combination of recorded states may never have occurred at the same time,
  - the recorded state is consistent
- State recorded locally at processes

Assumptions

- Neither channels nor processes fail; communication is reliable
- Channels are unidirectional and provide FIFO message delivery
- The graph of processes and channels is strongly connected (there is a path between any two processes)
- Any process may initiate a global snapshot at any time
- Processes may continue with their execution and send and receive normal messages while the snapshot takes place

Chandy and Lamport's snapshot algorithm

- For each process, **P*i*** :
  - Incoming channels from which **P*i*** receives messages
  - Outgoing channels to which **P*i*** sends messages
- Each process must record both its local process state as well as states of all its incoming channels
- Marker messages
  - A special message used in the algorithm

Initiate the snapshot

- **P*i*** initiates the snapshot by first recording its own local state
- for *j=1 to N* except *i*
  - **P*i*** sends out a Marker message on outgoing channel $\mathbf{C}_{ij}$
- Starts recording the incoming messages on each of the incoming channels at **P*i***: $\mathbf{C}_{ji}$ (for *j=1 to N* except *i*)

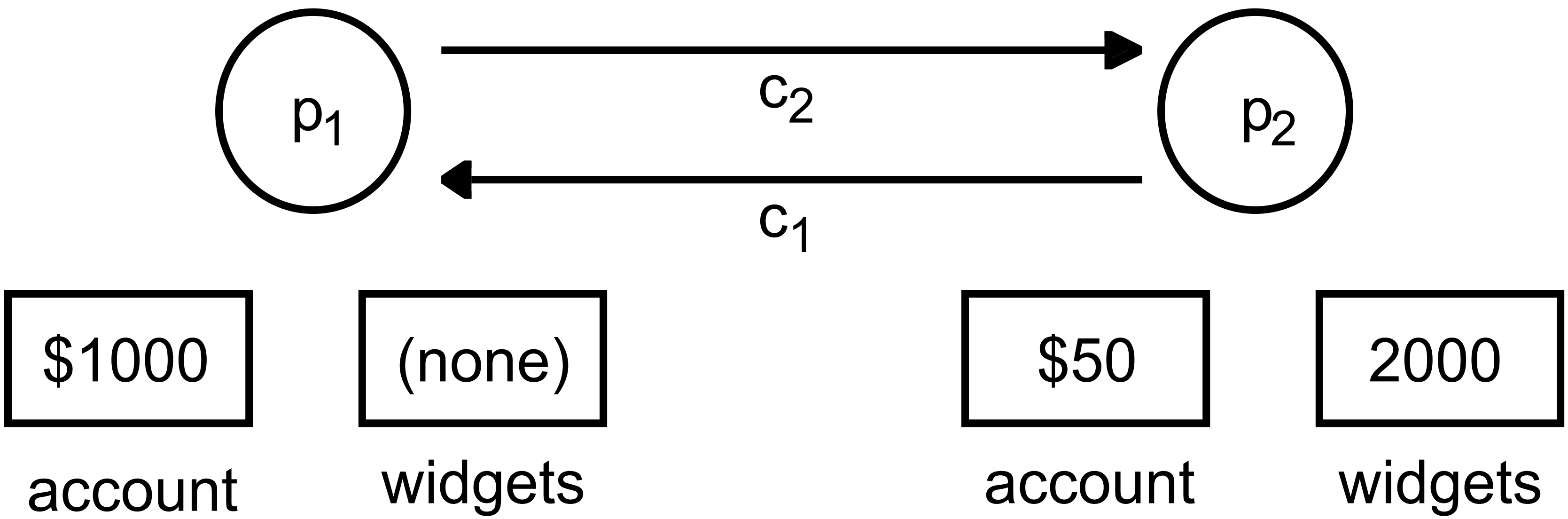
Marker receiving rule

Whenever a process P*i* receives a Marker message on an incoming channel C_{ji} ($P_j \rightarrow P_i$)

- **if** (this is the first Marker **P*i*** is seeing)
  - **P*i*** records its own state first
  - Marks the state of channel C_{ji} as "empty"
  - for *j=1 to N* except *i*
    - **P*i*** sends out a Marker message on outgoing channel C_{ij}
  - Starts recording the incoming messages on each of the incoming channels at **P*i***: C_{ji} (for *j=1 to N* except *i*)
- **else // already seen a Marker message**
  - Mark the state of channel C_{ji} as all the messages that have arrived on it since recording was turned on for C_{ji}
  - Stop recording on channel C_{ji}

Two processes and their initial states

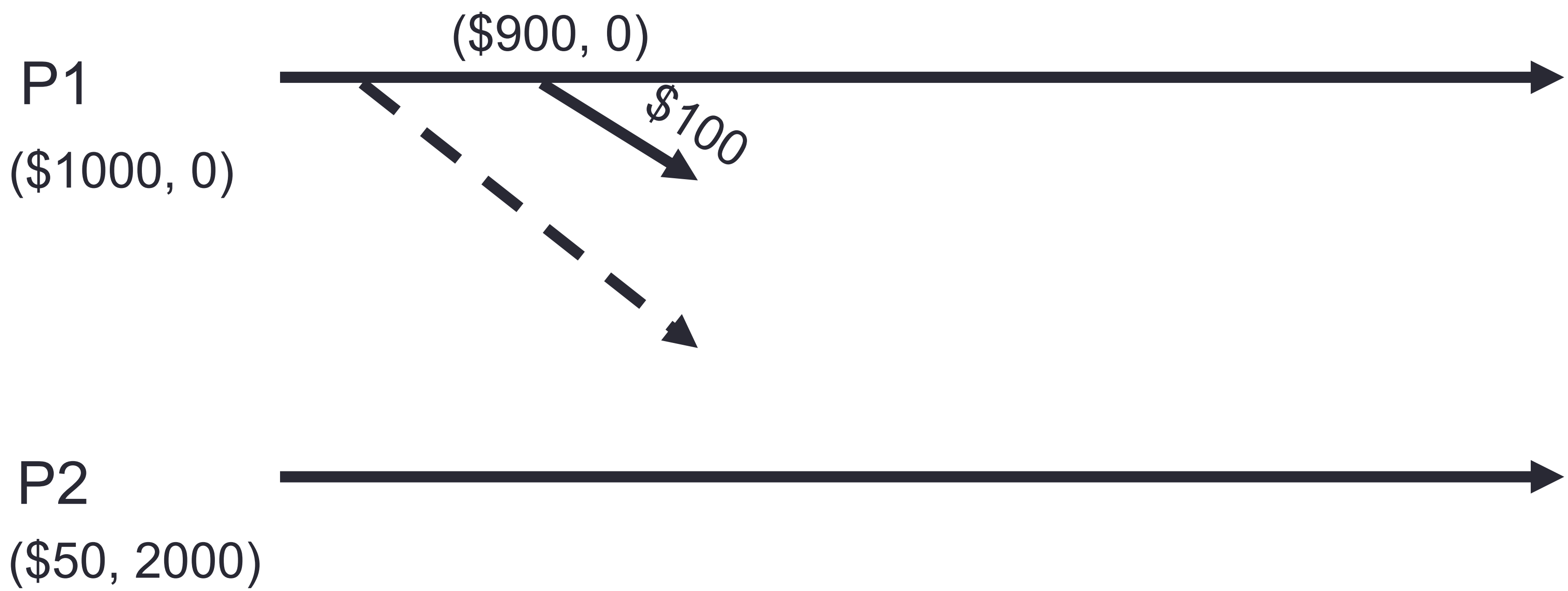
- Assume that p_2 has already received an order for five widgets, which it will shortly dispatch to p_1



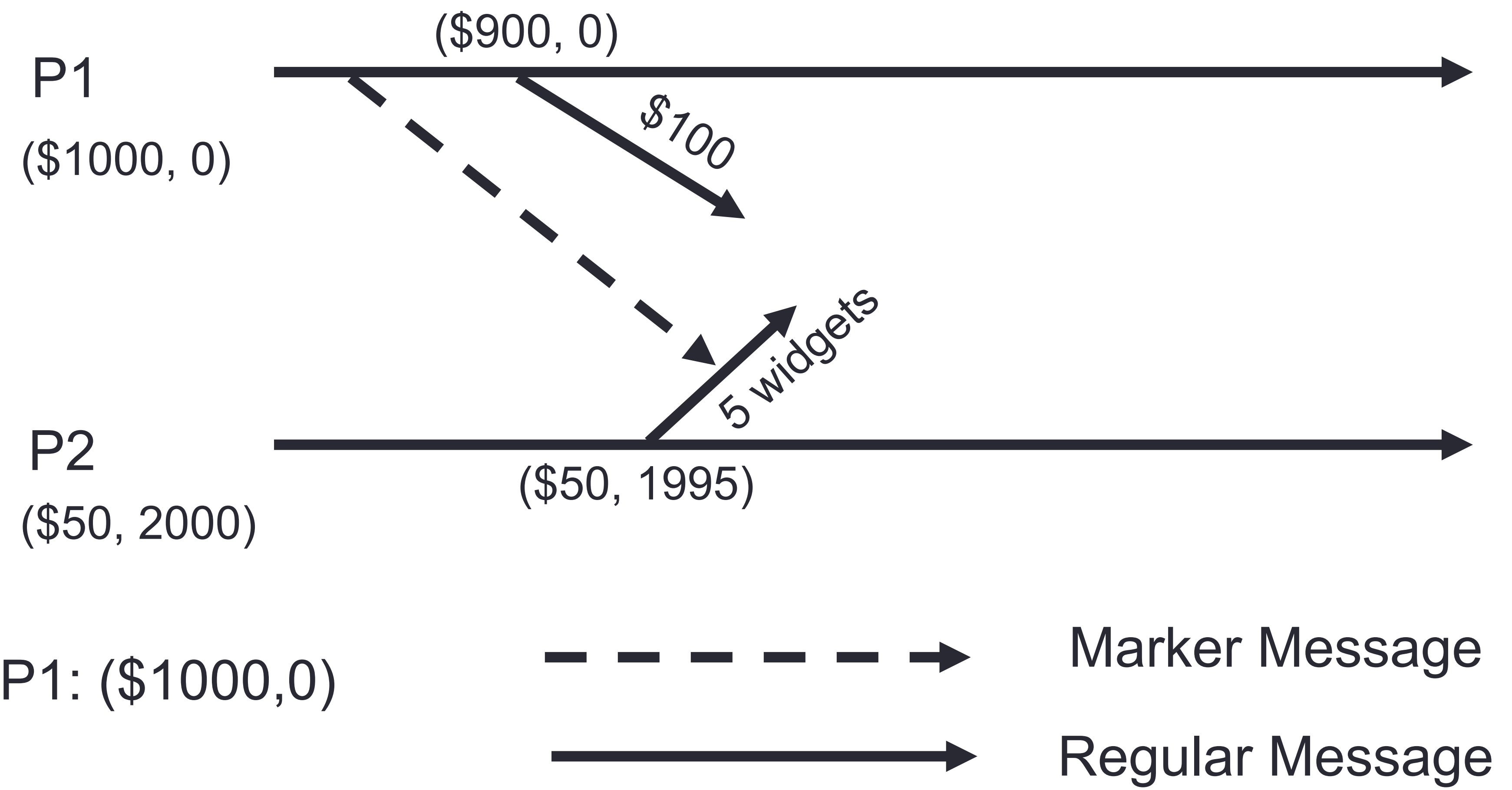
P1: ($1000,0)

Marker Message

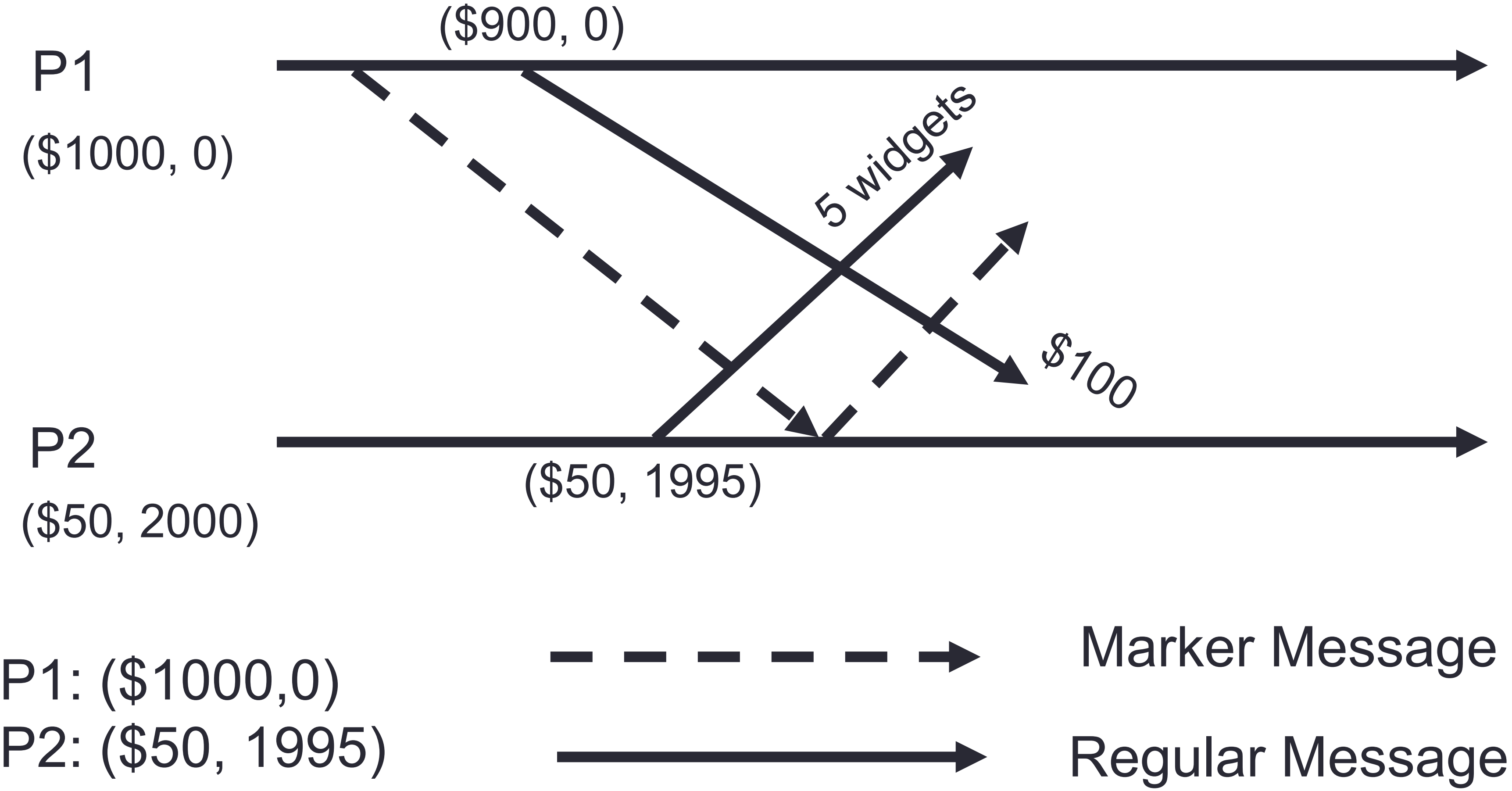
Regular Message



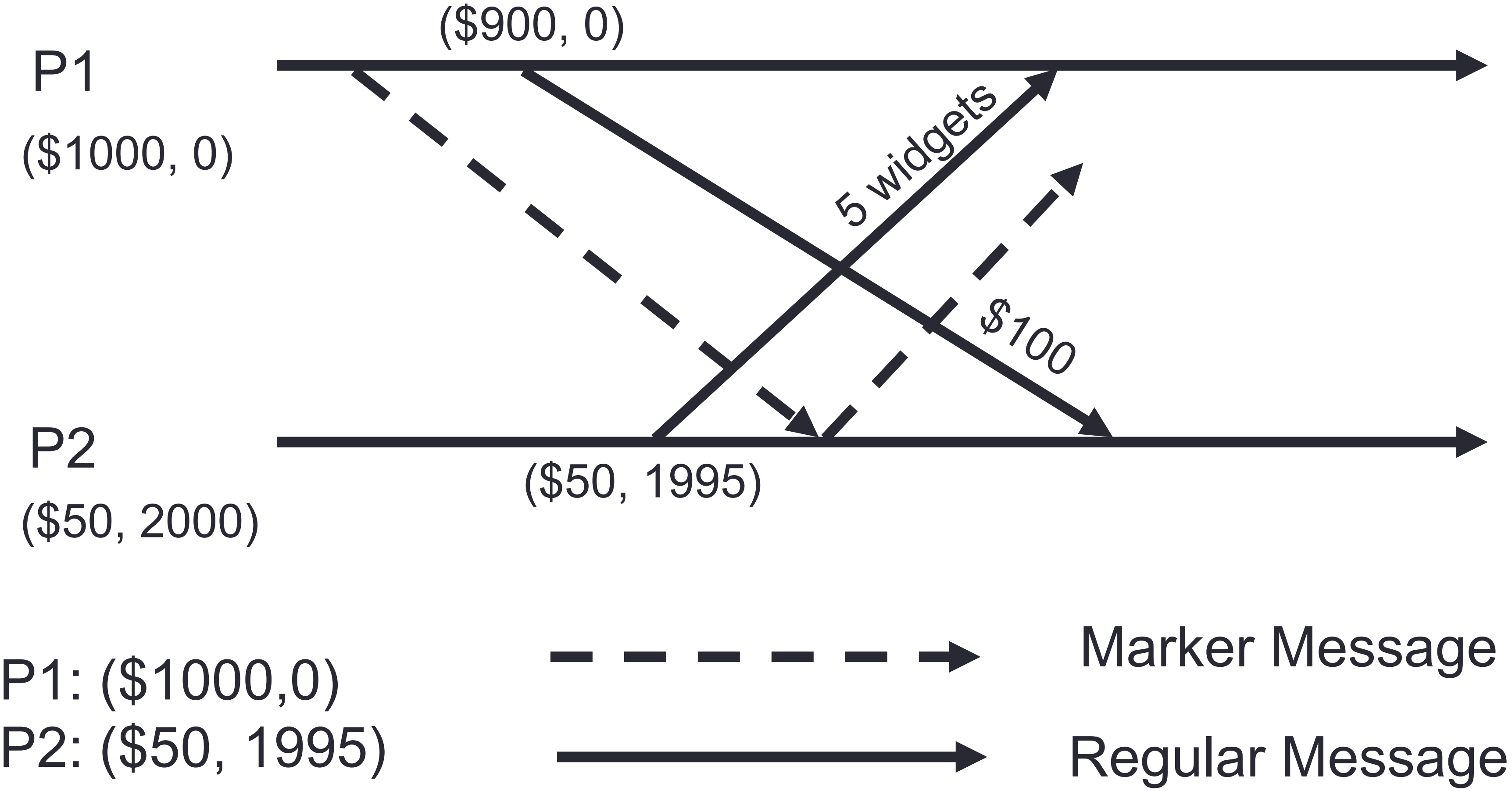
P1: ($1000,0)


P1
($1000, 0)
($900, 0)
$100
P2
($50, 2000)
($50, 1995)
5 widgets
Marker Message
Regular Message

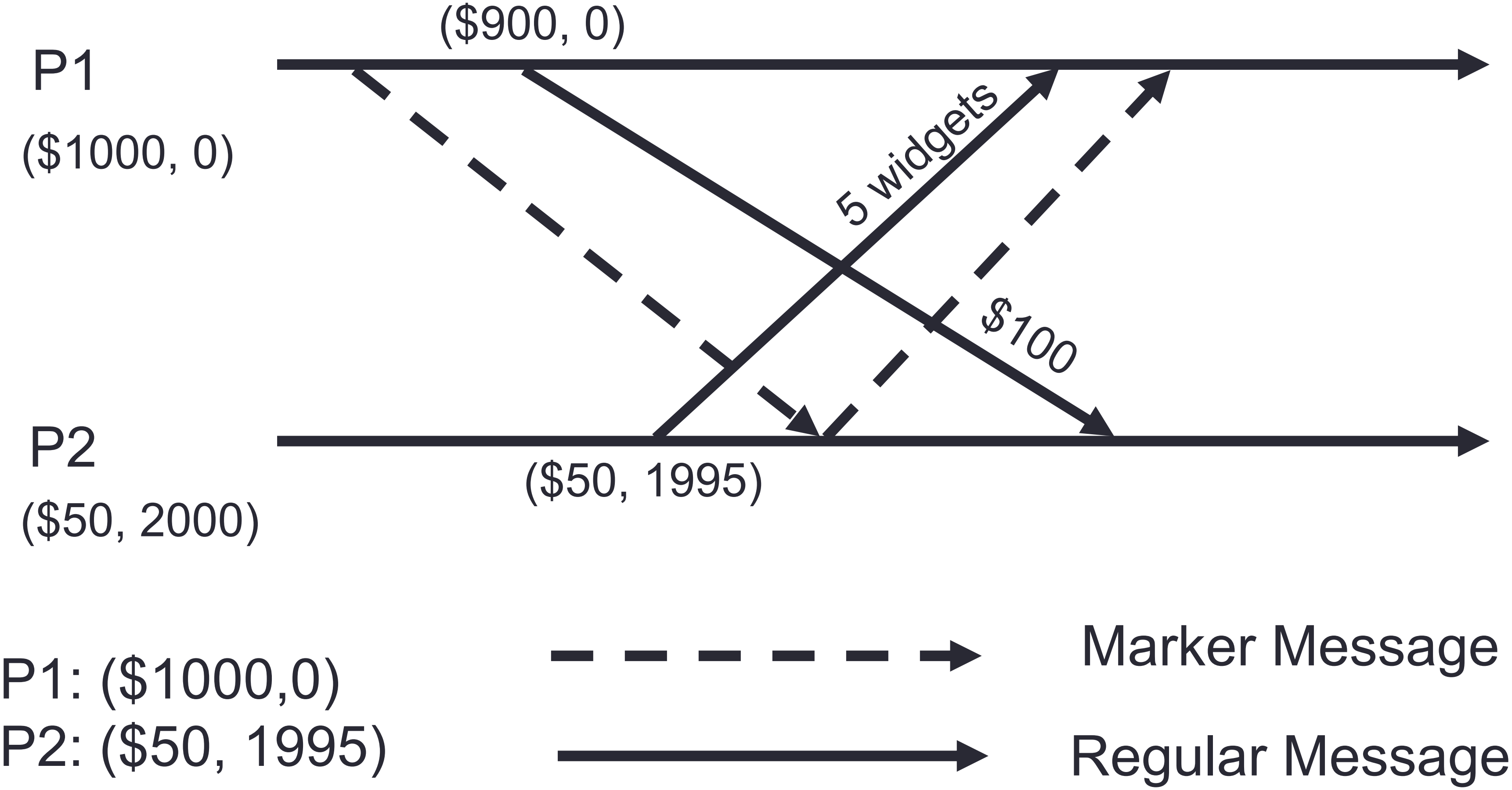
P1: ($1000,0)



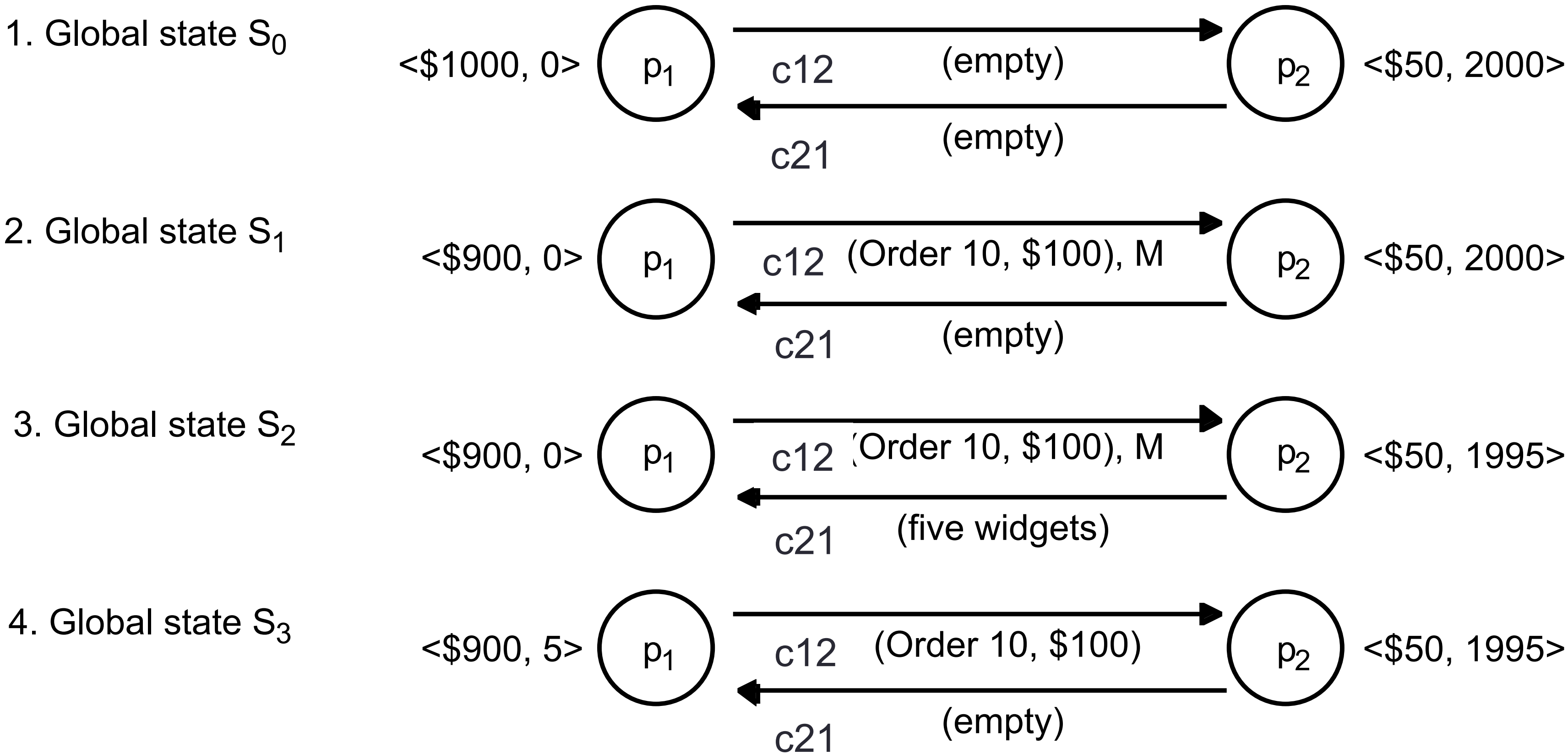
P1: ($1000,0)
P2: ($50, 1995)
C12: empty



P1: ($1000,0)
P2: ($50, 1995)
C12: empty



P1: ($1000,0)
P2: ($50, 1995)
C12: empty
C21: (five widgets)



(M = marker message)

Final recorded state: P1: ($1000,0), P2: ($50, 1995), C12: empty, C21: (five widgets)

Reading

- Section 14.5 of CBook
- Paper on myCourses