# Feb 6
# Distributed Snapshots

Vijay Chidambaram

# Using Vector Clocks in the Network Observer

- Network Observer maintains array D, initialized to 0
- Deliver message M with time stamp V from J as soon as:
  - D[J] = V[J] - 1
  - D[K] >= V[K], for all K != J
- When Network Observer delivers M, D is updated by setting D[j] = V[j]

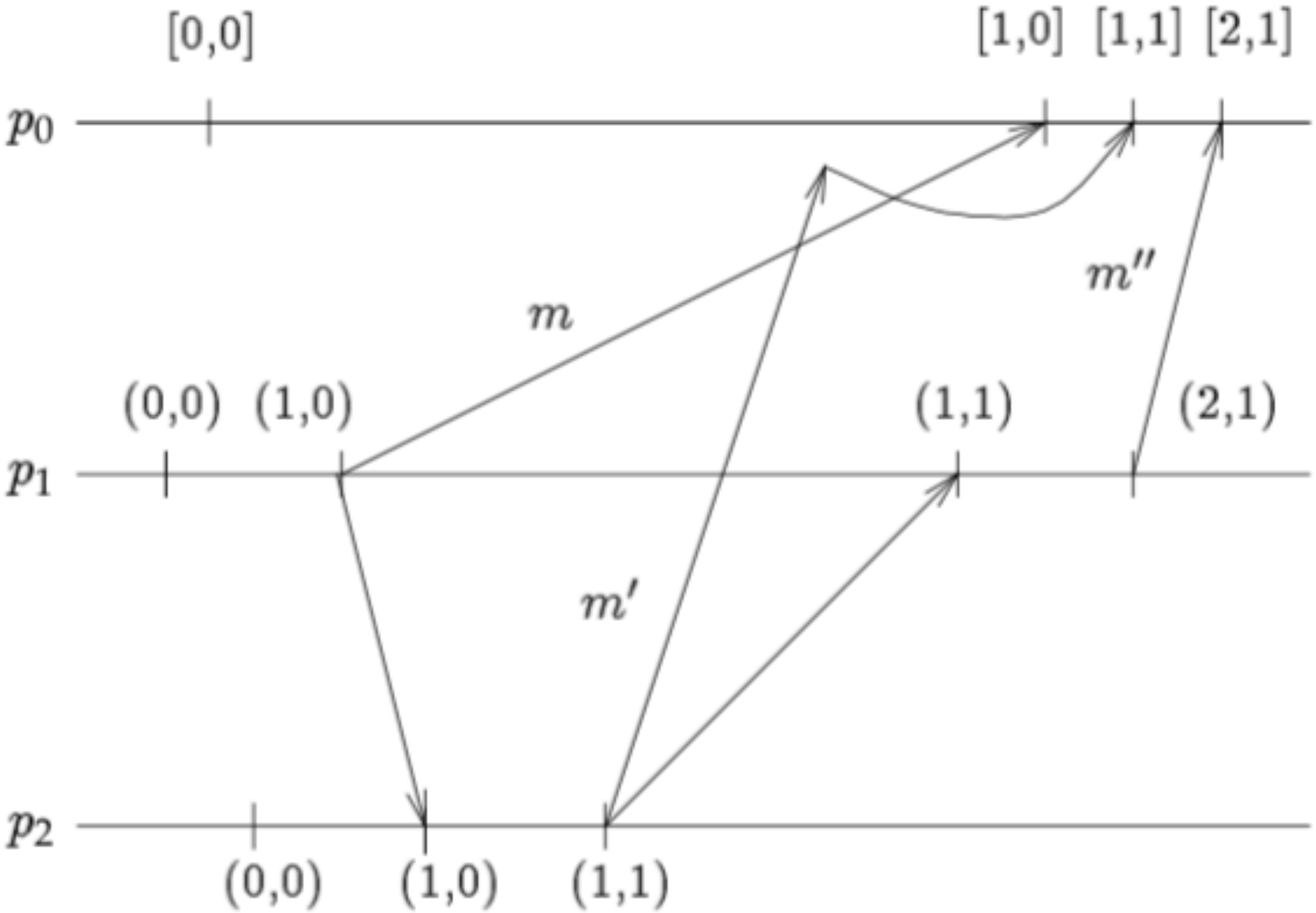


**Figure 8. Causal Delivery Using Vector Clocks**

# State Machines

- Each process in a distributed system is modeled as a state machine
- The process is in an initial state I
- Upon getting a message M from another process, the process transitions to another state S1
- In state S1, process responds to other messages by moving to other states S2..SN
- Processes transition only on receiving messages

# Distributed Snapshot Algorithm

- How to compose a global snapshot based on the snapshot of individual nodes?
- How to deal with double-counting or missing state because of messages that were in flight?

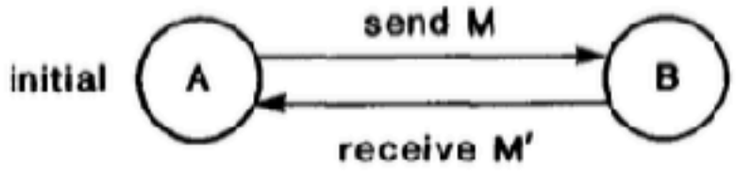


Fig. 5. State-transition diagram for process $p$ in Example 2.2.

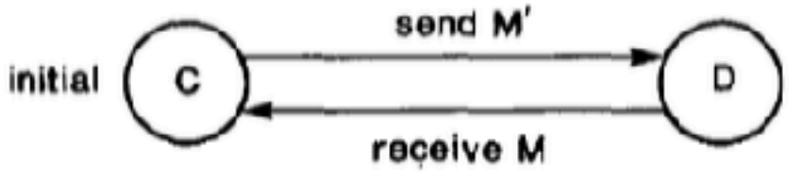


Fig. 6. State-transition diagram for process $q$ in Example 2.2.

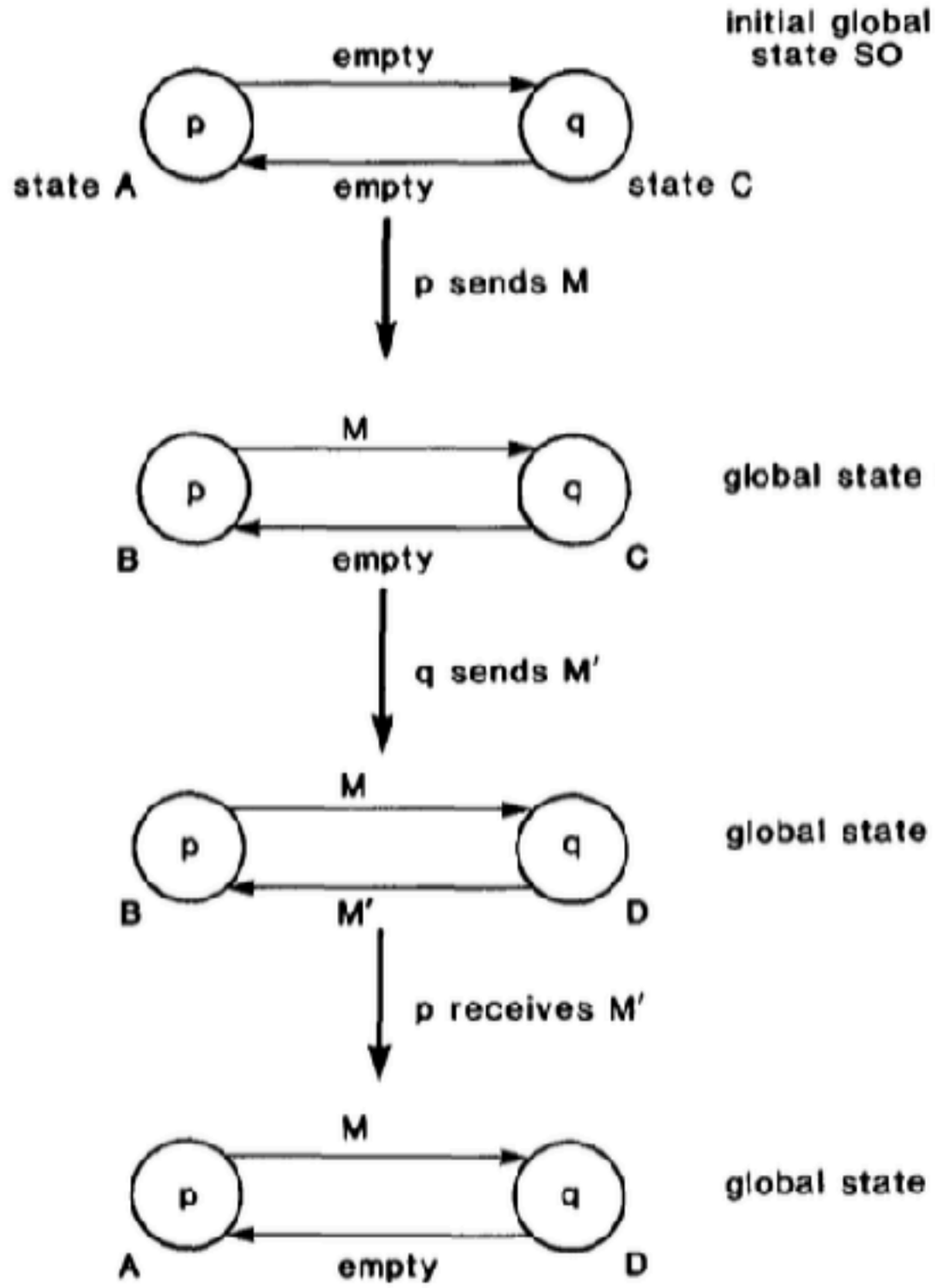


Fig. 7. A computation for Example 2.2.

# Chandy Lamport Protocol

- Assumptions:
  - No message remains forever in transit
  - Messages can be delayed but not lost
  - If the graph is not strongly connected, at least one node in each component starts the process

# Chandy Lamport Protocol

- Process p0 starts the protocol by sending itself a "take snapshot" message.

- Let pf be the process from which pi receives the "take snapshot" message for the first time. Upon receiving this message, pi records its local state i and relays the "take snapshot" message along all of its outgoing channels. No intervening events on behalf of the underlying computation are executed between these steps. Channel state (f,i) is set to empty and pi starts recording messages received over each of its other incoming channels.

- Let ps be the process from which pi receives the "take snapshot" message beyond the first time. Process pi stops recording messages along the channel from ps and declares channel state si as those messages that have been recorded.

# Validating Predicates

- Stable predicates can be faithfully validated
- Unstable predicates are tricky:
  - Algorithm may detect state that never held in an actual run of the distributed computation
  - Predicate may have changed by the time the observer gets to know

# Defining predicates

- Possibly(P): There exists a consistent observation O of the computation such that P holds in a global state of O.

- Definitely(P): For every consistent observations O of the computation, there exists a global state of O in which P holds.

# Predicates

- Possibly( P) and Definitely (not P) can hold at the same time!
- How? By being true in different global states of the same run

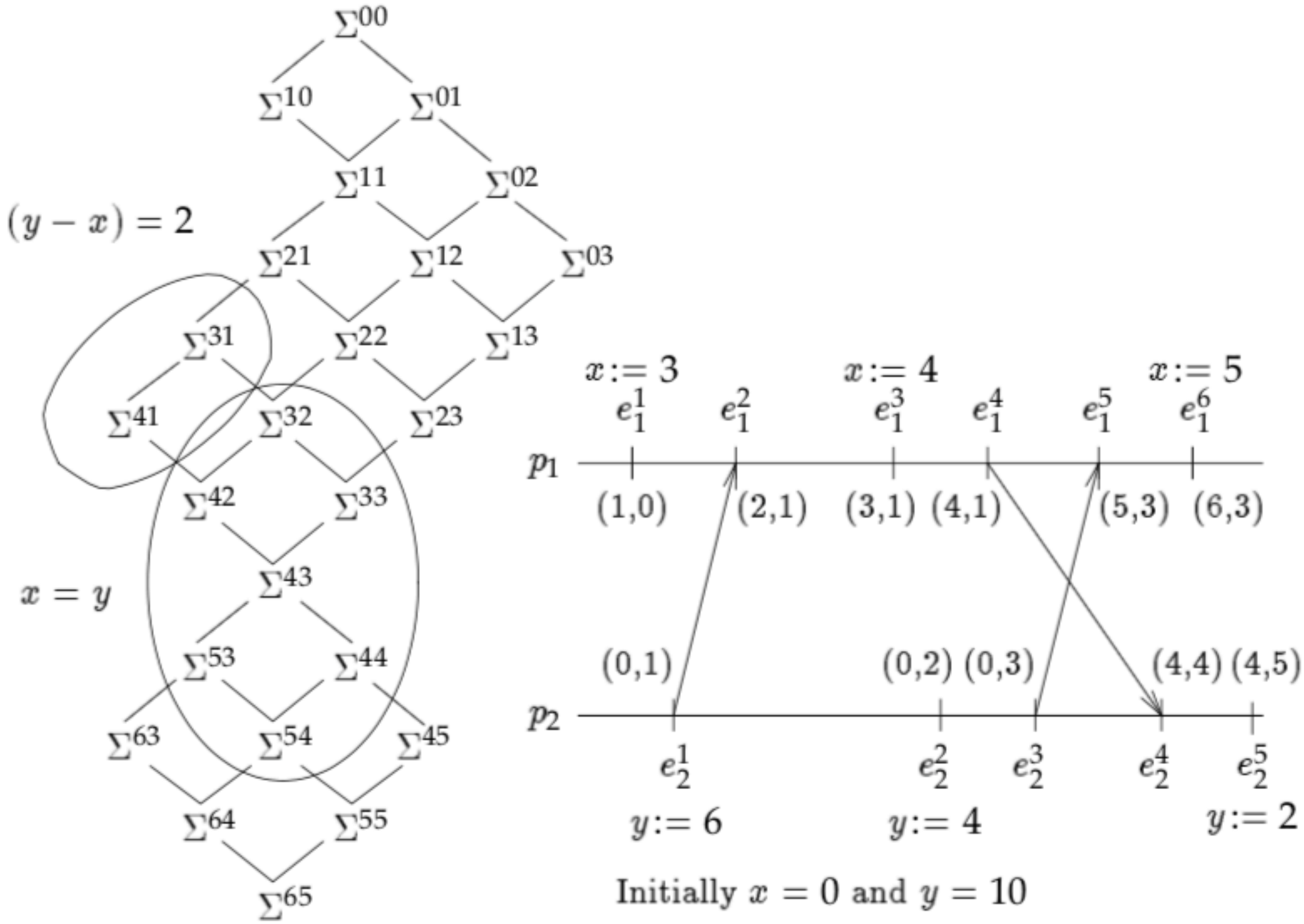


**Figure 16. Global States Satisfying Predicates $(x=y)$ and $(y-x)=2$**