Global State

- Global state of a distributed system
  - Local state of each process
  - Messages sent but not received
- Global state is needed for failure recovery
- Problem: how can you figure out the state of a distributed system?
  - Each process is independent, but send messages to other processes
  - No global clock or synchronization
- Distributed snapshot: a consistent global state

Global State

a) A consistent cut. b) An inconsistent cut
Distributed Snapshot Algorithm

• Assume each process communicates with another process using bidirectional reliable channels (e.g., TCP connections)
• Any process can initiate the algorithm
  – Checkpoint local state
  – Send marker on every outgoing channel
• On receiving a marker
  – Checkpoint state if first marker and send marker on all outgoing channels, save messages on all incoming channels until:
  – A marker is received on an incoming channel: stop saving state for that channel

Distributed Snapshot Algorithm

• A process finishes when
  – It receives a marker on each incoming channel and processes them all
  – State: local state plus state of all channels
• Any process can initiate snapshot
  – Multiple snapshots may be in progress
    • Each is separate, and each is distinguished by tagging the marker with the initiator ID and sequence number
Termination Detection

- Detecting the end of a distributed computation
- Two types of markers: Done and Continue
- Set up predecessor/successor relationships
  - Your first marker came from your predecessor
  - You are your successor’s predecessor
- Send “Done” to predecessor when
  - All your successors have sent you a “Done”
  - You are done with local computation
- Otherwise, send a “Continue”
- Computation has terminated if the initiator receives “Done” messages from everyone