Distributed Snapshots

- What is the state of a distributed system?

System model

- N processes in the system with no process failures
  - Each process has some state it keeps track of
  - There are two first-in, first-out, unidirectional channels between every process pair P and Q
    - Call them channel(P, Q) and channel(Q, P)
  - The channel has state, too: the set of messages inside
  - All messages sent on channels arrive intact, unduplicated, in order

Aside: FIFO communication channel

- “All messages sent on channels arrive intact, unduplicated, in order”
  - Q: Arrive?
  - Q: Intact?
  - Q: Unduplicated?
  - Q: In order?
  - TCP provides all of these when processes don’t fail
Global snapshot is global state

- Each distributed application has a number of processes running on a number of physical servers
- These processes communicate with each other via channels
- A global snapshot captures
  1. The local states of each process (e.g., program variables), and
  2. The state of each communication channel

Why do we need snapshots?

- Checkpointing: Restart if the application fails
- Collecting garbage: Remove objects that aren’t referenced
- Detecting deadlocks: The snapshot can examine the current application state
  - Process A grabs Lock 1, B grabs 2, A waits for 2, B waits for 1...
  - ...
- Other debugging: A little easier to work with than printf...

System model: Graphical example

- Let's represent process state as a set of colored tokens
- Suppose there are two processes, P and Q:

  Process P:  
  - Q  
  - R
  - Process Q:  
  - R  
  - Q

Correct global snapshot = Exactly one of each token

When is inconsistency possible?

- Suppose we take snapshots only from a process perspective
- Suppose snapshots happen independently at each process
- Let's look at the implications...
Problem: Disappearing tokens

- P, Q put tokens into channels, then snapshot

This snapshot misses Y, B, and O tokens

Problem: Duplicated tokens

- P snapshots, then sends Y
- Q receives Y, then snapshots

This snapshot duplicates the Y token

Idea: “Marker” messages

- What went wrong? We should have captured the state of the channels as well
- Let’s send a marker message ▲ to track this state
  - Distinct from other messages
  - Channels deliver marker and other messages FIFO

Chandy-Lamport Algorithm: Overview

- We’ll designate one node (say P) to start the snapshot
  - Without any steps in between, P:
    1. Records its local state (“snapshots”)
    2. Sends a marker on each outbound channel
- Nodes remember whether they have snapshotted
- On receiving a marker, a non-snapshotted node performs steps (1) and (2) above
Chandy-Lamport: Sending process
- P snapshots and sends marker, then sends Y
  - Send Rule: Send marker on all outgoing channels
    - Immediately after snapshot
    - Before sending any further messages

Chandy-Lamport: Receiving process (1/2)
- At the same time, Q sends orange token O
- Then, Q receives marker ▲
  - Receive Rule (if not yet snapshoted)
    - On receiving marker on channel c record c's state as empty
      channel(P,Q) = {}
Take-away points

- Distributed Global Snapshots
  - FIFO Channels: we can do that!
  - Chandy-Lamport algorithm: use marker messages to coordinate