Distributed state and dist. snapshots

*Distributed Snapshots: Determining the Global States of a Distributed System.*
Announcements

• Please set a *zoom photo* of yourself

• Recordings and slides posted for previous lecture

• Marks for piazza posts will be posted to canvas before this weekend

• My document camera arrived! :-}
Distributed state

• What is it?

• Why should we care? What is it for?

• Distributed snapshots — why bother?

• Their algorithm… umm it’s not a real state? what!?

• Stable properties: second time! Are they really that important to evaluate?
What is distributed state?

• Snapshot of the system in time: what everyone is doing at that moment

• Birds of flock metaphor

• **Distributed global state**: State at process + State of channels

• Snapshot starts … snapshot ends

  • Resulting snapshotted state is in-between (potential state that is reachable from start, and can reach end state)
Channels?

- Msgs might be in transmission
- Either you capture in process, or in channel
- So you have to capture channel
- They capture sent but not received msgs

**Why do we want to capture msgs?**

- So they can lie to us!?

  - Snapshots ~ debugging. But, not real. Debugger is faking the view of the system as best as it can. You need channels to have a “consistent” view.

- B/c messages influence state: capturing state alone is not enough. You want to capture whatever can influence state in the model that they use.
Distributed global state

• Allows us to reason about properties of system
  • Stable property detection

• Can use it for backup/recovery!
  • How would you use S* for recovery?
  • Recovery stage 1. reload the state on each process
  • Recovery stage 2. process msgs in all the channels (i.e., receive them). Make sure that you retain FIFO semantics.
Distributed global state

• Use it for debugging.
  • Photos are useful
  • Photos of dist. systems are useful

• Capture state before bug

• Capture snapshot before/after some event

• Use it for watchpoints: capture snapshot when something weird happens at one process
  • This answers the q. of *when* to take a snapshot

• Can use this to make sure that processes are behaving “correctly”. That they received msgs that were sent.
Distributed global state

- What is this thing that is being captured?
- see whiteboard
Distributed global state

- Assumptions
  - Graph of process is connected
    - Is it really distributed sys. if not connected?
    - What about disconnections, or *node churn*?
  - Symmetric channels? (all of their examples are like this)
  - Msgs are processed in finite time
  - Buffers are FIFO, links are reliable
Distributed global state

• Reliability assumption

• What if I don’t have reliable channels?

• What if I just re-try? This is fine! As long as it’s finite time (eventually the marker gets through)

• Can also use other (more reliable) links to deliver markers (this is an optimization)

• But.. alg only terminates when marker received on *every* channel (uni-directional)
Distributed global state

- FIFO assumption
- Necessary for correct snapshots
- Otherwise capture msg when should not capture
- And this is identical to capturing node in wrong state
Distributed global state

• Using TCP

• Which msgs should I record (SYN/ACK/DATA)?

• Answer 1: Only record Data (necessary)

• SYN/ACK subsidiary msgs.. only useful for securing reliability (overhead!)

• How to decide?
Distributed global state

- Msging stack: TCP | HTTP

- Do I record all HTTP msgs? How to decide?

- A: Depends… on how to define state of process

- Q. is about msgs, but is about state… ?!

- msgs ~ states

- **Answer:** define msg **consistently with definition of state**

- Important for determining level of abstraction in your system
Final/closing thoughts

• Can I have a process that records another process’ state?

• How widely observable is a process’ state?

• Is it visible to another process?

• In token example: token lives at only one process, or is in channel
  • When p doesn't have token
  • Then… token is in channel or in q
  • I can reason about another process based on MY local state
  • Because my state is related to the other process state (through msgs)
  • I can record it if (1) processes are correct, (2) I have a way of relating process states (inference to learn the state of other processes based on some info)

• Distributed knowledge concept
Final/closing thoughts

• Take to extreme: why bother with snapshot if you can reason about state relationships

• In restore: this would trade off computation for storage (reconstruct other process state, and this requires “running” the state machine to compute the state after Rx msgs). Inference has a cost.

• Use blockchain to store states! Nodes could “export” their state into a “ledger” that is visible to everyone else
  • Then snapshotting is easy: it’s all on the ledger
  • Cost: huge storage and bandw cost (consensus)
  • Opposite of inference
Next class

- Ousterhout. The Role of Distributed State. TR 1990.
  - The practical side of distributed state
  - Note the diff. definition of distributed state
  - Related to distributed file systems (of the late 80s)
  - Many engineering lessons!