Distributed Snapshots: Determining Global States of Distributed Systems

Phil Gibbons

15-712 F15

Lecture 8

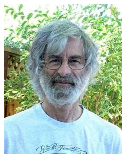
Today's Reminders

- **Summaries**
  - Everyone should have received feedback on their summaries
  - Please turn in missing summaries for 9/11-9/18 (before Monday)
  - Starting this week: enforcing 48 hour cut-off
  - Starting Monday: will be able to benefit from seeing others' summaries, once 48 hour cut-off has passed
- **Projects**
  - Should be formed by next Wed so can start brainstorming ideas
  - Week after that: we will post possible project ideas
  - OK (encouraged) to overlap project idea with current research under advisor

Distributed Snapshots: Determining Global States of Distributed Systems

Mani Chandy and Leslie Lamport [TOCS 1985]

- **Mani Chandy (Caltech, NAE)**
- **Leslie Lamport (MSR, NAE, Turing Award)**



"This paper takes the idea of consistency for distributed predicate evaluation, formalizes it, distinguishes between stable and dynamic predicates, and shows precise conditions for correct detection of stable conditions. The fundamental techniques in the paper are the secret sauce in many distributed algorithms for deadlock detection, termination detection, consistent checkpointing for fault tolerance, global predicate detection for debugging and monitoring, and distributed simulation." – SigOps HoF citation

Leslie Lamport on Today's Paper

"The distributed snapshot algorithm described in this paper came about when I visited Chandy, who was then at the University of Texas in Austin.

He posed the problem to me over dinner, but we had both had too much wine to think about it right then.

The next morning, in the shower, I came up with the solution.

When I arrived at Chandy's office, he was waiting for me with the same solution.

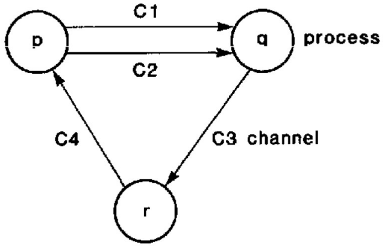
I consider the algorithm to be a straightforward application of the basic ideas from [27]."

[27] is "Time, Clocks, and the Ordering of Events in a Distributed System"

Global State Detection

System Model

- Finite labeled, directed graph in which vertices represent processes & edges represent channels



- Channels have infinite buffers, in-order delivery, arbitrary but finite delays, are uni-directional & error-free

Events

- process p
- state s of p immediately before the event
- state s' of p immediately after the event
- channel c (if any) whose state is altered by the event
- message M (if any) sent/received along c

Example

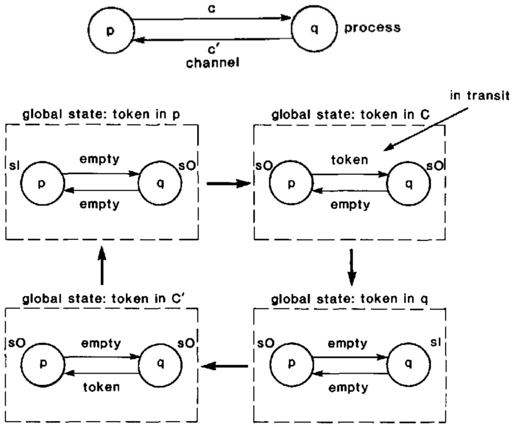
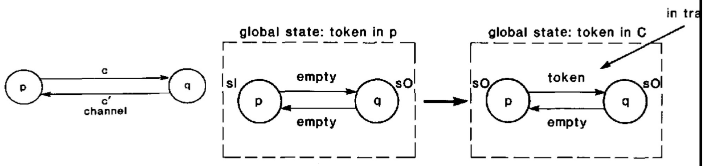


Fig. 4. Global states and transitions of the single-token conservation system.

Inconsistent Global State



- state of p: in-p (p has token)
- state of c: in-p (empty)

state transitions to in-C

- state of q: in-C
  state of c: has token
  state of c': empty
- state of p: in-C
  state of q: in-C
  state of c': empty

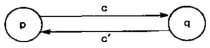
Problem: global state shows

- **2 tokens in system**
- **0 tokens in system**

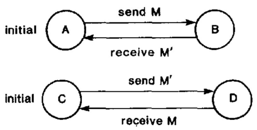
Global-State-Detection Algorithm

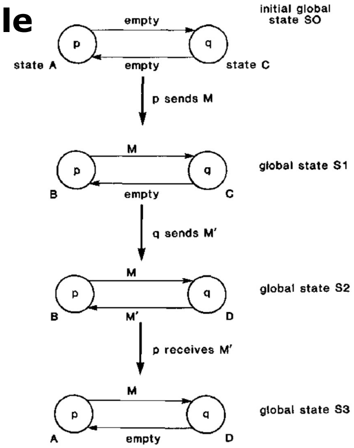
- **Marker-Sending Rule for p**
  For each channel c outgoing from p:
  - p records state, then sends a marker as its next message on c
- **Marker-Receiving Rule for q**
  On receiving a marker along a channel c:
  - If q has not recorded its state then
    q records its state; q records the state c as empty
  - Else q records state of c as the sequence of messages received along c after q's state was recorded yet before q received the marker along c

Termination: As long as at least 1 process spontaneously records it state & no marker remains stuck in a channel
& the graph is strongly connected,
then all processes record their states in finite time

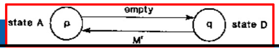


Example





1. In S0, p records state=A,
   puts marker on c
2. p puts M on c (S1)
   q puts M' on c' (S2)
   p receives M' (S3)
3. Marker received by q;
   q records state=D, c=empty,
   q puts marker on c'
4. Marker received by p;
   p records c'=<M'>



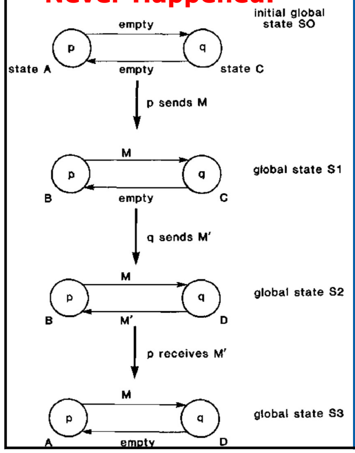
Never Happened!

So…In What Way is the Recorded Global State "Meaningful" ?

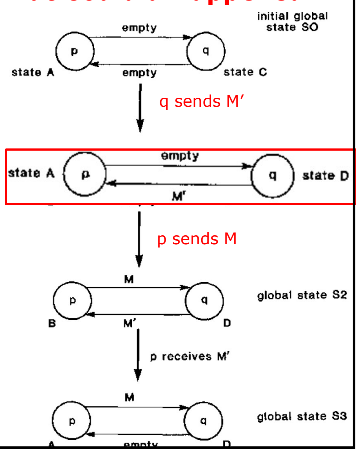
- **It *could* have occurred**
- **There is a computation where**
  - Sequence of states before the DS algorithm starts is unchanged
  - Sequence of states after the DS algorithm ends is unchanged
  - Sequence of events in between may (only) be reordered
  - Recorded global state is one of the states in between
- **But why is that useful???**

Applying to Prior Example

Never Happened!



But Could'a Happened



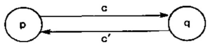
Theorem & Proof Sketch

- **There is a computation seq' derived from seq where**
  - Sequence of states before/after DS starts/ends is unchanged
  - Sequence of events in between may (only) be reordered
  - Recorded global state S* is one of the states in between

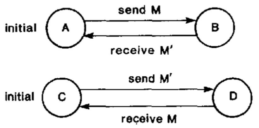
- **Prerecording event: occurs at p before p records its state**
  Postrecording event: …after…

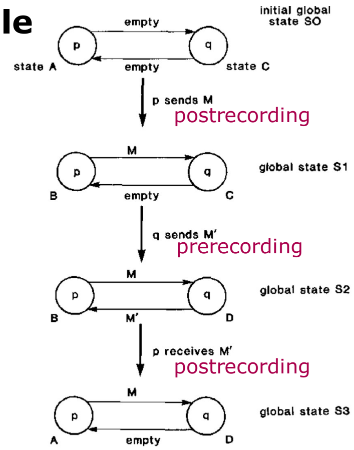
- **seq' is seq permuted such that all prerecording events occur before any postrecording events**

- **Must show:**
  - seq' is a legal computation
  - S* is the global state in seq' at the transition point



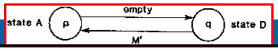
Example



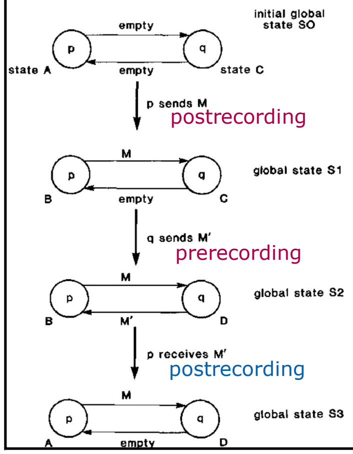


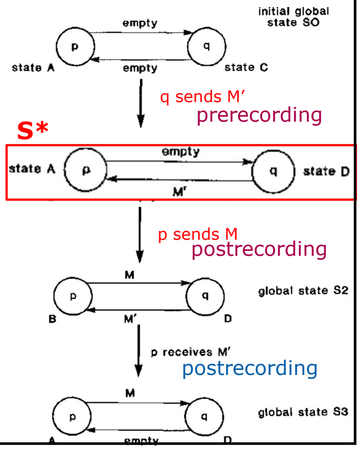
1. In S0, p records state=A, puts marker on c
2. p puts M on c (S1)
   q puts M' on c' (S2)
   p receives M' (S3)
3. Marker received by q;
   q records state=D, c=empty,
   q puts marker on c'
4. Marker received by p;
   p records c'=<M'>

S*



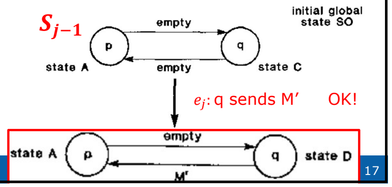
Example: Swapping Post and Pre





Swapping Post and Pre

- **Why legal to swap e_{j-1} (post) and e_j (pre) ?**
  - On different processes, say p and q
  - No message M' sent at $\boldsymbol{e_{j-1}}$ received at $\boldsymbol{e_j}$
    Why? Since $\boldsymbol{e_{j-1}}$ is post, marker already sent
    If M' received then q already received marker
    & recorded state, so e_j would be post
  - State of q not altered by occurrence of $\boldsymbol{e_{j-1}}$ since at p
  - If $\boldsymbol{e_j}$ is a receive M along c event, then M already at head of c before $\boldsymbol{e_{j-1}}$
  - Thus, $\boldsymbol{e_j}$ can occur in global state S_{j-1}



Swapping Post and Pre

- **Why legal to swap e_{j-1} (post) and e_j (pre) ?**
  - State of p not altered by occurrence of $\boldsymbol{e_j}$
  - Thus, $\boldsymbol{e_{j-1}}$ can occur after $\boldsymbol{e_j}$
  - Moreover, state after $e_1, \dots, e_{j-2}, e_j, e_{j-1}$ is same as $e_1, \dots e_{j-1}, e_j$
- **Repeatedly swap until all pre before any post**
- **S* is the same as state at pre-to-post transition**
  - Follows from Marker-Send and Marker-Receive rules

QED

Stability Detection

- **Input: Any stable property y**
  Stable: y(S) implies y(S') for all global states S' reachable from S
- **Return:**
  - FALSE only if property y did not hold when DS algorithm starts
  - TRUE only if property y holds when DS algorithm ends

  Note: If y starts holding after DS start, ok to return false
- **SD Algorithm:**
  - Record a global state S*; Return y(S*)
- **Correctness:**
  - y(S*)=true implies y(DS end state)=true [reachable, y stable]
  - y(DS start state)=true implies y(S*)=true [reachable, y stable]

Collecting the Global State

- **Each p repeatedly floods along all outgoing channel what it knows about the global state**

Monday’s Paper

Scale and Performance in a Distributed File System

[AFS from CMU]

**John Howard, Michael Kazar, Sherri Menees,
David Nichols, M. Satyanarayanan,
Robert Sidebotham, Michael West [TOCS 1988]**

SigOps Hall of Fame paper