Lazy Snapshots

Nigamanth Sridhar and Paul A.G. Sivilotti
Computer and Information Science
The Ohio State University
{nsridhar,paolo}@cis.ohio-state.edu

Outline

- **Global state**
- Inequality characterization of marker-based approach
- Lazy snapshot algorithm
  - Some specializations
- Conclusion

Distributed Systems

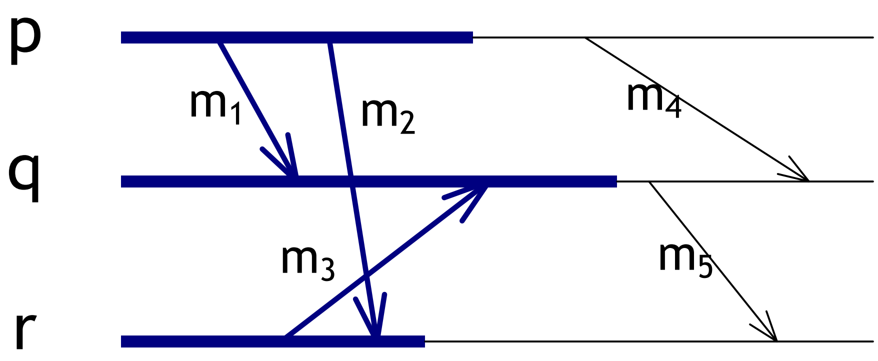
- Finite set of processes and a finite set of FIFO channels
- No globally shared memory or clock
- Process communication is via message passing
- Described by a directed graph
  - The nodes represent processes; edges represent channels

Global State

- Union of the local states of the processes, as well as the states of the channels
- Since there is no sharing of memory between the processes, the global state has to be detected by all the processes cooperating in some way
- A **global snapshot** is the state of the entire system at a particular point in time
  - state of each process
  - state of each channel (messages *in transit*)

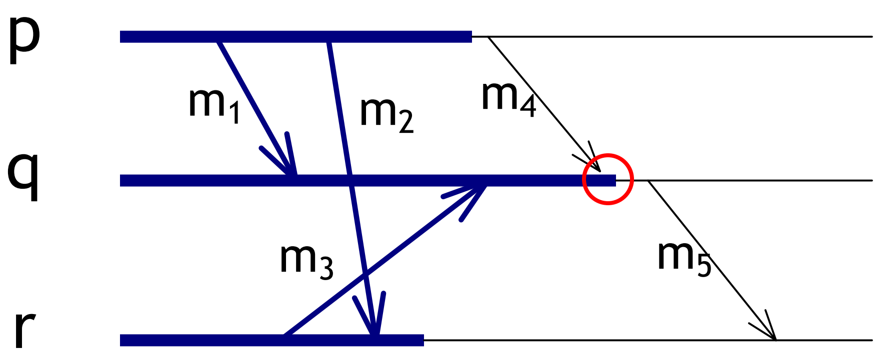
Consistent Cut

- Meaningful global state
- Every message recorded as *received* has also been recorded as *sent*
  - No *orphan* messages



Inconsistent Cut

- Global state is meaningless
- System could never be in such a state
- Channels may include orphan messages



Marker Approach to Snapshots

- Marker messages are used to distinguish events before and after the local snapshot in each process
  - Marker messages signal when a process should take its local snapshot
- Union of all these local snapshots yields global snapshot
- Marker messages must be sent so that resultant cut is consistent
  - Ordering of marker messages should rule out orphan messages

Marker Algorithm Desiderata

- Safety: The state gathered is consistent
  - Every message recorded as *received* must be recorded as *sent*
  - Every message recorded as *in transit* must be recorded as *sent*
- Progress
  - The algorithm must terminate to yield a global snapshot

Outline

- Global state
- **Inequality characterization of marker-based approach**
- Lazy snapshot algorithm
  - Some specializations
- Conclusion

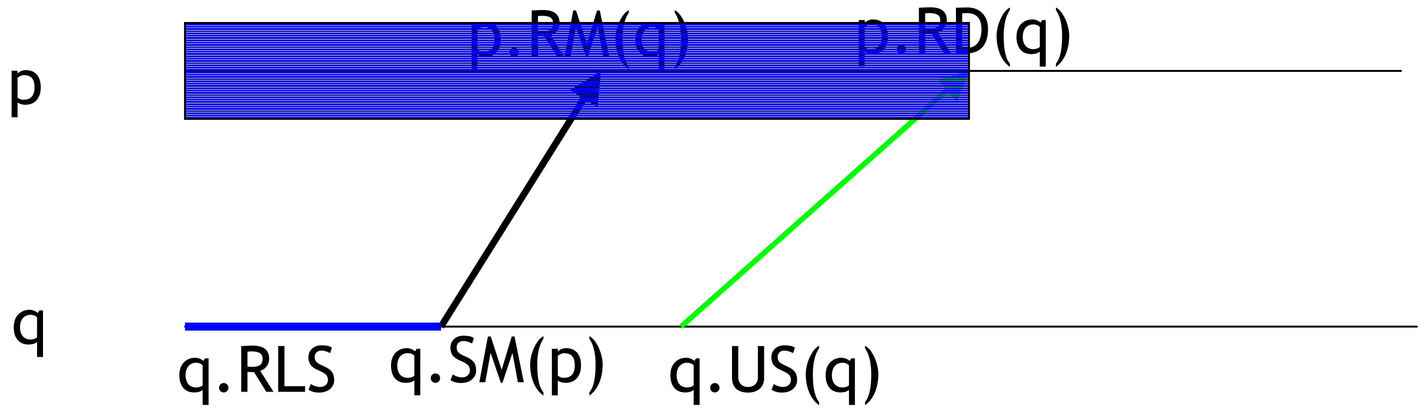
Some Terms

- **p.RLS**: process *p* records its local state
- **p.SM(q)**: process *p* sends marker to *q*
- **p.RM(q)**: process *p* receives marker from *q*
- **p.RD(q)**: process *p* receives a message from *q* after receipt of marker from *q* (on a *dirty* channel)
- **p.US(q)**: process *p* sends a message to *q* after its local snapshot (unrecorded send)
- **p.LMR(q)**: last message sent by process *p* to *q* before its local snapshot (last recorded send)

Characterization of Marker Algorithm

L1. $(\forall p :: \mathbf{p.RLS} \leq (\mathrm{Min}\ q :: \mathbf{p.RD(q)}))$

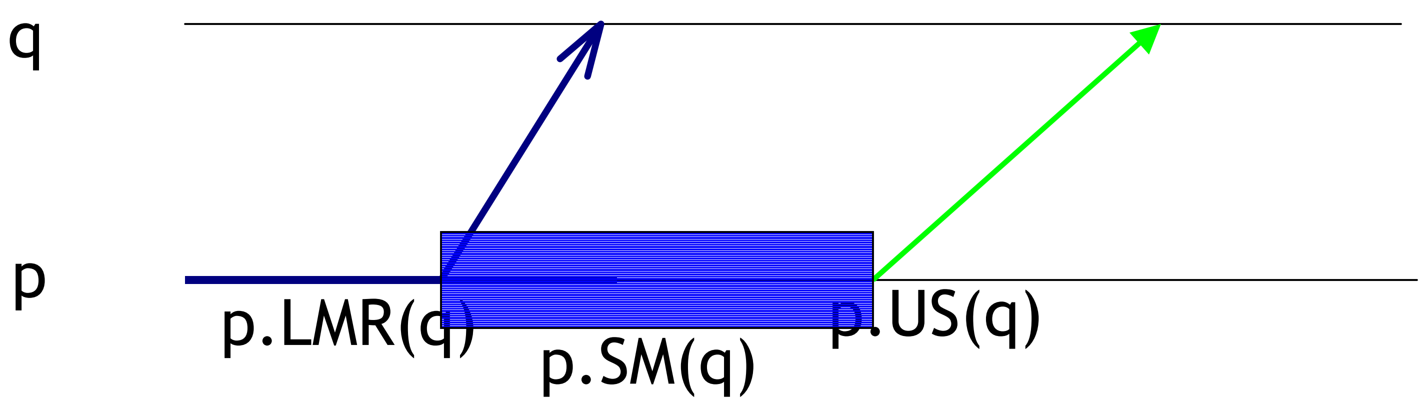
- Process *p* must record its local state before the first message along a dirty channel is received



Characterization of Marker Algorithm (contd.)

L2. $(\forall\, p, q :: \textbf{p.LMR(q)} < \textbf{p.SM(q)} < \textbf{p.US(q)})$

- Process *p* must send a marker along each of its outgoing channels before sending any unrecorded messages along that channel but not before the last *recorded* message



Outline

- Global state
- Inequality characterization of marker-based approach
- **Lazy snapshot algorithm**
  - Some specializations
- Conclusion

Lazy Snapshots: A Marker Algorithm

Marker Sending Rule for process *p*.

For each outgoing channel C, p sends one marker along C, in accordance with L2

Marker Receiving Rule for process *q*.

On receiving a marker along channel C, mark C as *dirty*;

If *q* has not recorded its local state

q records state of C as empty

Else *q* records the state of C as the sequence of messages received along C upto this point after q recorded its local state

State Recording Rule for process *p*.

Process *p* records its state before receiving any messages along a *dirty* channel (L1)

Specializing Lazy Snapshots

- The inequalities L1 and L2 characterize a **class of algorithms** that gather global state in a distributed system
- Depending on the application, the level of "laziness" can be varied
  - Processes have flexibility in scheduling their local snapshot

Chandy-Lamport Algorithm

- Local state recording is tightly coupled to marker receiving
  - Process records local state immediately upon receiving first marker
  - Markers are sent out from a process after local snapshot
- Constrains flexibility, but easy to prove correctness

Piggybacking Algorithm

- In this scheme, marker messages are not sent separately
- Messages in the underlying computation are augmented with marker information
  - Each message carries with it information about whether it is a “before” message or “after” message
- Extreme case of laziness
  - Local snapshot is postponed as much as possible

Conclusions

- The new characterization captures an entire class of marker algorithms
- A generalized lazy snapshot algorithm
- Applications can choose the level of laziness

Questions?

Author Contact:
Nigamanth Sridhar and Paul Sivilotti
Computer and Information Science
The Ohio State University
2015 Neil Ave
Columbus OH 43210
{nsridhar,paolo}@cis.ohio-state.edu

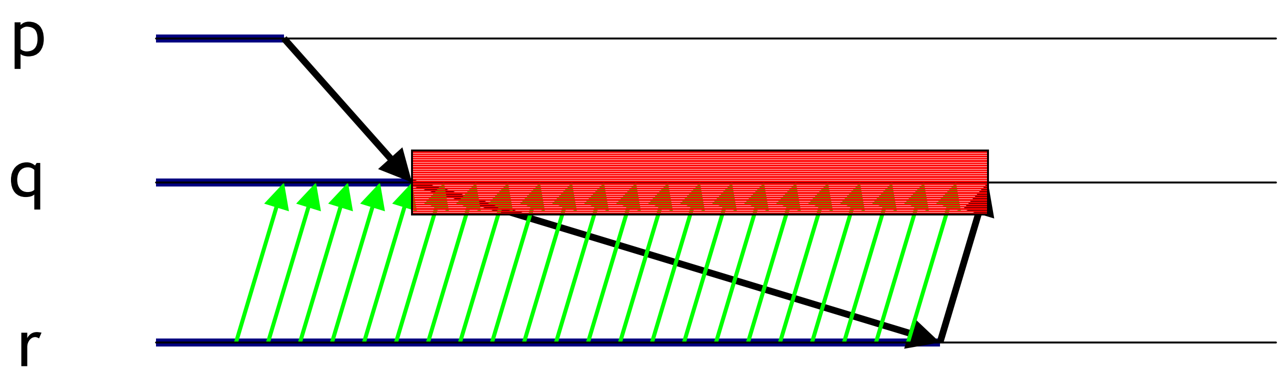
Chandy-Lamport Marker Algorithm

- Markers used to distinguish events that happened *before* and *after* the snapshot
- Algorithm outline
  - Initiator sends out markers to all its neighbors
  - Each process, on receiving its first marker,
    - takes its local snapshot
    - Sends markers on all its outgoing channels
  - Each process, on receiving each subsequent marker
    - Updates the channel state to include messages between markers

Marker Algorithm Properties

- No message received at a process *p* after the *first* marker is included in *p*'s local state
- Each subsequent marker causes *p* to update the state of the channel on which the marker was received
- In a high-traffic system, this could mean inefficiency of system execution

Global State Detection using the Chandy-Lamport Algorithm



- Process q need not have taken its local snapshot when its first marker arrived

An Optimization

- The safety spec does not mandate recording local state immediately upon receipt of the first marker
- The recording of local state can be postponed as long as no orphan messages are included in the snapshot

Lazy Snapshots

- On receiving a marker from *q*, process *p*
  - "remembers" the marker (marks the channel dirty)
  - sends markers along all outgoing channels
  - postpones the recording of its local state
- Local state recording can be postponed as long as *p* does not receive a message along a dirty channel
- If a process *p* has received markers along all its incoming channels and has still not taken its local snapshot, it is done now

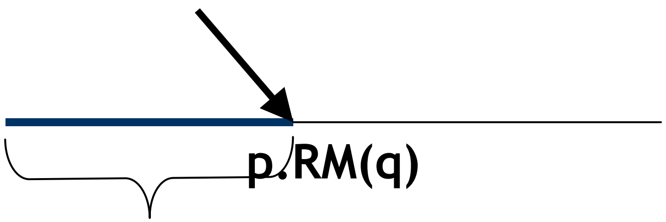
Lazy Snapshots: Advantages

- The number of “in-transit” messages in the global state is reduced
- Processes have flexibility in choosing when to schedule the recording of local state

New Characterization of Marker Algorithm

- Process *p* must record its local state before, or at the latest, at the time of receiving its first marker

E1. $(\forall p :: \mathbf{p.RLS} \leq (\mathrm{Min}\ q :: \mathbf{p.RM(q)}))$

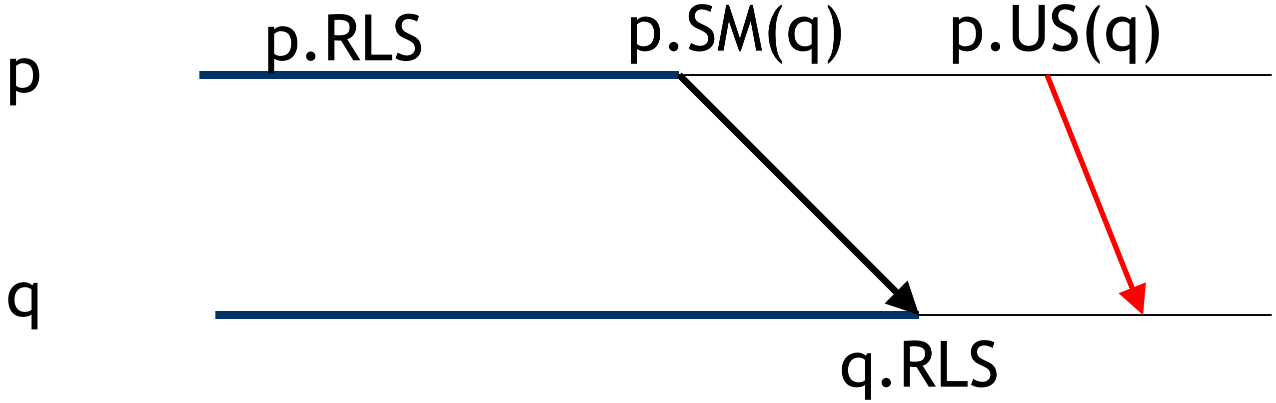


Local snapshot can occur
Anywhere here (**p.RLS**)

New Characterization of Marker Algorithm

- Process *p* must send a marker along each of its outgoing channels <u>after</u> recording its local state and <u>before</u> sending any messages along that channel

E2. ($\forall$ *p*, *q* :: **p.RLS** $<$ **p.SM(q)** $<$ **p.US(q)**)



Proof of Correctness

- Safety
  - S1. Every message recorded as received has been recorded as sent
  - S2. Every message recorded as in transit has been recorded as sent
- Progress
  - Every process takes its local snapshot