Distributed Snapshots: Determining Global States of Distributed Systems

K. Mani Chandy
University of Texas at Austin
Leslie Lamport
Stanford Research Institute

Presented by Prateek Goel
October 29, 2014

David R. Cheriton School of Computer Science
University of Waterloo
About the Authors

- **K. Mani Chandy**
  - University of Texas at Austin
  - Now CS Professor at CalTech.
  - Proposed new solution to Dining Philosophers Problem

- **Leslie Lamport**
  - Stanford Research Inst.
  - Now with Microsoft Research
  - Won Turing Award in 2013

Image Source: Google images and Wikipedia
Interesting Facts

- How the Snapshot Algorithm came to be?
  → Wine and Dine!!!

- Awards
  - Edsger W. Dijkstra Prize in Distributed Computing, 2014
  - American Academy of Arts and Sciences, 2014
  - ACM SIGOPS Hall of Fame Award, 2013
What is a Global State?

“The global state of a distributed computation is the set of local states of all individual processes involved in the computation plus the state of the communication channels.”
Why is there a need for Global State?

- Helps solve important class of problem: *Stable Property Detection*.

- Examples
  - computation has terminated
  - system deadlock
  - all tokens in a token ring have disappeared
Problems associated with determining global states in distributed systems?

- Distributed systems
  - information is spread across multiple systems

- Local Knowledge
  - a process in the computation only know its own state
Problems associated with determining global states in distributed systems?

- Synchronized recording
  - processes do not share common clocks
What is a Snapshot?

- Ex. Group of photographers observing a panoramic, dynamic scene

- Composite picture should be “Meaningful”

Image Source: http://www.upside-down.ca/cherry-oxford.jpg
Model of a Distributed System

- Processes: Finite
- Channels: Finite, infinite buffers, error-free, ordered delivery (FIFO)
Event e if defined by:

1. Process p in which event occurs
2. State s of p immediately before the event
3. State s’ of p immediately after the event
4. Channel c
5. Message M sent along c

- Defined by 5–tuple <p, s, s’, M, c>
Model of a Distributed System: Single-token conservation system
Model of a Distributed System: Non Deterministic Computation

- Initial state: A sends M to B, receives M'.
- State A: empty
- Initial global state: S0

- State C: send M' to D, receive M

- Global state S1: P sends M

- Global state S2: Q sends M'

- Global state S3: P receives M'

- Final state: Q sends M.
The global–state recording algorithm is superimposed on underlying computation without interfering with the underlying computation.
Snapshot Algorithm: Single-token system, Scenario 1 (2 tokens)
Snapshot Algorithm: Single-token system, Scenario 2 (No tokens)
Inconsistency in 2-token problem
\[ n < n' \]

Inconsistency in No token problem
\[ n > n' \]

To ensure consistent global state
\[ n = n' \]

\( n = \#\text{messages sent along } c \text{ before } p\text{'s state is recorded} \)
\( n' = \#\text{messages sent along } c \text{ before } c\text{'s state is recorded} \)
Similarly,

\[ m = m' \]

\( m = \# \text{messages received along } c \text{ before } q\text{'s state is recorded} \)

\( m' = \# \text{messages received along } c \text{ before } c\text{'s state is recorded} \)

In every state,

\[ n' \geq m' \]

Which implies

\[ n \geq m \]
Snapshot Algorithm: Marker

- Process p sends special message called \textit{“marker”} along c, after the nth message and before sending further messages.

- Marker has no effect on underlying computation.
Marker–Sending Rule for process p:

p sends one marker along c after p records its own state and before p sends further messages along c

Marker–Receiving Rule for process q:

if q has not recorded its state
    begin q records its state
        q records the state c as empty sequence
    end
else q records the state of c as the sequence of messages received along c after q’s state is recorded and before q receives marker along c
Snapshot Algorithm Overview

- **Initiator (process p)**
  - save its local state
  - send marker tokens along channel

- **Other processes (process q)**
  - on receiving first marker, save state and propagate markers along outgoing channels

- Terminate algorithm after every process saves its state
**Snapshot Algorithm: Example**

- p records global state in $S_0$, state A
- p sends marker along c
- System goes to global state $S_1$, $S_2$, and $S_3$ while marker is in transit
- Marker received by q in global state $S_3$
- q records its state, state D
- q records state c to be empty space
- After recording its state, q sends marker along c’
- On receiving marker, p records state of c’ as message $M’$
Snapshot Algorithm: Example

- Recorded global state $S^*$
- Algorithm is initiated in global state $S_0$ and terminated in global state $S_3$

Global state $S^*$ is not identical to any of the global states $S_0, S_1, S_2, S_3$
Properties of Snapshot Algorithm

- $S^*$ is reachable from initial global states
- Final global state is reachable from $S^*$
- $y(S) \rightarrow y(S')$ for all $S'$ (stable property definition)
References


Thank you!