Ordering & Consistent Cuts

6 November 2007
barry burton
CS 614
Overview

- Same Author
- Complementary view on similar but different subjects
- Rigorous representation of distributed computation
Question

- Does this matter in implementations?
- Does this matter at all?
Terms

- Distributed System
- Distributed Computation
Terms

- Global State
- Stable Property
Order

- Relation to Time
- In a Distributed System
Distributed System
Happened Before

- Denoted $\rightarrow$
- Relation over Events
- Partial Order (Irreflexive)
Happened Before

- For 2 Events a & b, a→b if
  - a comes before b in the same process
  - a sends a message received by b
Happened Before

- Otherwise $a \nrightarrow b$
- Always $a \nrightarrow a$
- If $a \nrightarrow b \& b \nrightarrow a$
- $a \& b$ are Concurrent
Logical Clocks

- Monotonically Increasing Integer Counter
- $C_i\langle a \rangle < C_j\langle b \rangle$ if $a \rightarrow b$
Logical Clocks

- $C_i$ is incremented between 2 events in $P_i$
- Upon receiving message with timestamp $T_m$, $C_j$ is set to a value $> T_m \& \geq C_j \text{ current}$
Total Ordering

• $a \Rightarrow b$ if
  
  • $C_i\langle a \rangle < C_j\langle b \rangle$
  
  • $C_i\langle a \rangle = C_j\langle b \rangle$ and $P_i < P_j$
Application

- Mutual Exclusion Problem
- Single Resource
- Requests Granted In Order
- Completes
Sideband Problem

- Logical Clocks may not always agree with intuitive notions of order
- External Events are Ignored
Physical Clocks

- $C_i(t)$ is the reading of clock $C_i$ at time $t$
- For all $i,j$ $|C_i(t) - C_j(t)| < \epsilon$
Global State

- Restrict slightly the notion of a distributed system
- Messages must be delivered in order
Channels

Fig. 1. A distributed system with processes p, q, and r and channels c1, c2, c3, and c4.
Global State

- Single Token Conservation System
- Consistency
Snapshot

- A Consistent Global State
- The only way to check for Stable Properties
Consistent States

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

• Each process records its own state
• Both processes on a channel record the state of the channel
• Ordering?
Recording Algorithm

• Ensure that messages only appear once in the global state
  • avoid recording a message at multiple processes
  • avoid recording a message at a process and in a channel
Recording Algorithm

• Initiated by some process deciding to record its state

• Terminates when all processes have recorded their states
Recording Algorithm

- Use marker
- After recording state, send marker on all incident, outgoing channels
Recording Algorithm

- On receiving marker
  - Record state of process if not yet done
  - Otherwise, record state of channel which marker came on
Recorded Global State

- May not have happened at the same time
- But could have
Recorded Global State

- The recorded Global State $S^*$ is reachable from the initial Global State and the final Global State is reachable from it.
- There exists a permutation of the actual computation which does contain $S^*$. 
Stability Detection

- Record the Global State
- Apply the given predicate to the Global State
- Return value of TRUE means property holds at termination of recording
- Return value of FALSE means that property does not hold at initiation of recording
Summary

- Elegant, provable properties of distributed systems.
- Unquestionably helpful for reasoning about distributed systems
Summary

- Practical for real implementations?
- Scalable to geographically large systems and/or high number of processes?
Summary

- Is relaxing the same ideas even helpful for designing larger scale / higher performance systems?