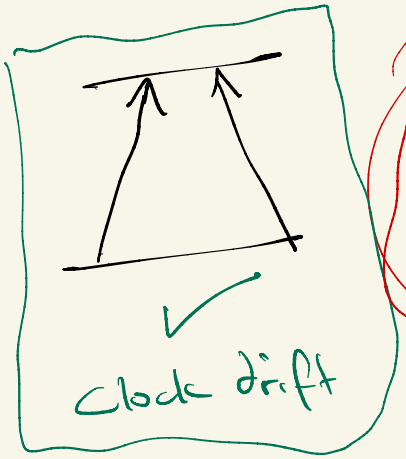
CSCI 1380 : Day 12

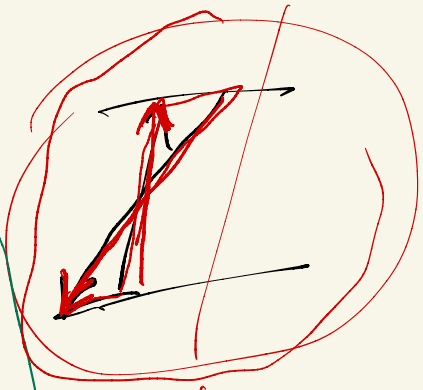
Time + Global State

Last Class

Real time is bad (not monotonic & hard to synch)
Systems want ordering that is monotonic

Ordering
- Total, FIFO → logical clocks
- Causal → VC





Today

Global state
* consistent snapshots
* Chandy Lamport

Replication (Passive/Active)
* quorum V consensus
* Raft = passive

Global state (global snapshot)

Global state (distributed snapshot) = distributed backup of the cluster

Timebased snapshot

@ some time T (2pm) all servers make backup (snapshot)

Potential problems

1. 2pm all servers stop running
2. 2pm is different on every server

manual snapshot

manually log into each server to create a snapshot

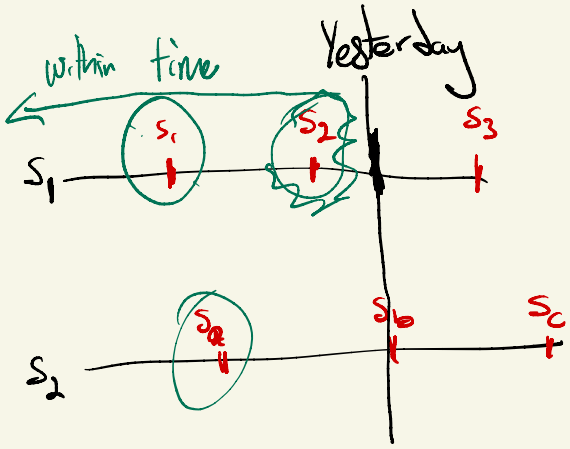
Problems

1. takes too long
2. inconsistency

Continuous snapshot

every N hours each server takes a snapshot

Dont pause during snapshot & dont try to coordinate



The trick is when you rollback (restore)

You look across servers for a consistent snapshot

1. Pick a time that you want to roll back to
2. figure all snapshots within that time
3. is latest snapshot from s_1 consistent with s_2 if yes ⇒ done
   else go back a snapshot

Problems

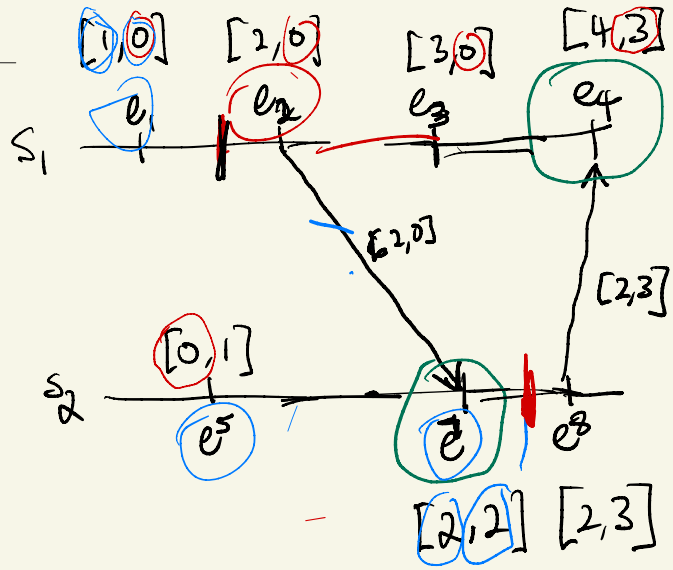
1. still have some notion of real time
2. Cascading rollback ⇝ the only consistent snapshot is @ T=0 when system starts

main problem

no guarantee to find a consistent snapshot

Consistent snapshots

only way for causality or to interstwine state on different servers is via msg



consistent = recv event also have their send event in the snapshot

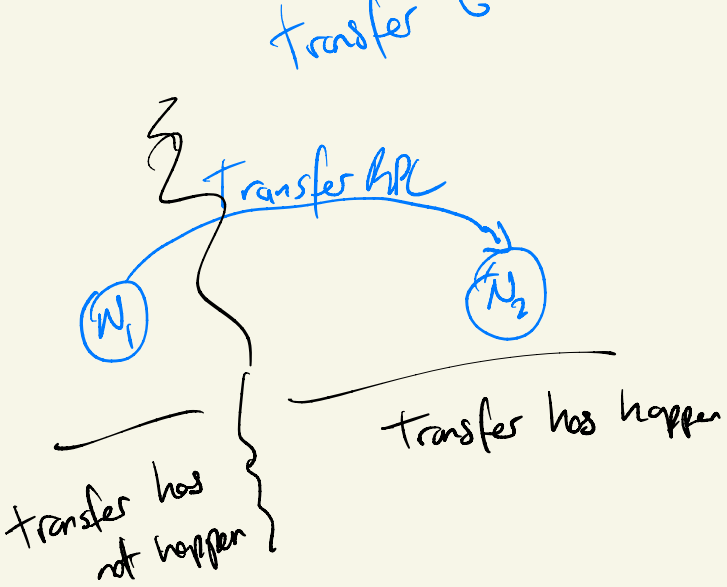
Pick random snapshot (e^5, e^2)

S_2 [0, 1] versus S_1 [2, 0]

if pick e_i from S_i

$VC_{e_i}[j] > VC_{S_j}[j]$

$j != i$



Detecting Inconsistent Snapshot

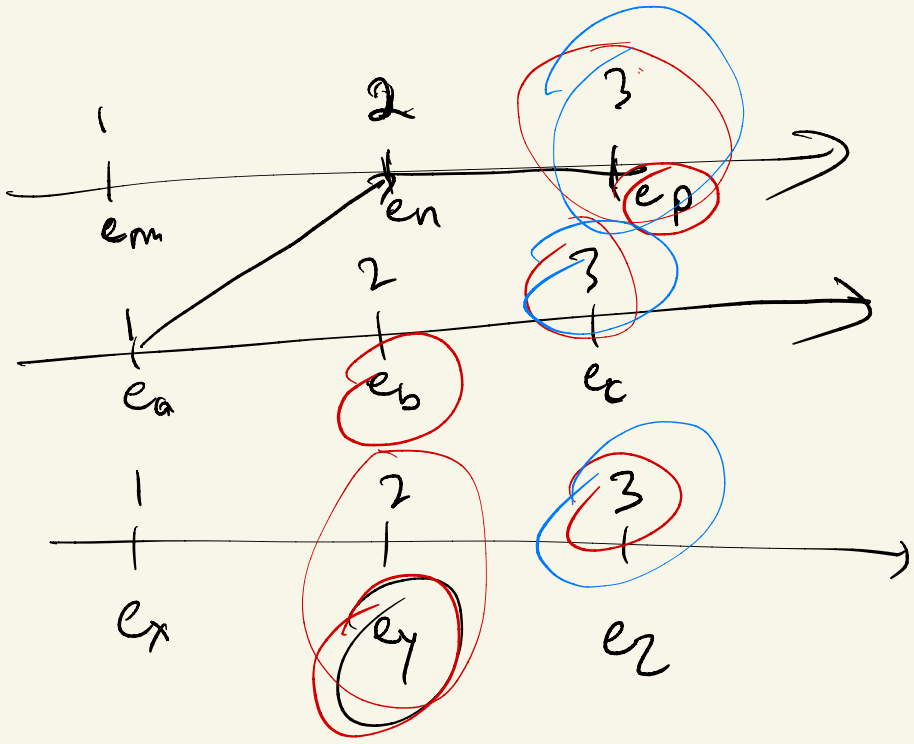
if an event at S_i has a vector clock VC_i

and an event at S_j ($S_i != S_j$) has a vector clock VC_j

then the snapshot is inconsitent if $VC_i[j] > VC_j[j]$

Thus S_i knows about more events at S_j than S_j does

hence inconsistency



logical clocks don't capture causality

& can't help infer msg btw servers

so they can't help with consistent snapshot

Chandy – Lamport

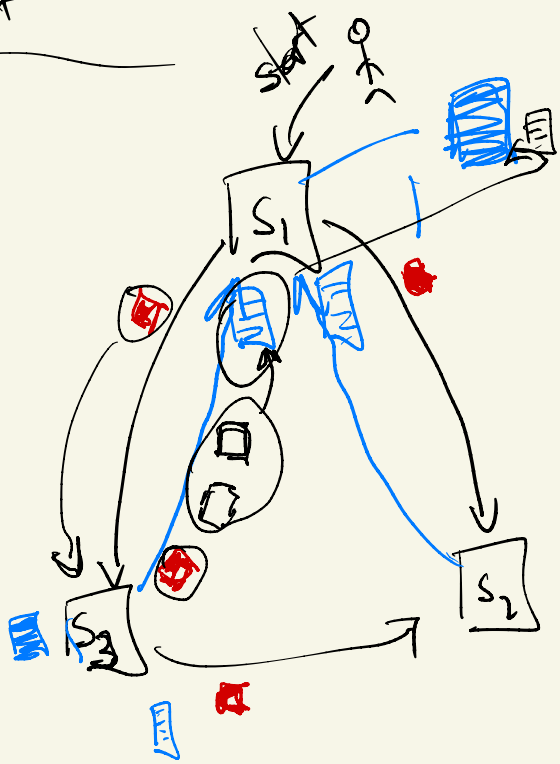
Rules

① any node can initiate a snapshot
* Sends a marker to all other nodes
* Start watching for markers for all other nodes
* Create a snapshot of state & maintain queues for each server

② when you get your first marker
* checkpoint of state
* Send a new to every one
* maintain queues for every one but the person that sent you a marker

③ each server considers the checkpoint complete after recieving markers for everyone else

marker



marker ⇒ signal that a node has done a checkpoint
msg

Assumptions

(1) Servers never crash (if they do the system stops)

(2) external entity starts the snapshot process

(3) Msg are deliver in order (FIFO order)

(4) also Msg don't get lost

↳ if marker is lost then protocol never ends

out of order introduces inconsistency

Today

Global state (distribute checkpoint/ snapshot)

* approaches for capturing global state
* determining consistent snapshots
  ⟹ using vector clock
* Chandy – Lamport algorithm for distributed snapshot