CSCI 1380 : Day 13

Global State / Application

Last Class

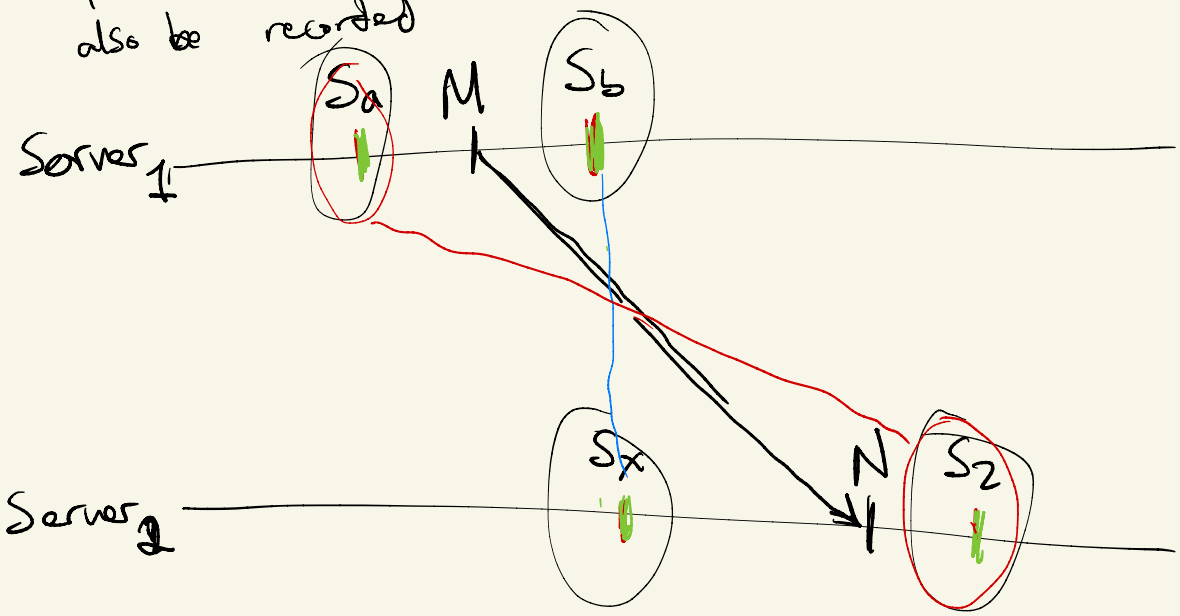
1. Various approaches for global snapshots
   a. time synchronized
   b. continuous (periodic)
   c. manual (instantaneous)
   d. Chandy Lamport algorithm
2. Using Vector Clocks to detect inconsistencies

This Class

1. Chandy Lamport analyzed
2. Passive / Active replication
   a. linearizability
   b. Tradeoffs
   c. Overheads
   d. Implications of failure

Consistent Snapshot

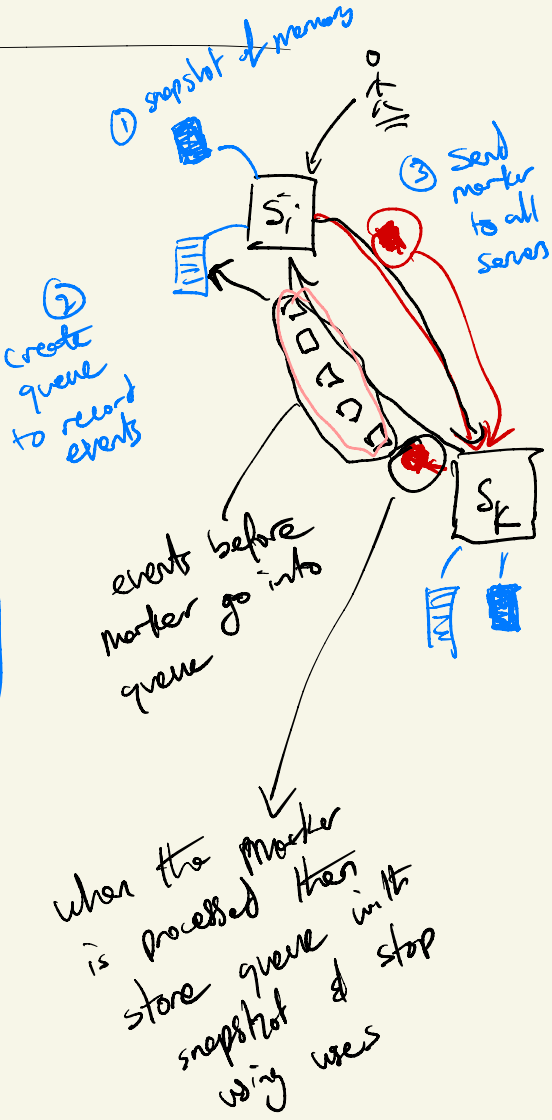
if the recv event is recorded then the send must also be recorded



Consistent = $[S_a, S_x]$, $[S_b, S_z]$, $[S_b, S_x]$

not consistent = $[S_a, S_z]$

Chandy-Lamport



① Snapshot is triggered manually
② one server who creates a snapshot & sends out markers & starts recording events

② Rules for processing a Marker from S_k
if server hasn't processed any markers
then create checkpoint of state
start recording events from other servers ($! S_k$)

else
stop recording events from S_k & place recorded events in checkpoint

③ Markers sending rules
after creating initial checkpoint
Send markers to all other servers
before sending other message

④ algorithm end after all servers have received markers from every other servers
① each server sends checkpoint to the initiator

initial checkpoint + queued events recorded

⑤ only one snapshot at a time

Chandy-Lamport

(1) Server failure

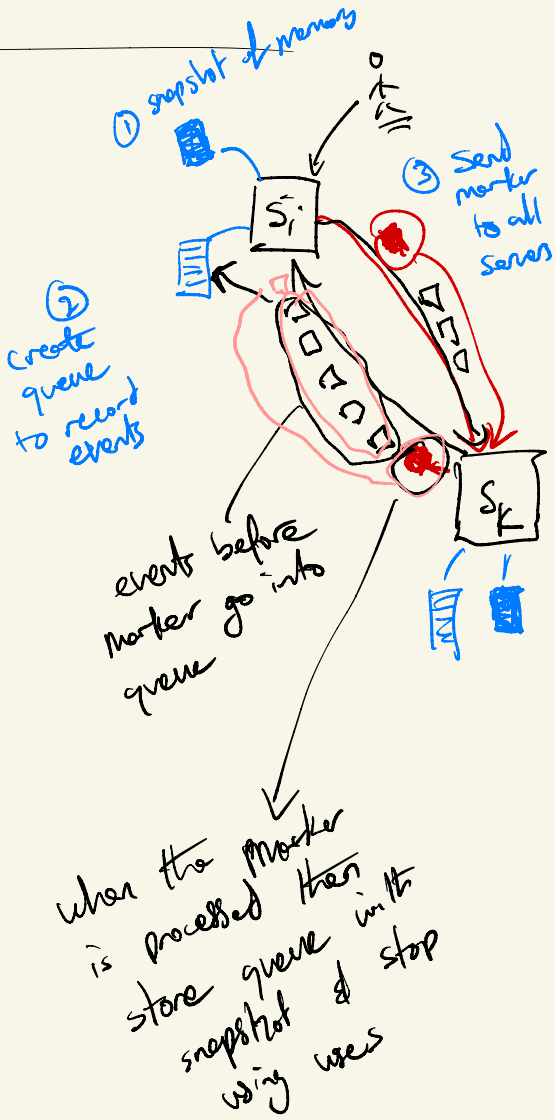
if a server fail it will not process or send out markers

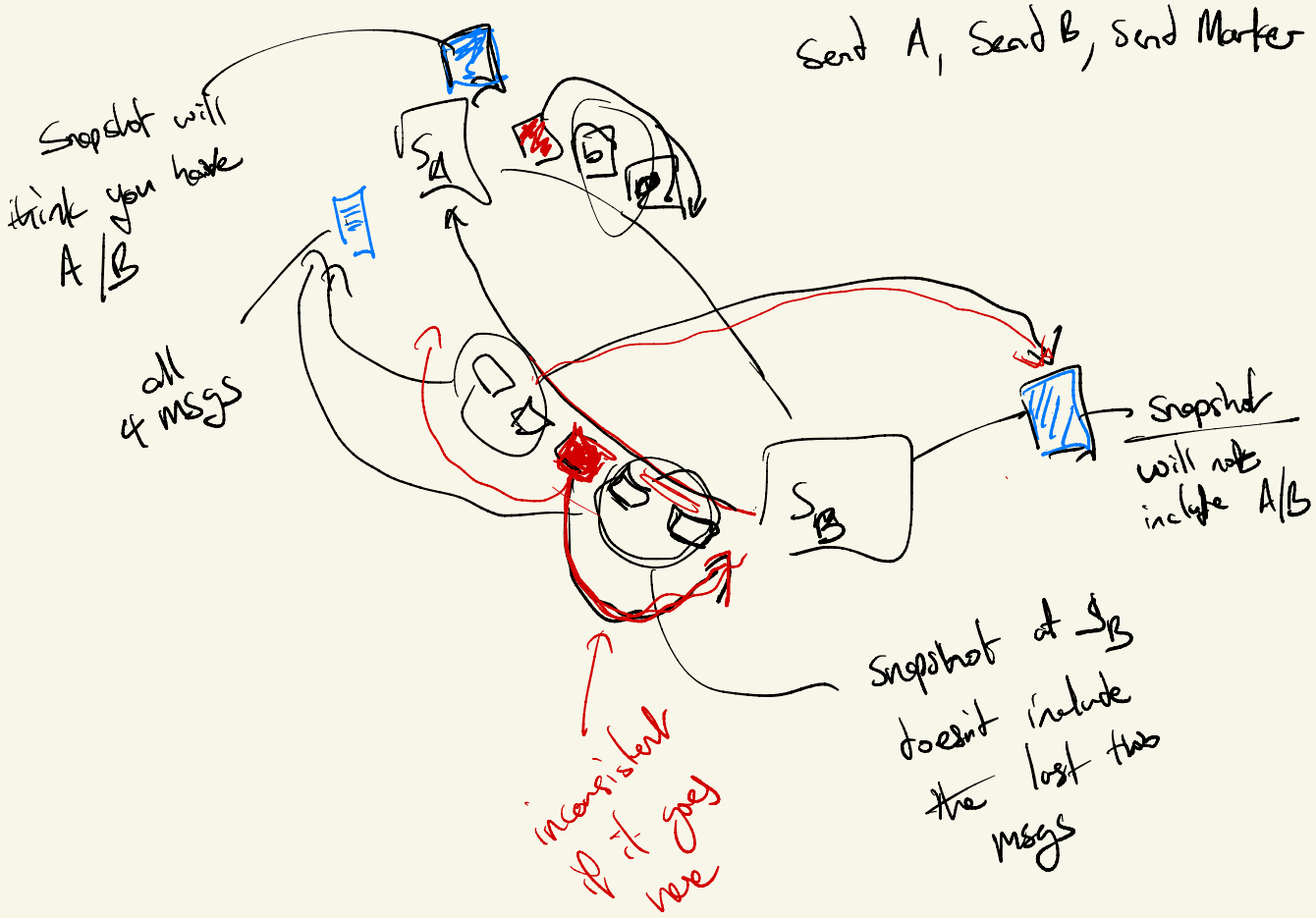
Protocol may never end if all markers are not sent

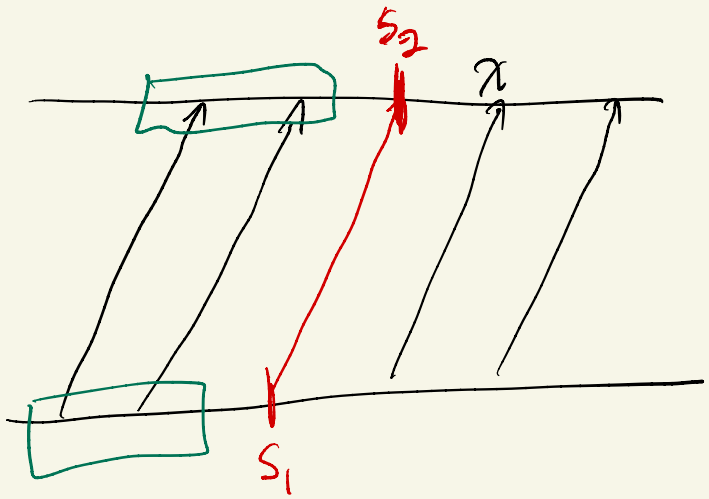
This is key because each server waits for Markers from ALL other servers

(2) Packet reordering

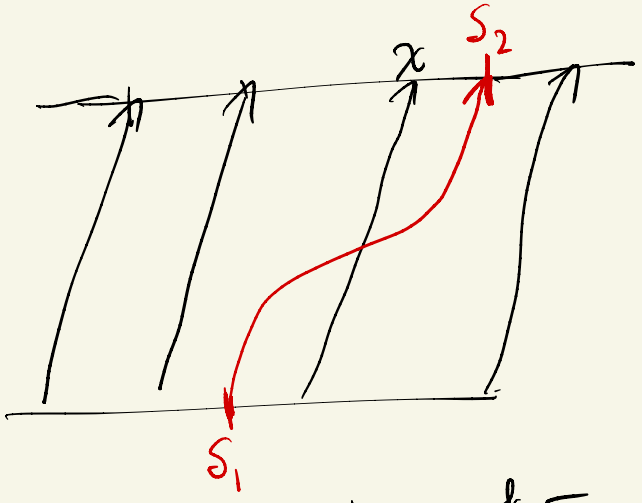
network can reorder but the protocol assumes that reordering will never happen





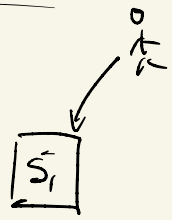


Consistent Snapshot



With reordering the marker is delivered after x & not before x hence a huge problem

Chandy-Lamport



(1) Snapshot is triggered manually
   (a) one server who creates a snapshot & sends out markers & starts recording events

(2) Rules for processing a Marker from S_k
   if server hasn't processed any markers
   then create checkpoint of state
   start recording events from other servers ($! S_k$)

   else
   stop recording events from S_k &
   place recorded events in checkpoint

(3) Marker sending rules
   after creating initial checkpoint
   send markers to all other servers
   before sending other message

(4) algorithm end after all servers have received markers from every other servers
   (1) each server sends checkpoint to the initiator

initial checkpoint + queued events recorded

Analyzing Chandy-Lamport algorithm

Termination

① each server must receive markers from all other servers

② provided no pkt loss & all servers are operating correctly then termination is guarantee

Termination Condition

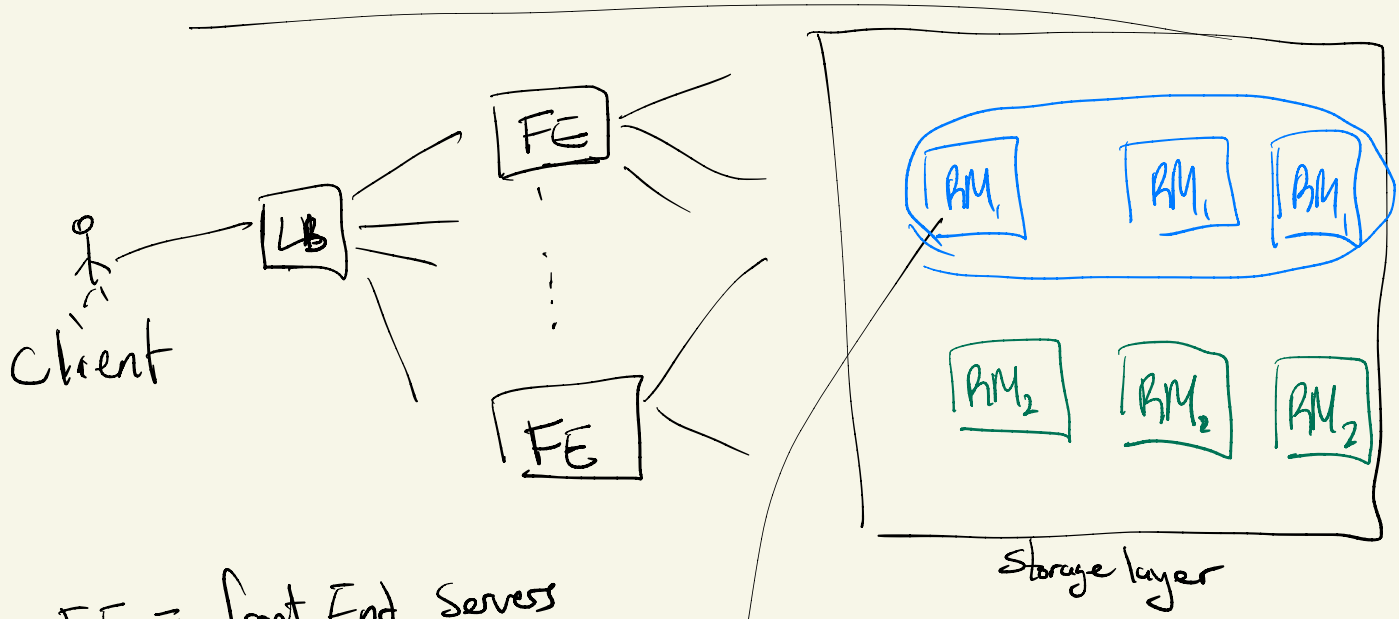
Agreement / Correctness

① each distributed snapshot has both recv/send event

② as long as there's no reorder

- send events are captured before send markers
- recv capture in snapshot or in queue before you get a marker

Replication (storage layer)



FE = front End Servers (Web/application servers)

Replica managers = store a copy of data

Blue RMs ⇒ store imgs

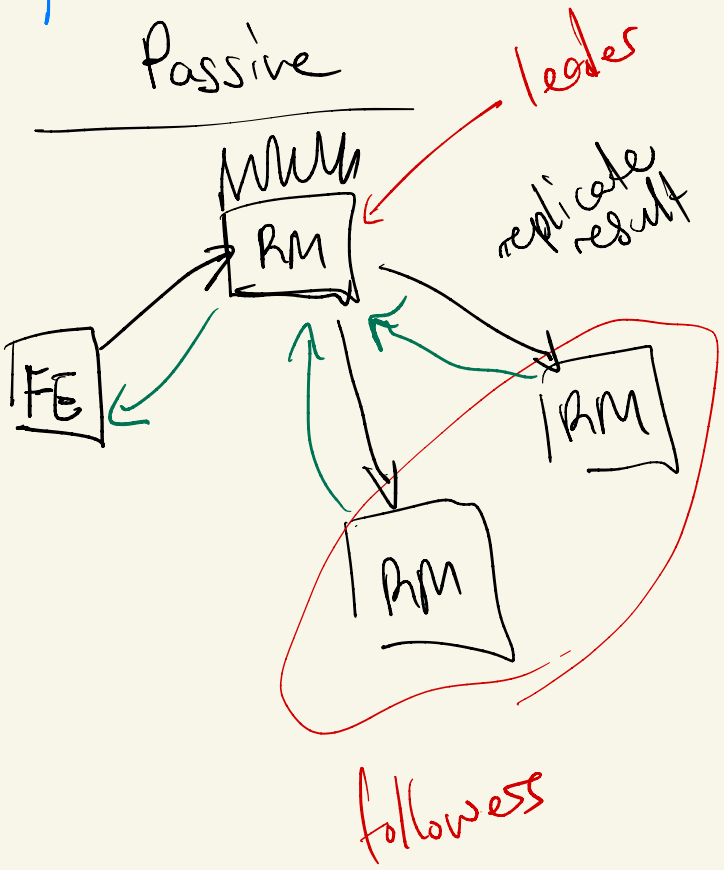
green RMs ⇒ store video

Properties of Replication Layer

(1) Transparency = client is never aware of all the copies of the data the illusion of just one copy

Raft / Zookeeper / etcd

Passive



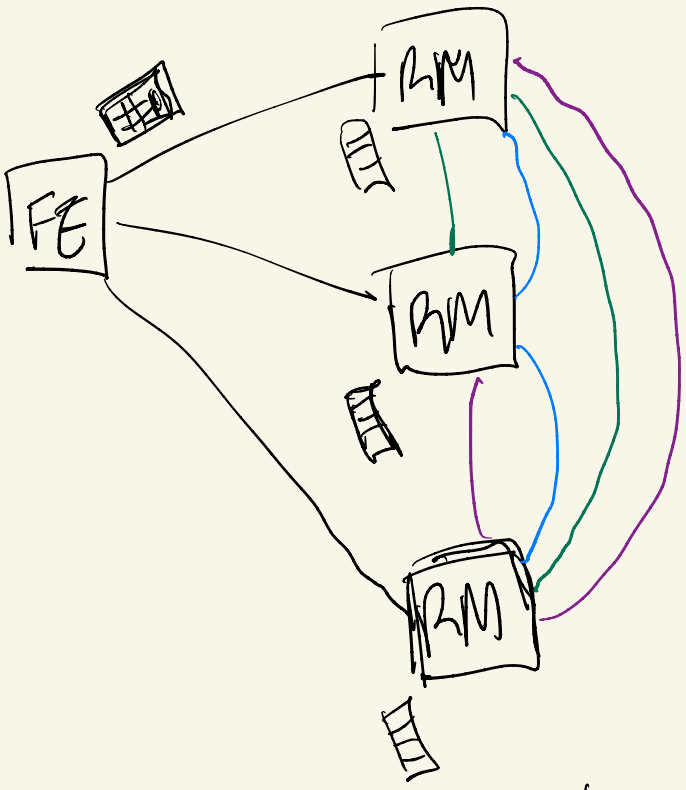
(1) FE always interacts with leader RM

(2) Leader RM applies changes locally then replicates to follower

(3) after all (or majority) of followers acknowledge then leader reports to FE

Active/Active LevelDB/RocksDB

Active



(1) FE interacts with all RMs

(2) Each RM lets the other RMs know of events received

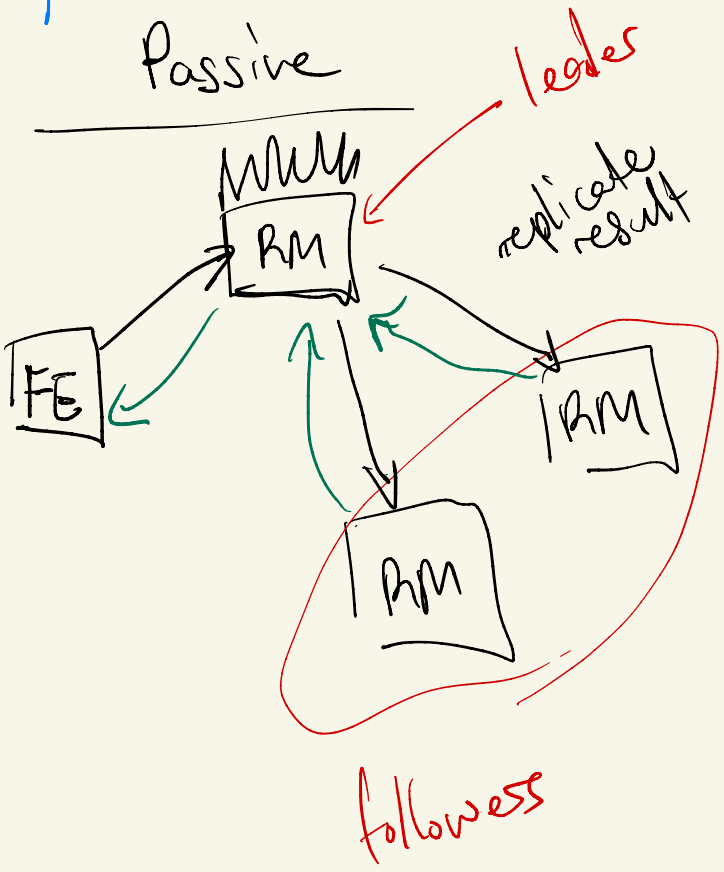
(3) Each RM maintains an order of event & processes the first event with acknowledge from all

Raft / Zookeeper / etcd

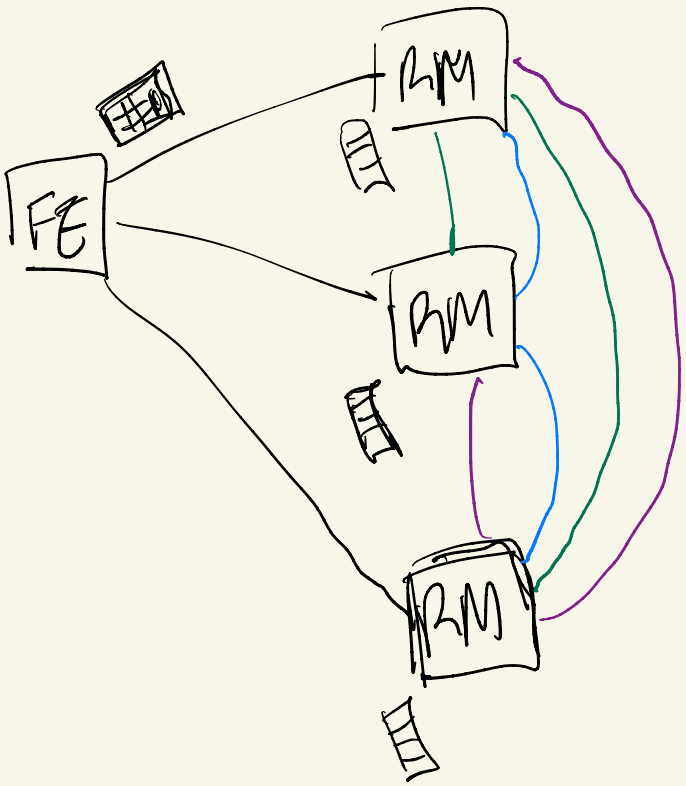
Active/Active

LevelDB / RocksDB

Passive



Active



(1) Complex failure scenario when leader fails you need a new leader before progress

(2) only contact leader or overwhelm

(1) RMs need to wait for responses

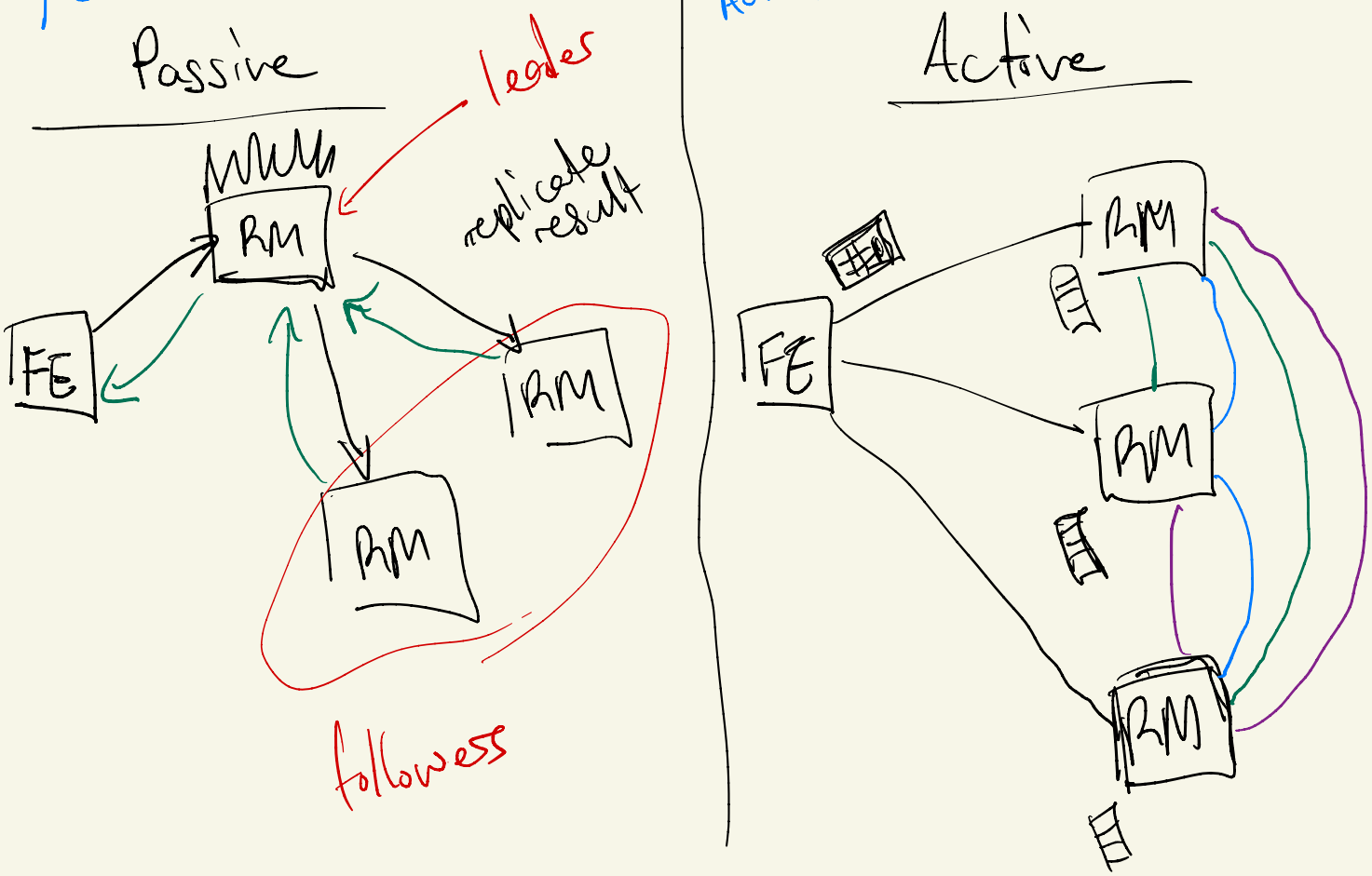
(2) lots of msg (expensive)

(3) hard to maintain consistent view

Raft / Zookeeper / etcd

Active/Active

LevelDB / RocksDB



consensus (agreement) = All or majority

All:
- node/network is slow = then replication is slow
- failure = stall

majority:
- can tolerate some failures / slow nodes

for Active Replication each RM only needs multicast from majority

for Passive the leader needs a majority of followers

Passive

1. leader fails the protocol slows down
2. fewer msg
3. the leader does a lot of work & followers do almost no work
4. can have non-deterministic code

Active

1. failure of small # has almost no impact as long as majority is not affected
2. lots of msg
3. every node does a lot of work
4. must be deterministic

choices of passive V. Active this based on performance during failures