

CSCI 1380 : Day 13

Global State / Replication



Last Class

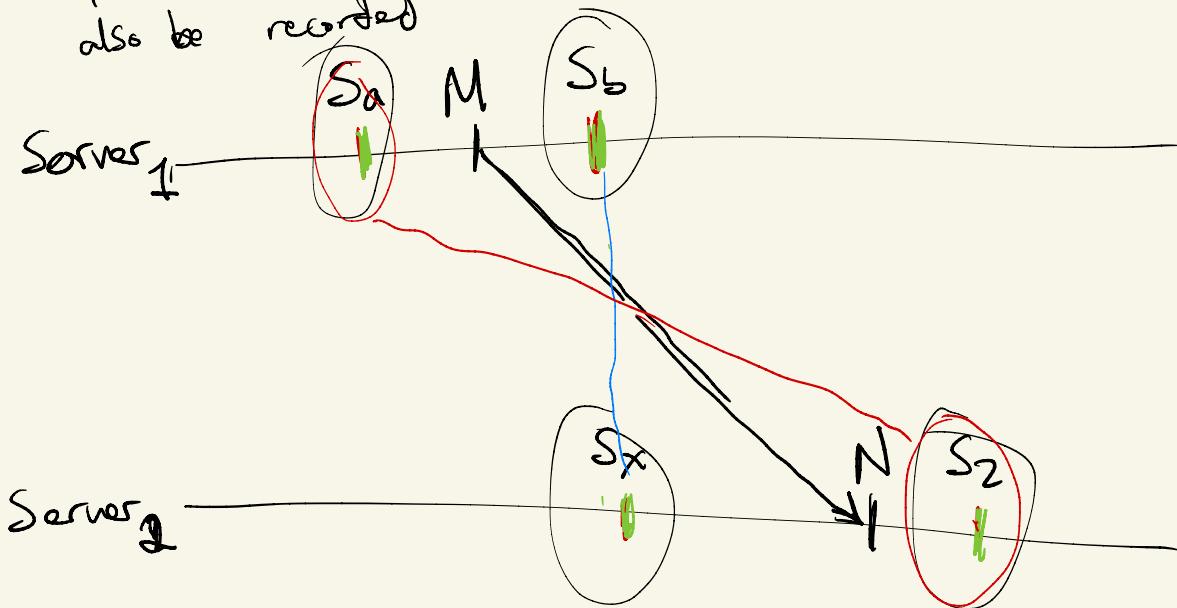
-
- (1) Various approaches for global snapshots
 - (a) time synchronized
 - (b) continuous (periodic)
 - (c) Manual (instantaneous)
 - (d) Chandy Lamport algorithm
 - (2) Using Vector clocks to detect inconsistencies

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

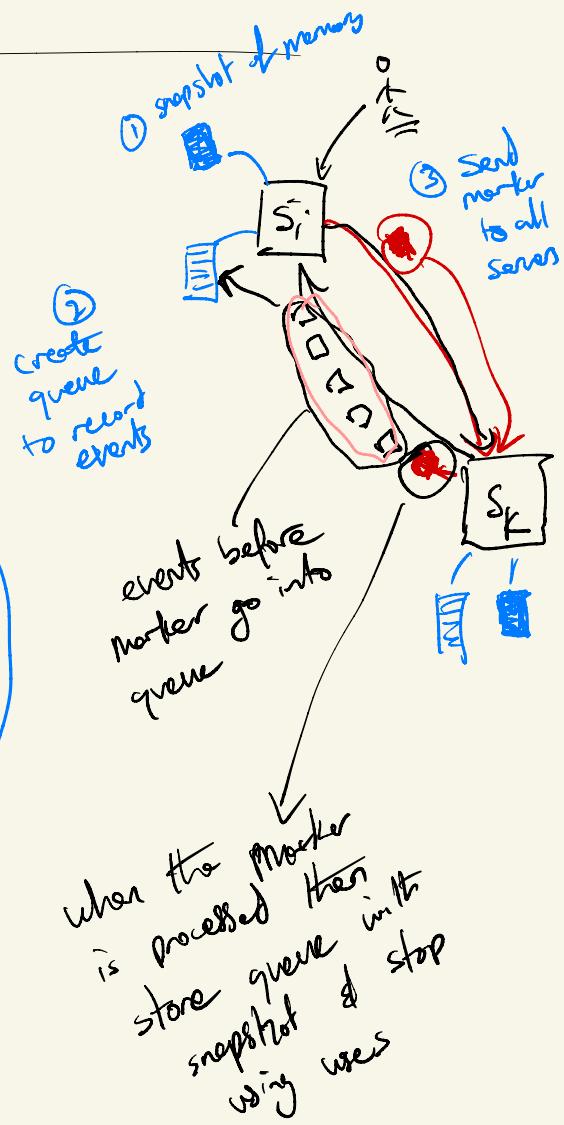


consistent = $[S_a, S_x]$, $[S_b, S_2]$, $[S_b, S_x]$

not consistent = $[S_a, S_2]$

Chandy-Lamport

- (1) Snapshot is triggered manually
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 ($\neq S_k$)
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 ends 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

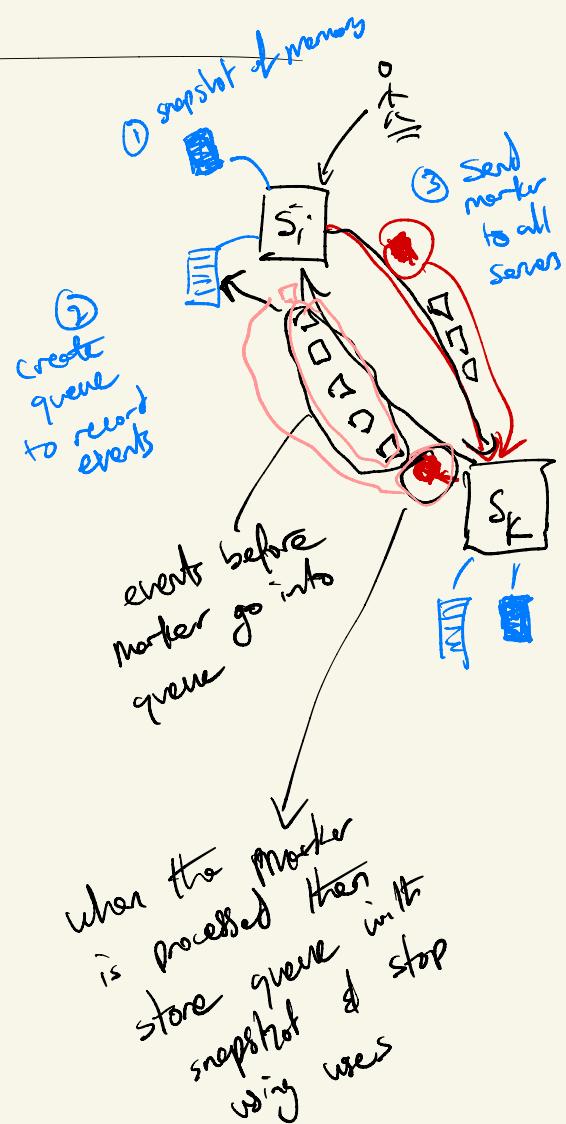
Chandy-Lamport

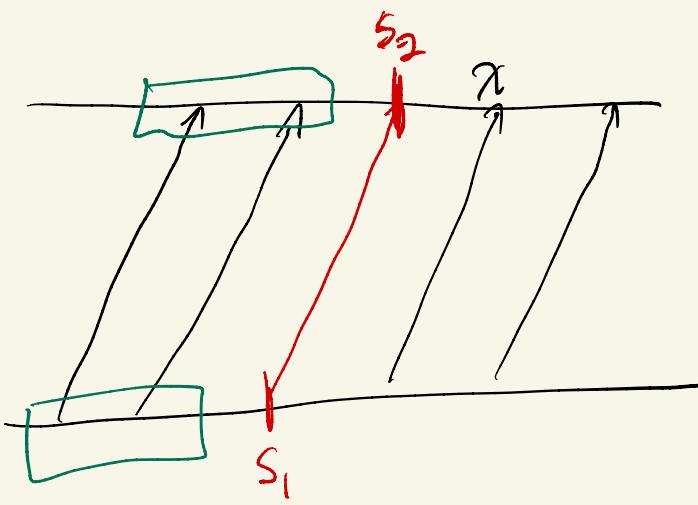
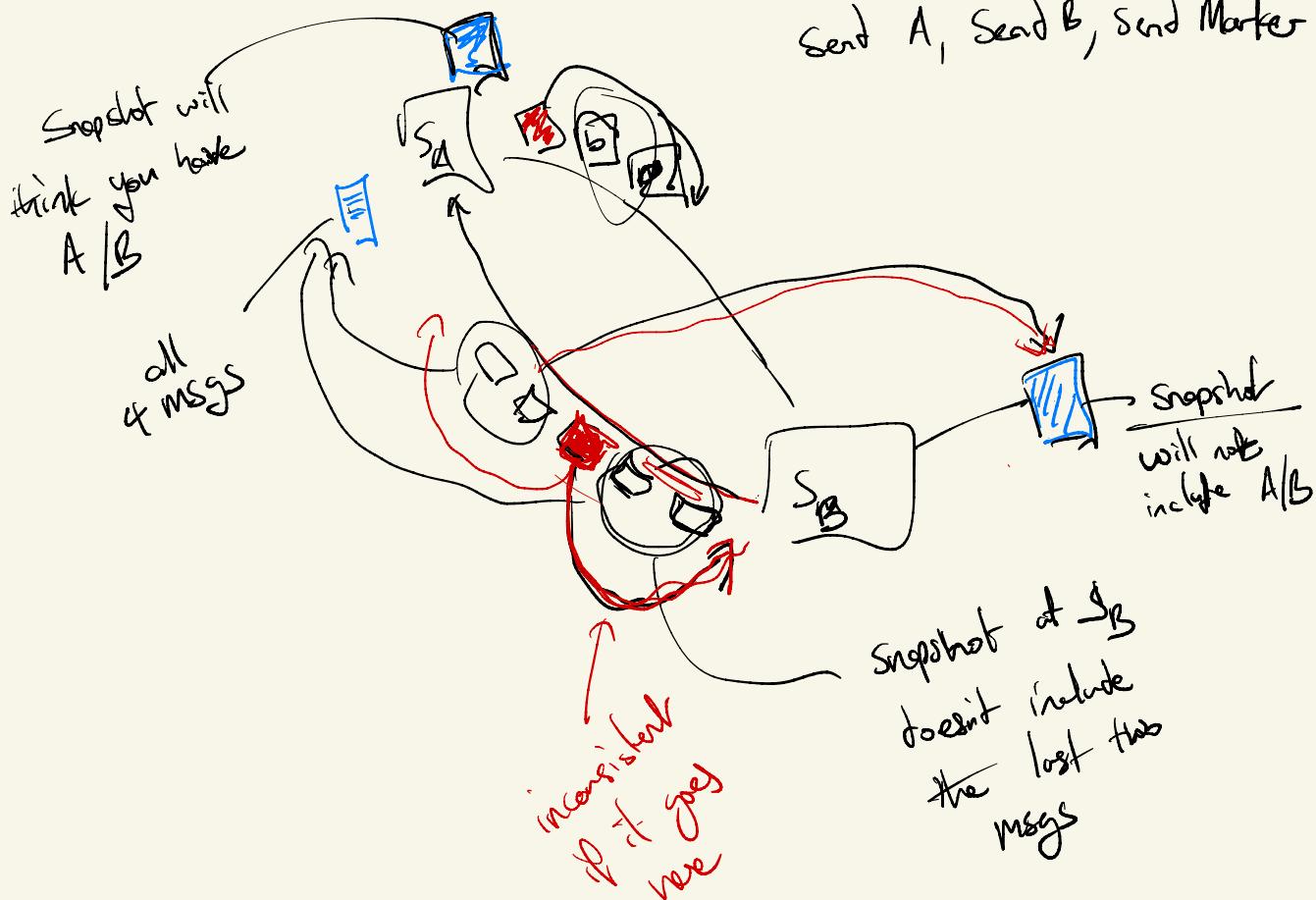
① Server failure

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

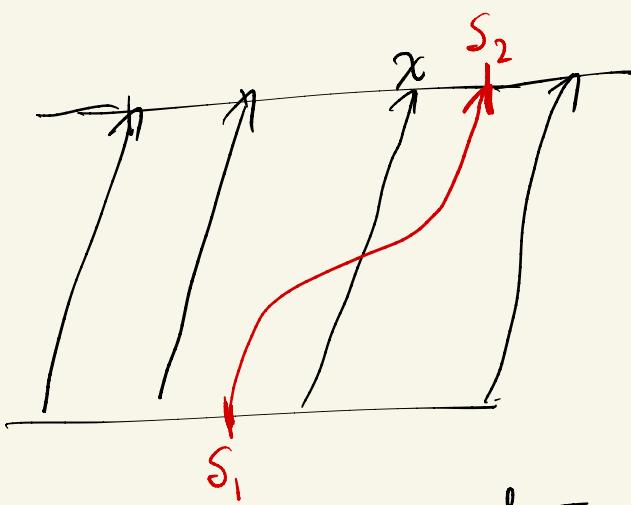
② Packet reordering

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





consistent snapshot



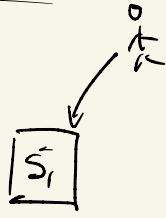
With reordering the marker is delivered after X & not before X hence a huge problem

Chandy-Lamport

- (1) Snapshot is triggered manually
① 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 ($\neg S_k$)
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 ends 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 p/t loss & all servers are operating correctly then termination is guaranteed

Termination Condition

Agreement / Correctness

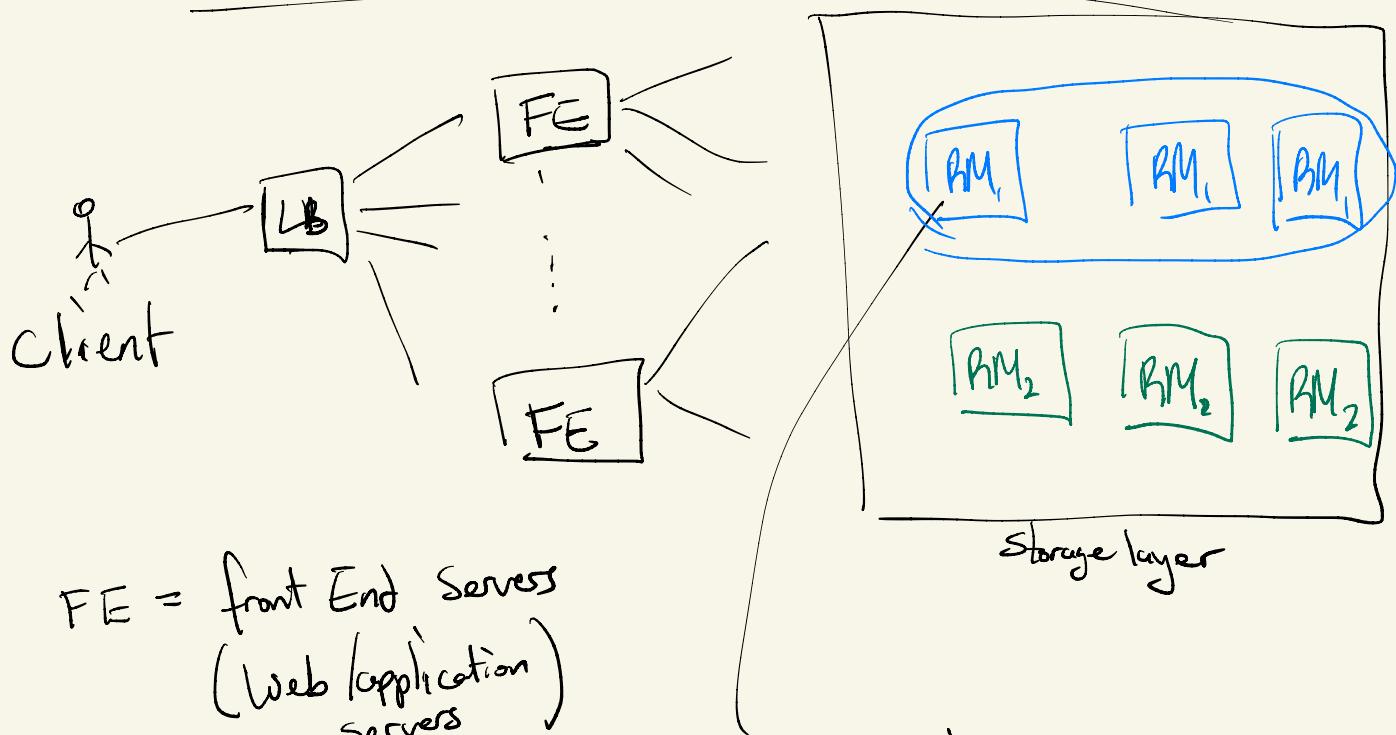
- ① each distributed snapshot has both new / sent event

- ② as long as there's no reorder

sent events are captured before sent markers

new capture in snapshot or in queue before you get a marker

Replication (storage layer)



FE = front End Servers
(Web Application)
servers

Replica Manager = store a copy of data

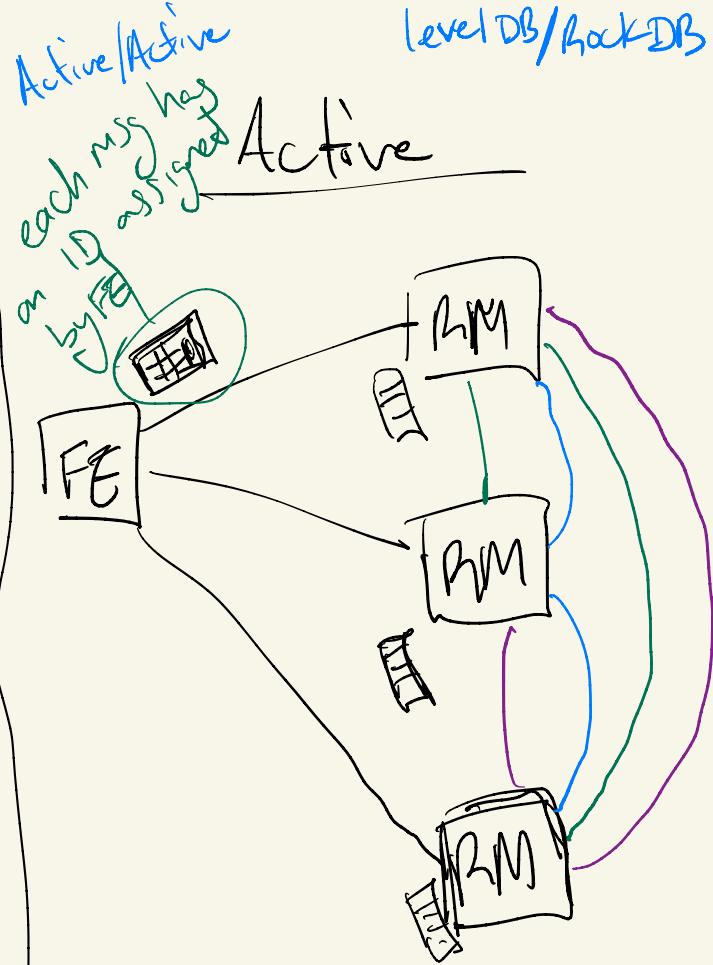
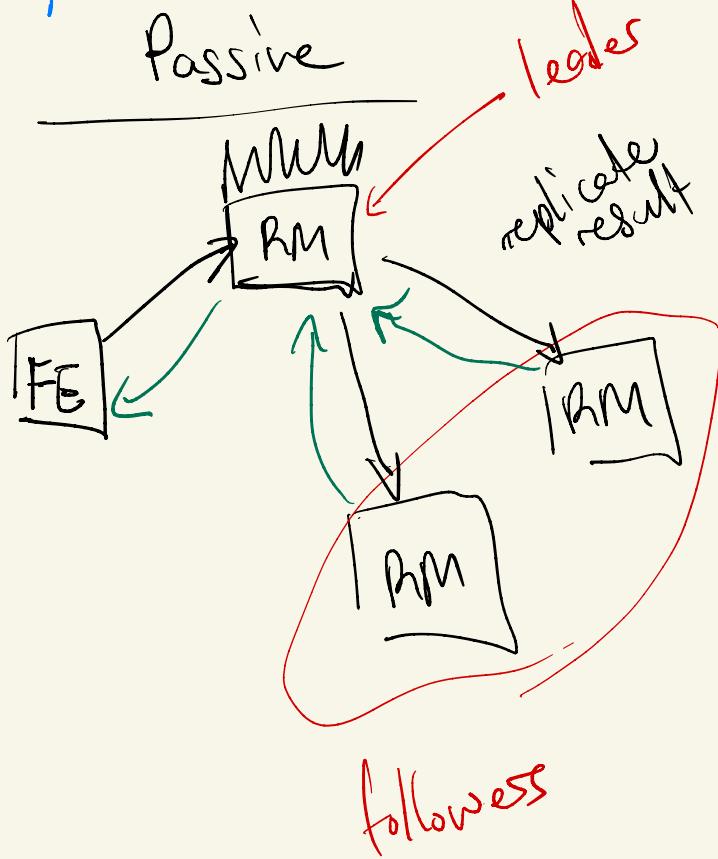
Blue RM's ⇒ store imgs

green RM's ⇒ 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

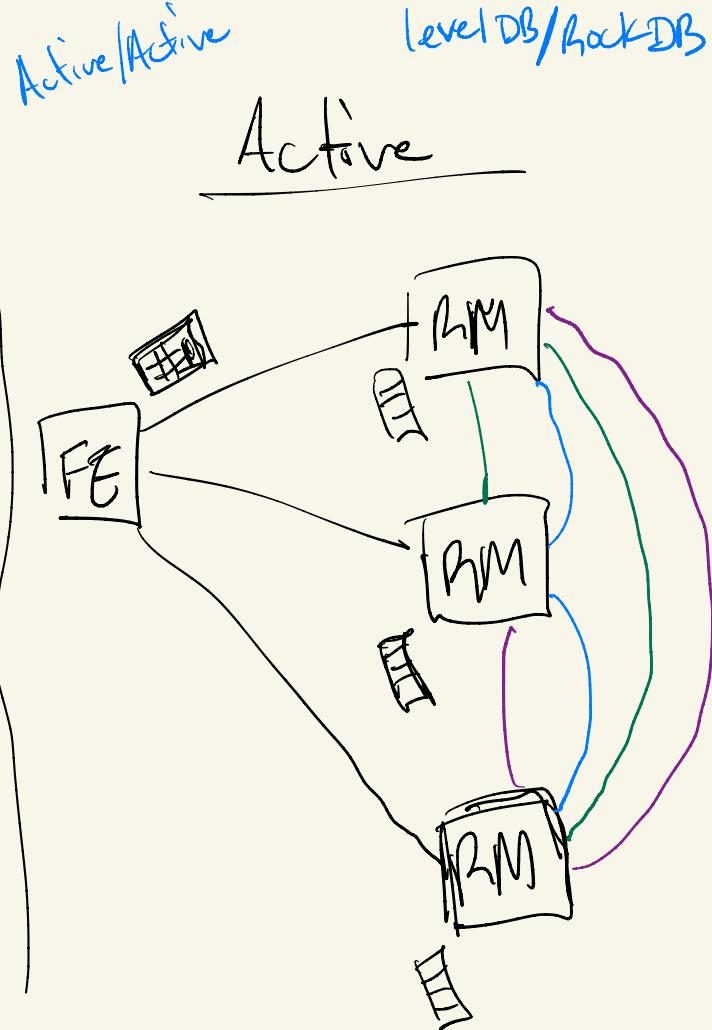
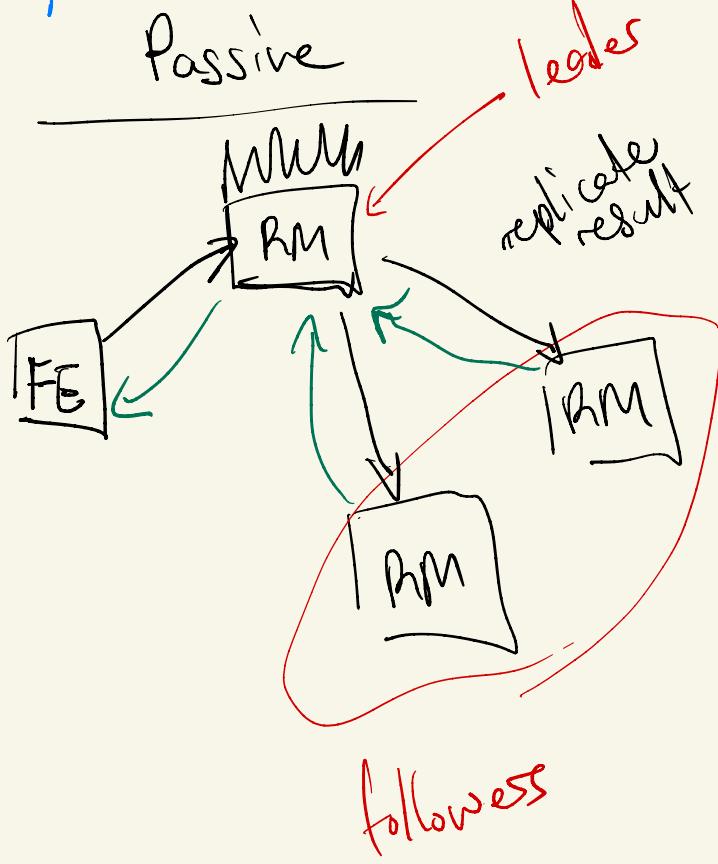
Raft | Zookeeper / etcd



- 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 responds to FE

- ① FE interacts with all RMs (each msg has an ID)
- 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 (events are processed in order of FE assigned IDs)

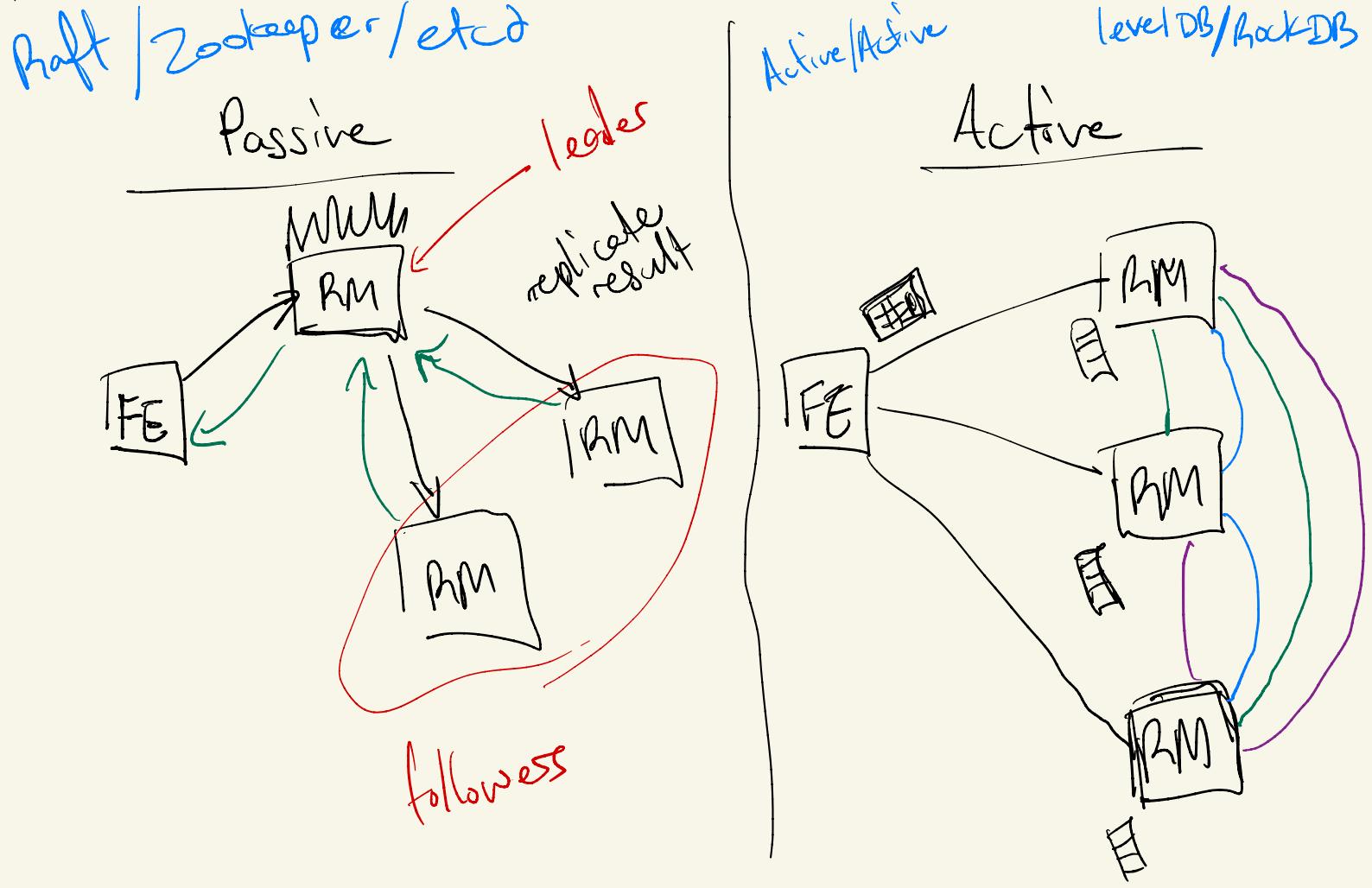
Raft | Zookeeper / etcd



① Complex failure scenario
When leader fails you need
a new leader before
progress

② only contact leader
can overwhelm

- ① RMs need to wait for responses
- ② lots of msg (expensive)
- ③ hard to maintain consistent view



consensus = All or

(agreement)

node/network
↳ 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

Summary

Replication
Terminology (FES/RMS)
Properties (Transparency)

Replication Strategies

- * Passive
- * Active
- * Tradeoffs between Active & Passive