Recent Approaches I:

"Secure" Hardware

as

Failure Detectors

WHY?

- QUESTION I (& MANY OTHERS) HAVE BEEN PONDERING

  "CAN ATTESTATION AND/OR CONFIDENTIAL COMPUTING HELP WITH DISTRIBUTED SYSTEMS?"

  ↳ PRETTY EASY TO FIND TODAY.

- RELATES TO FAILURE DETECTORS (WHAT WE STUDIED 2 WEEKS AGO)
- SOME OF YOU WERE CONCERNED ABOUT THE AGE OF WHAT WE STUDIED.

FIRST, A BIT ABOUT HARDWARE

- ATTESTATION

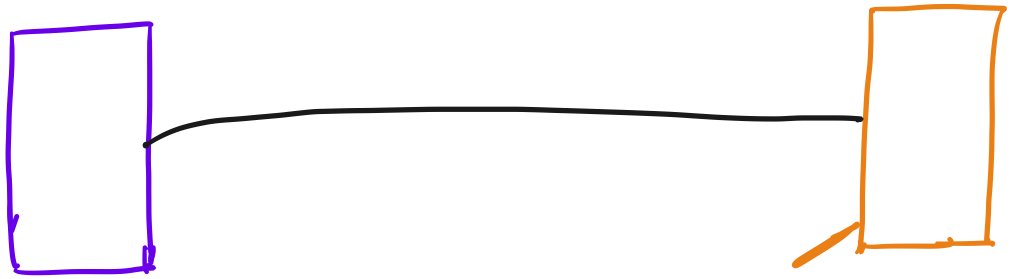
WHAT PROGRAMS ARE RUNNING ON THIS LAPTOP?

WHAT OTHER PROGRAMS ARE RUNNING ON THIS LAPTOP?

WHY?

How?

Remote Attestation

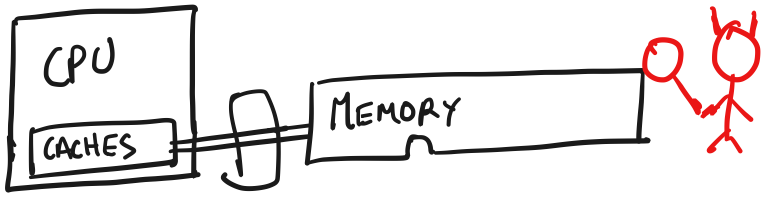


Need proof showing other node is runnin program P

Why?

How?

Enclaves / Confidential Computing



PROTECTING DATA FROM ATTACKERS WHO HAVE PHYSICAL ACCESS.

WHY?

TAMPER PROOF LOGS : PROTECTING DATA FROM MODIFICATION

Requirements

- Do not override previous entries
- Malicious or buggy server cannot respond to a request with incorrect or old values

→ For insertions: Check whether or not insertion succeeded

→ For lookups: Correct value is returned

→ No replay

How (today)?

→ Attest that a trusted log is used.

→ Require log to sign all responses

↳ With what key?

→ What information does a response include?

→ Prevent replay?

- Can we implement this today?
  → Type of adversary

→ Performance costs?

→ Safety?

Why focus on tamper proof logs?

Q: The power of tamper-proof logs to solve Byzantine consensus?

Previously

- Partial synchrony + reliable channels + HMAC/Signatures

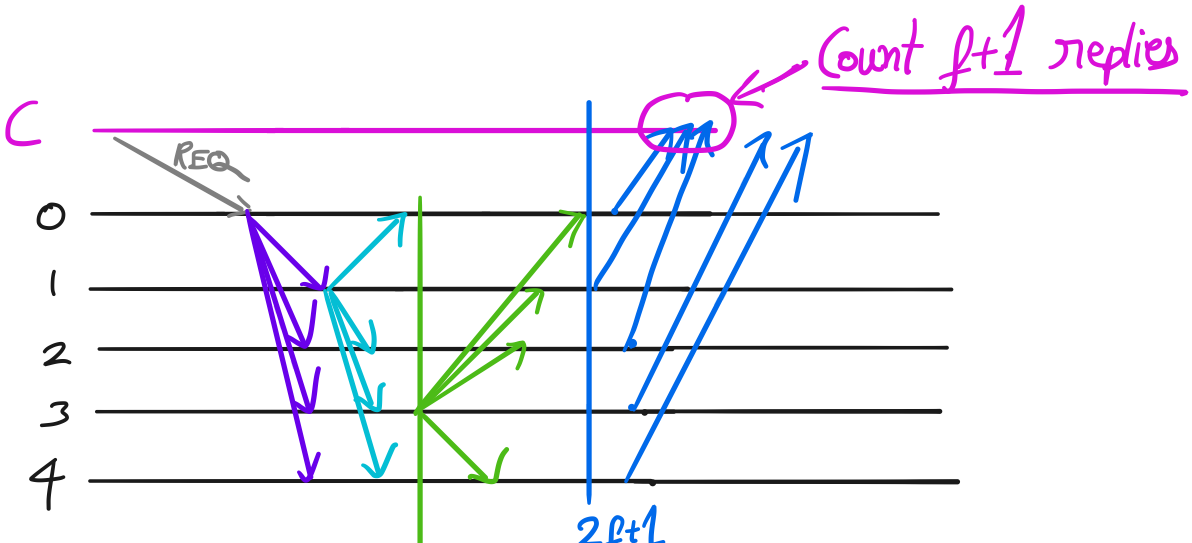
$3f+1$

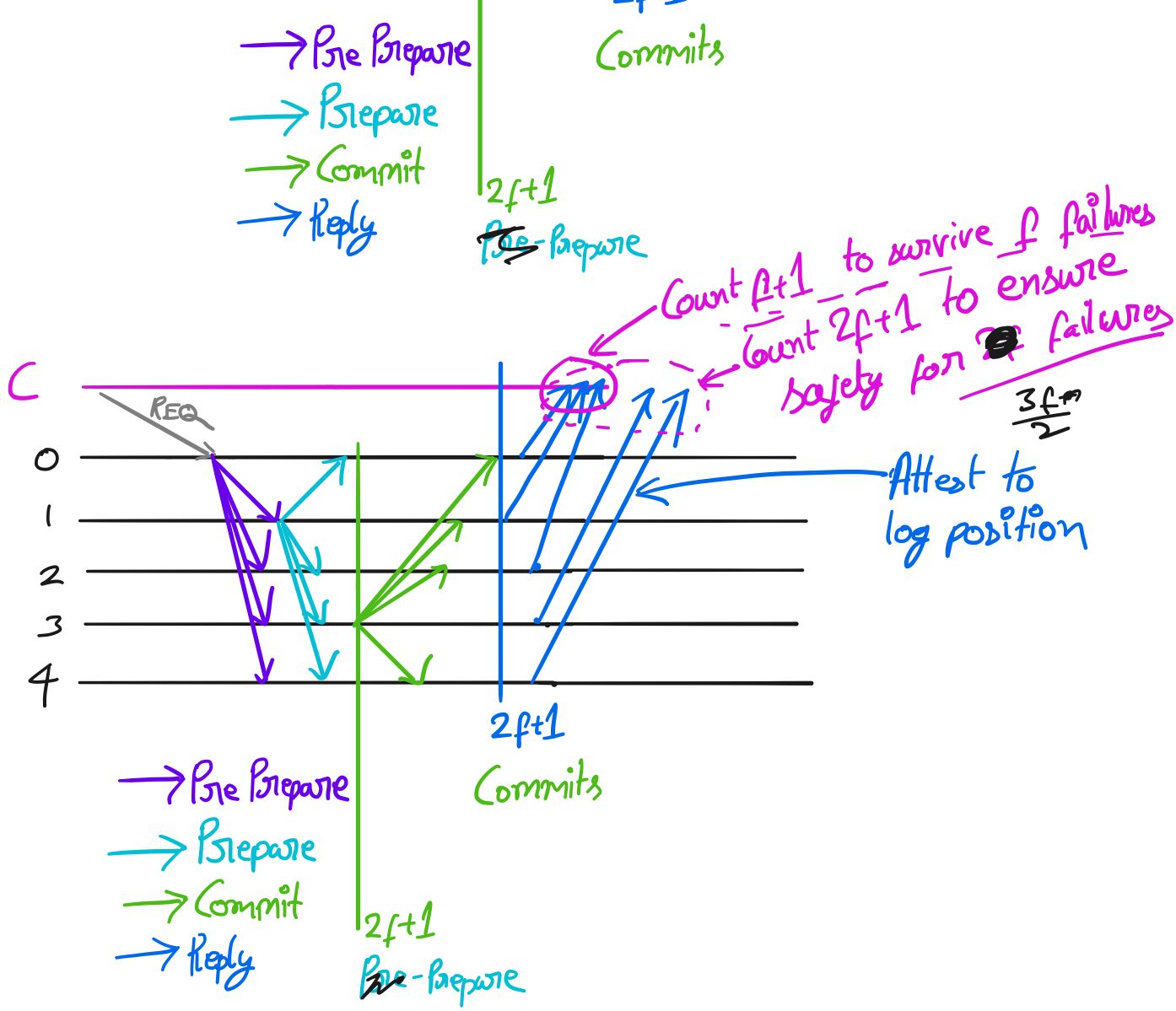
- Can we do better?

Q1: Do we need to change interfaces?

Q2: How to change the protocol?

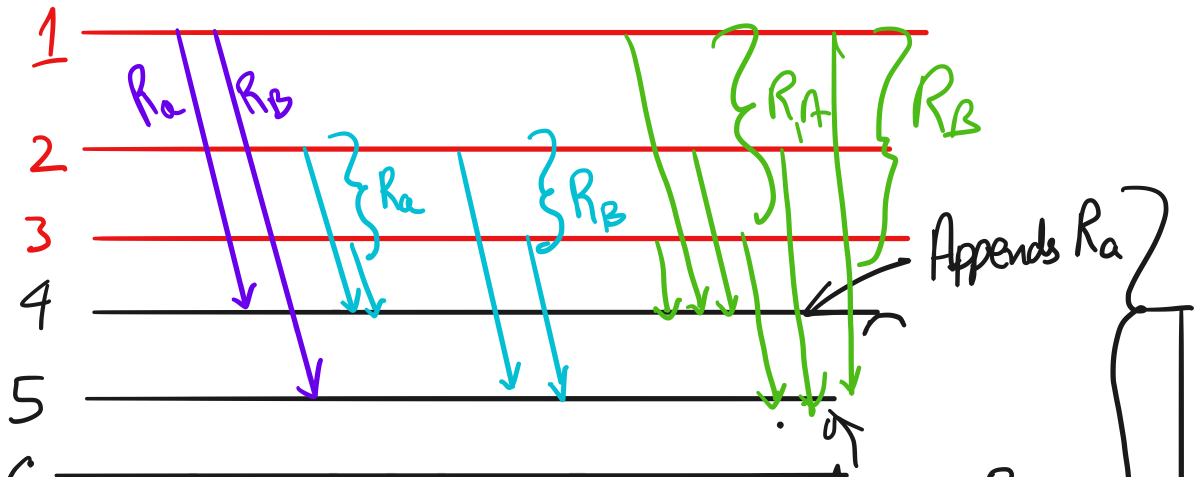
Attempt 1

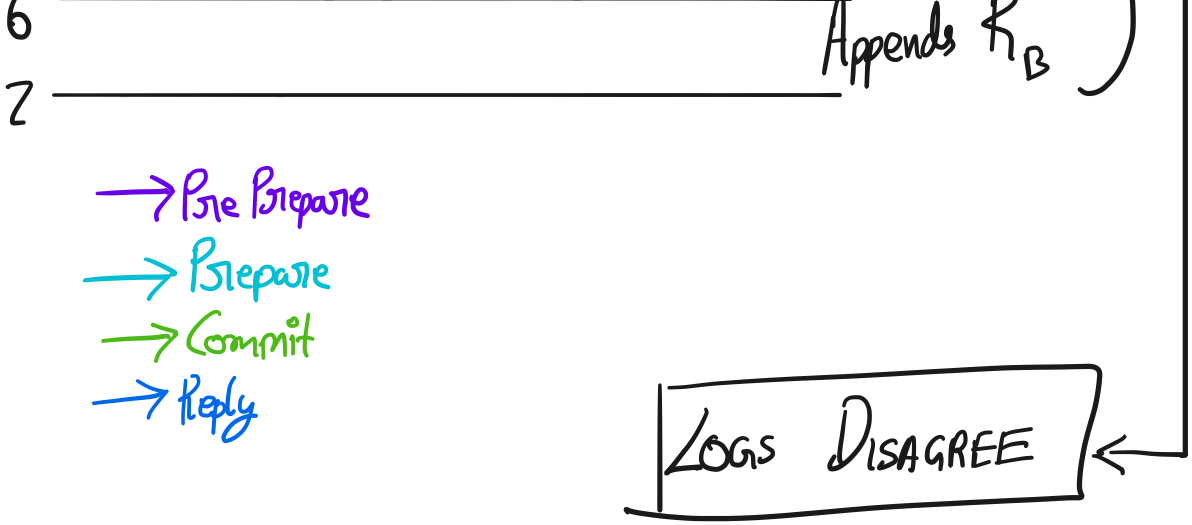




Safety: Log agreement $\leftrightarrow$ Linearizable order

Cannot ensure liveness. Why?

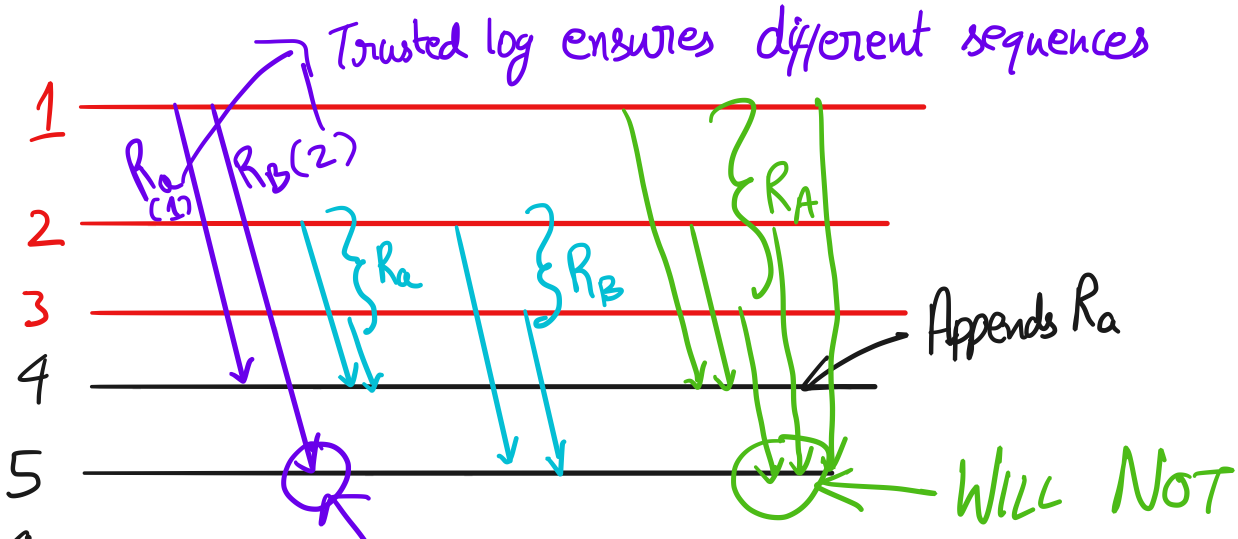


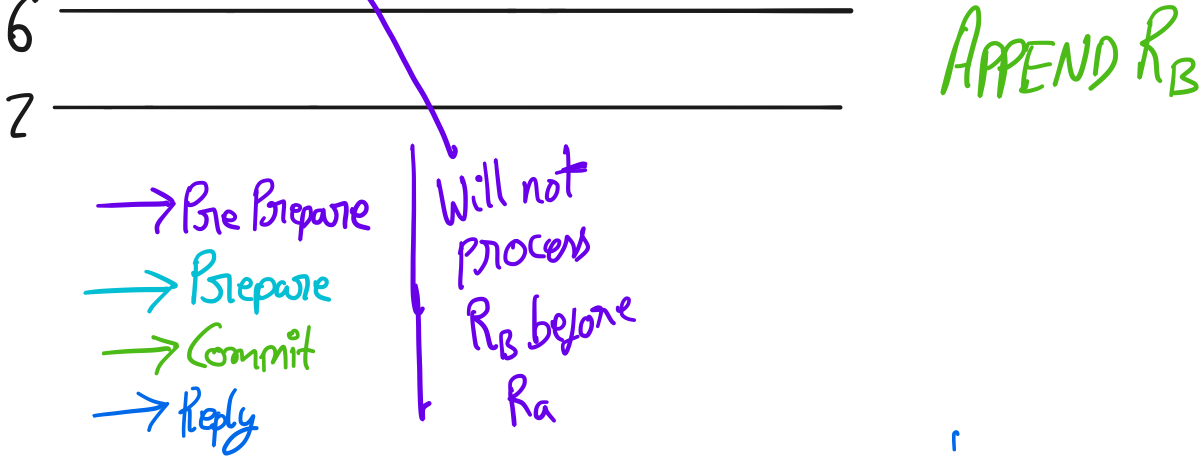


Problem: Attestation after commit does not prevent malicious processes from equivocating

Solution: Log and attest all messages on the way.

1. Process a single (or a fixed set) of client requests at a time
2. Add each message to a log before sending it. Include attestation indicating where in the log a message shows up.
3. Receivers process messages in log order.



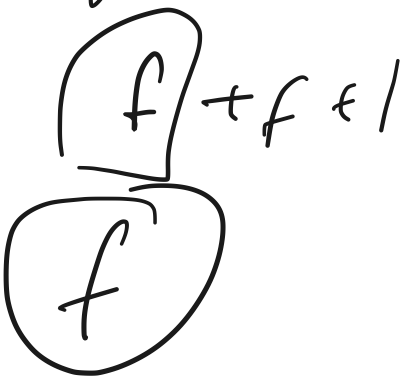


CONSEQUENCE: Faulty nodes cannot cause logs to diverge.

Q: Is this enough to ensure liveness?

PBFT view change: $2f$ valid view change requests from other replicas. Why?

View change here:



Fault Tolerance Thus Far

Need { f+1 responses at the client
       f+1 view change requests } ...

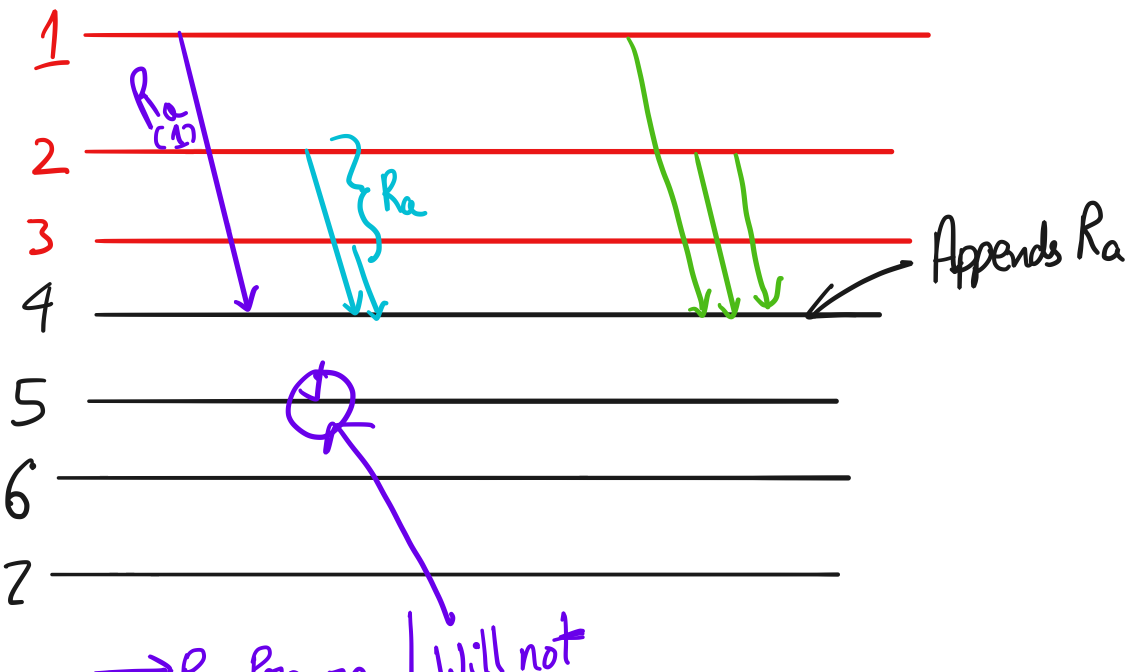
Up to f nodes can fail

Need 2f+1

ARE WE DONE? (f)

Observations

① Faulty processes can act so that a correct processes log Rs ahead of others



(2) Clients are responsible for preventing Denial of Service?

How?

(3) Is this a problem?

Where are we on this / other protocols.

- ToR / Private relays?

- Others?