Byzantine Fault Tolerance — BFT

You looked at this material before
↳ PLEASE PARTICIPATE

- FAIL-STOP

- FAIL-RECOVER

- REALITY  Bugs or attacks don't always lead to crashes!
  - If only they did

Goal: Avoid assumptions about the behavior of faulty

nodes :- BYZANTINE FAULT TOLERANCE

Where : High-assurance (The Real Byzantine Generals - Driscoll et al '04)

- Submarines
- Space

Cryptocurrency $$$

Supply chains, critical cloud services (soon)

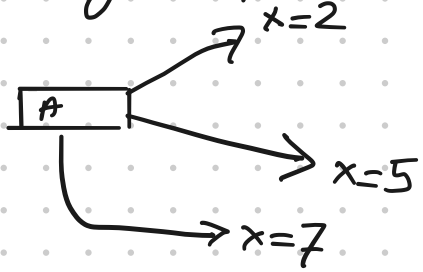
Really, a cost – benefit tradeoff

- BFT often slower, needs more computation, etc.
  So really about whether it is worth it to you!

Question : What can a faulty node do in the byzantine model?

- Crash 
- Not send some messages (omission)
- Pretend to be a different node (impersonation)

- Send contradictory messages (equivocation)



- Corrupt internal state

- Implement a different protocol

- ...

Q: Which of these do fail-stop protocols already handle?

Q: Which can we detect?

Problems that are FREQUENTLY A FOCUS

- IMPERSONATION
  - Why

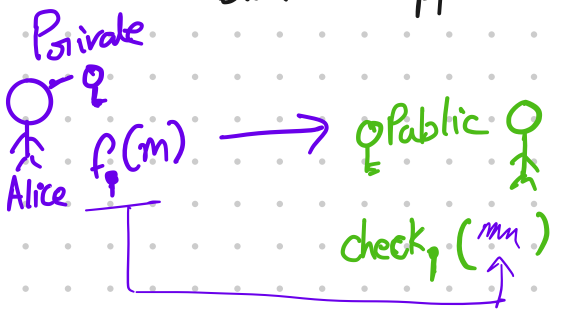
- Standard approach: Authenticate? Digital signatures?

Emerging problems with the approach

- EQUIVOCATION

- Why?

- Standard approach : Authenticated channels / Transferrable authentication



Signed messages +
Protocol to exchange messages

- Concerns: cost

- LIVENESS / TERMINATION

FROM LAMPORT

Q: From last week: assumed model

- Asynchronous

AUTH CHANNEL → SM(n)

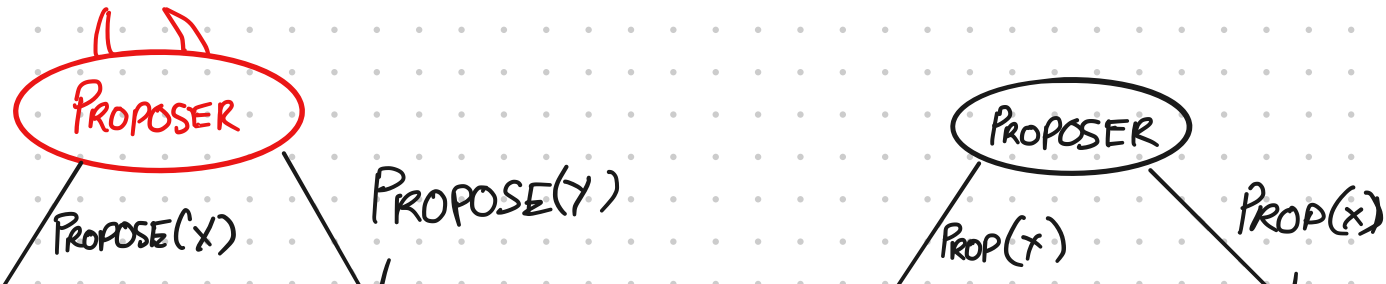
- Partially synchronous
- Synchronous

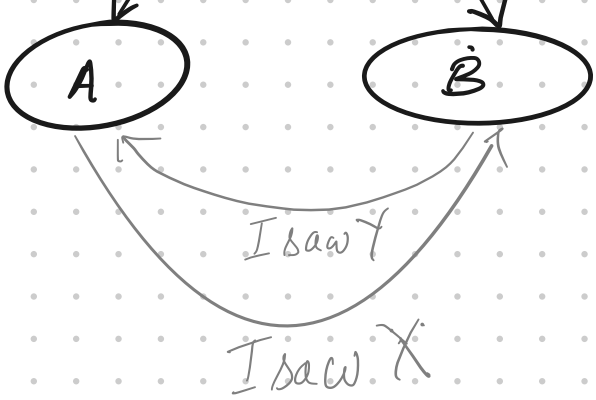
Why?

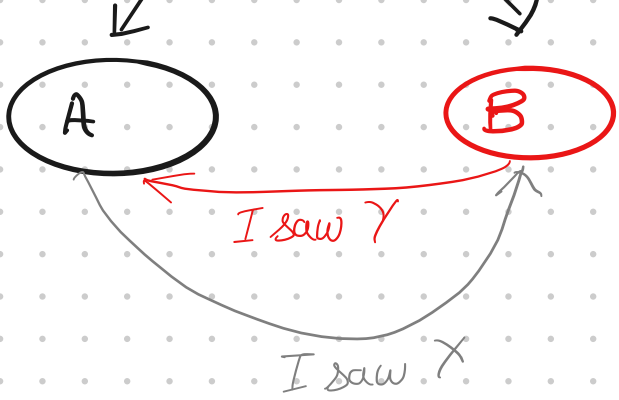
OM(n): Without auth. need $N \geq 3f+1$

Req.
- Agreement: All non-faulty nodes agree (IC1)
- Validity: Dgined assuming LEADER (DESIGNATED PROPOSER)
  If proposer correct →
  non-faulty nodes decide proposed value
- Termination: Eventually decision is reached.

Why $N \geq 3f+1$



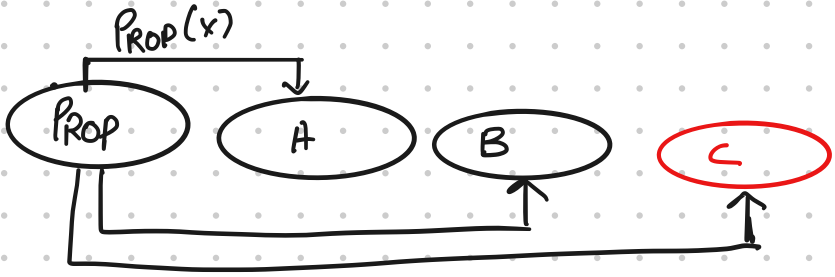




What should A do?

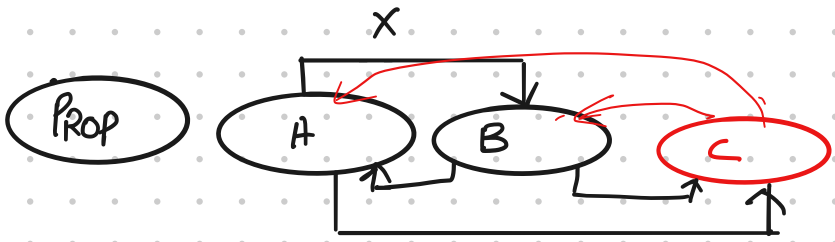
Q. Why would authenticated channels help?

Problem also suggests path forward: use voting to figure out proposal



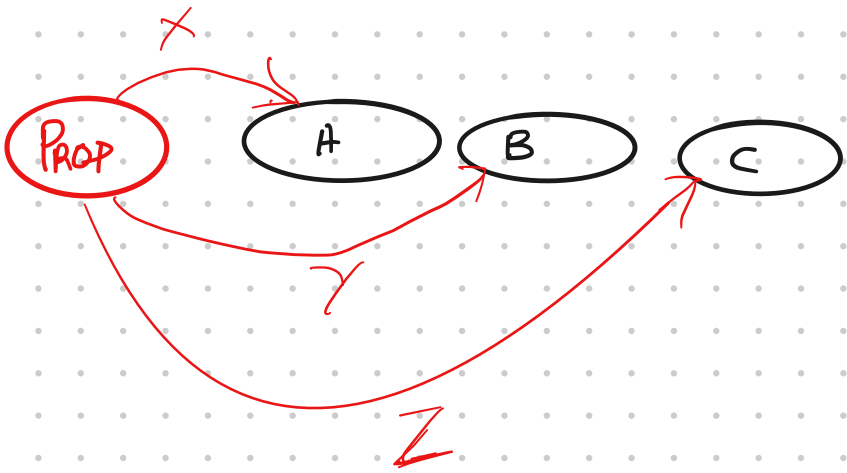
A

From	Value
Prop	X
B	
C	



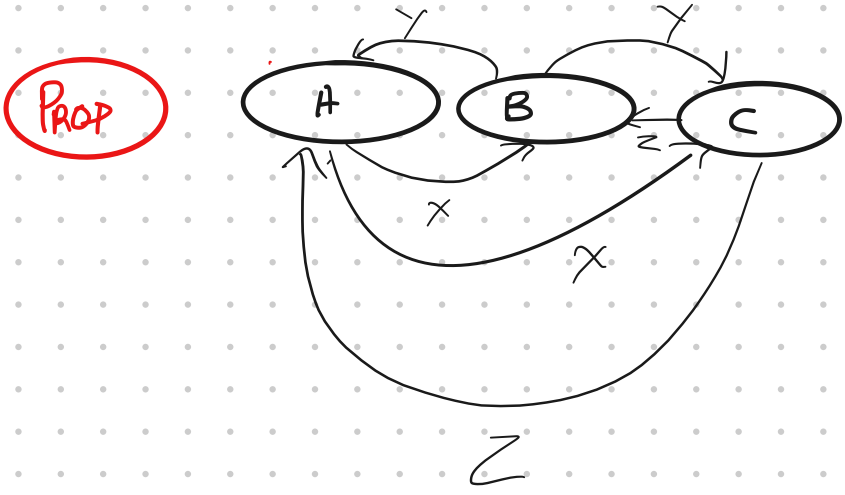
From	Value
Prop	X
B	X
C	?

{X}



A

From	Value
Prop	X
B	
C	



From	Value
Prop	X
B	Y
C	Z

Pick deterministically.

Trickier as we increase number of faulty servers but same principle

Synchrony is hard — what about partial synchrony

PBFT: Note, other protocols are available, this is just the popular one.

- Assumes authenticated channels
- Messages are broadcast
- Client counts responses to determine if op committed
  ↳ Why?

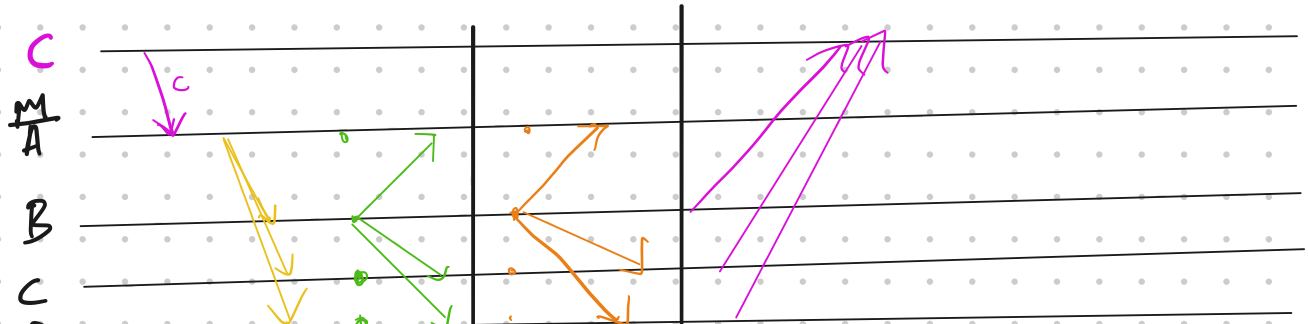
Similar to other protocols we saw: Quorum based. + W/Leader

But quorum of

$2f+1$: A majority of nodes that make a decision are correct $f < f+1$

Need $\geq 3f+1$ nodes
↳ Why?

Leader: Designated proposer ↔ Decides what to propose



D

Wait for ≥ 2f+1

- PRE-PREPARE ($\langle c \rangle_\sigma$, $\langle$ View (term), n, hash($\langle c \rangle_\sigma$)$_A$)

  ↳ Signed by c. Why?

  Signed by A.

- PREPARE ($\langle$ view, n, hash($\langle c \rangle_\sigma$), B$\rangle_B$)

  ↳ Why?

Wait for 2f+1 prepare for a previously pre-prepared command.

→ f+1 correct nodes agree

$\langle$ view, n$\rangle \longrightarrow c$

- COMMIT ($\langle$ View, N, hash($\langle c \rangle_\sigma$), B$\rangle_B$)

Wait for 2f+1 commit command for prev. prepared command.

Client waits for f+1 responses

- Why $f+1$ not $2f+1$?

Liveness

- Leader is faulty?

Client triggers view change

View change/leader election

Two requirements

(1) Faulty nodes cannot monopolize leadership
→ View ID determines leader

	A	B	C	D
Leader	0	1	2	3
In View	4	5	6	7

...

(2) Faulty nodes cannot trigger view change

- Huh?

- How?

  View active until $2f+1$ nodes agree.

BFT TODAY

- Regaining interest

- Question: Can we reduce overheads?

- How? Add information — failure detector

  - Trusted hardware

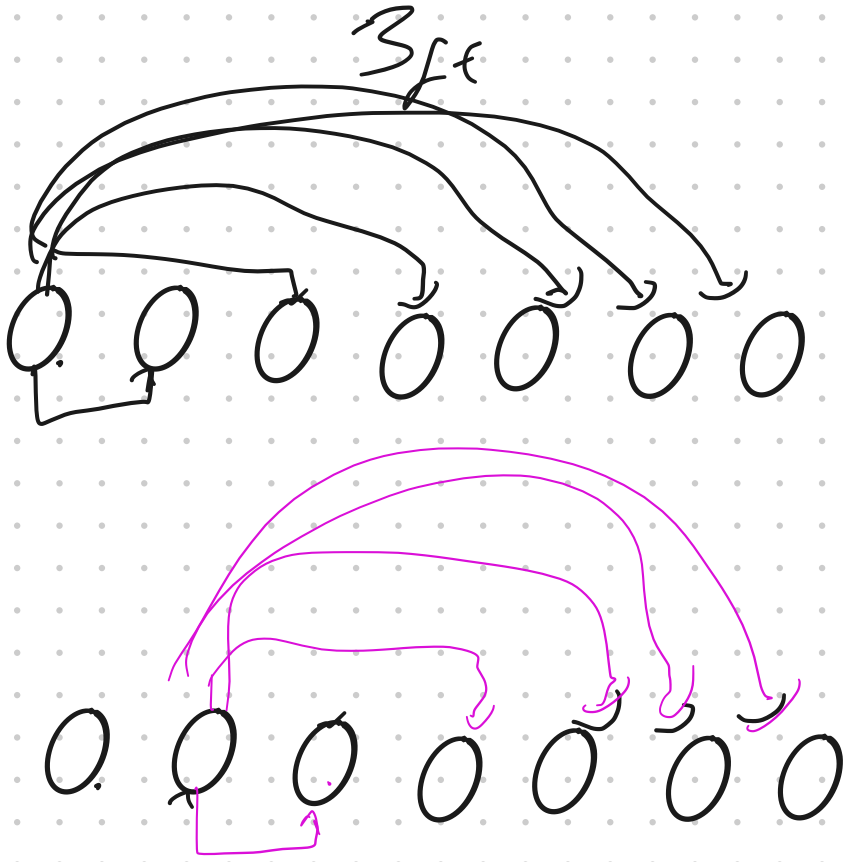
    ↳ TPM

    → Enclaves

    → ...

  - Timing assumptions

- ???


$\exists f$