Decentralized Systems Engineering

CS-438 – Fall 2025

DEDIS

Bryan Ford and Pierluca Borsò-Tan

EPFL

Credits: B. Ford, K. Driscoll, R. Stutsman, M. Freedman, K. Jamieson, R. Morris, F. Kaashoek, N. Zeldovich, L.Q. Torres, et al.

Consensus

Paxos, Multi-Paxos & TLC

(Homework 3)

Paxos: review

Key properties:

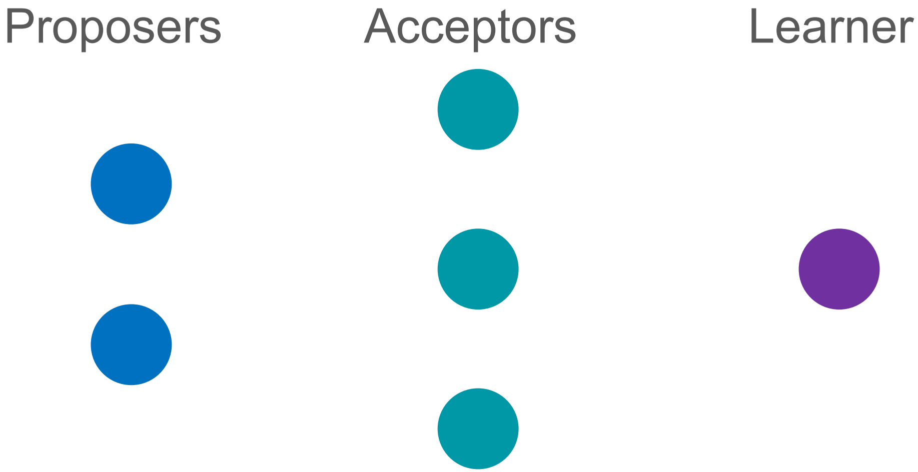
- Safety: all nodes agree on a (single) decision
- Liveness: eventually something is decided

Assumptions:

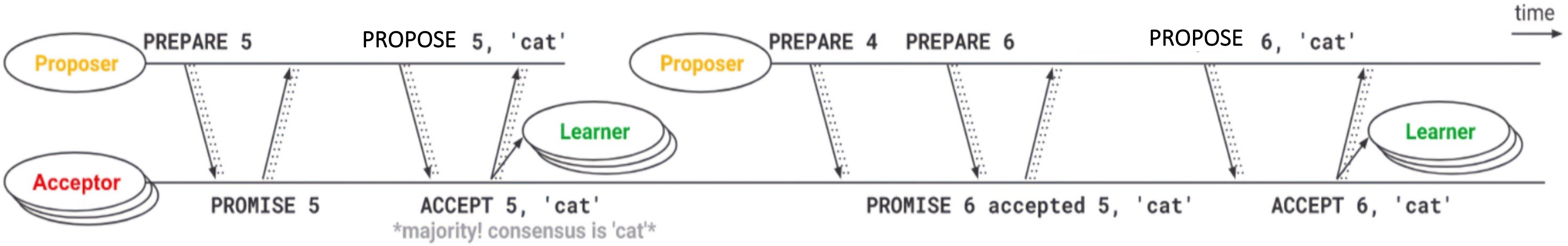
- Crash-stop model
- Partially synchronous
- # acceptors = 2f + 1

Protocol (choose 1 value):

- Phase 1: Prepare/Promise
- Phase 2: Propose/Accept



Paxos: review (cont'd)



⇨ Proposer wants to propose a certain value:
It sends **PREPARE** **IDp** to a majority (or all) of **Acceptors**.
IDp must be unique, e.g. slotted timestamp in nanoseconds.
e.g. Proposer 1 chooses IDs 1, 3, 5...
Proposer 2 chooses IDs 2, 4, 6..., etc.
Timeout? retry with a new (higher) IDp.

⇨ **Acceptor** receives a **PREPARE** message for IDp:
Did it promise to ignore requests with this IDp?
Yes -> then ignore
No -> Will promise to ignore any request lower than IDp.
Has it ever accepted anything? (assume accepted ID=**IDa**)
Yes ->Reply with **PROMISE** **IDp** **accepted** **IDa**, **value**.
No -> Reply with **PROMISE** **IDp**.

1 If a majority of acceptors promise, no ID<IDp can make it through.

⇨ Proposer gets majority of **PROMISE** messages for a specific IDp:
It sends PROPOSE **IDp**, **VALUE** to a majority (or all) of **Acceptors**.
Has it got any already accepted value from promises?
Yes - > It picks the value with the highest **IDa** that it got.
No -> It picks any value it wants.

⇨ **Acceptor** receives an PROPOSE message for IDp, value:
Did it promise to ignore requests with this IDp?
Yes -> then ignore
No -> Reply with **ACCEPT** **IDp**, **value**. Also send it to all **Learners**.

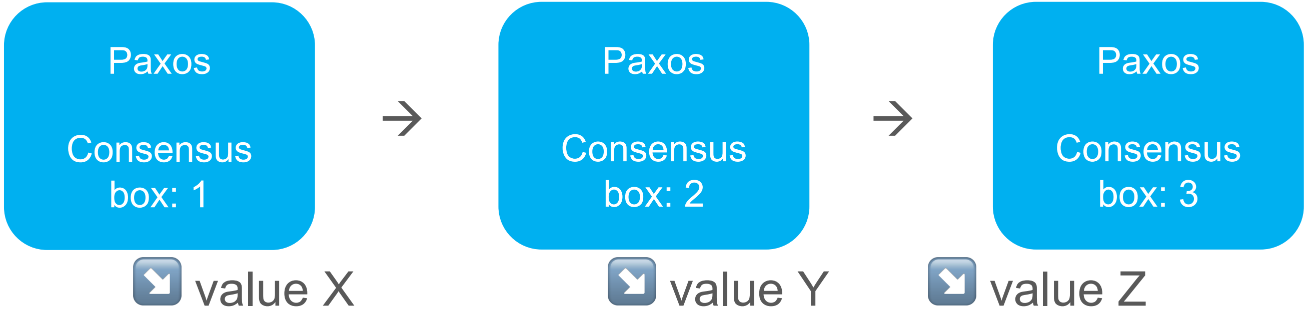
2 If a majority of acceptors accept IDp, value, consensus is reached. Consensus is and will always be on value (not necessarily IDp).

⇨ Proposer or **Learner** get **ACCEPT** messages for IDp, value:

3 If a proposer/learner gets majority of accept for a specific IDp, they know that consensus has been reached on value (not IDp).

Multi-Paxos

- What if we want to choose multiple values ?



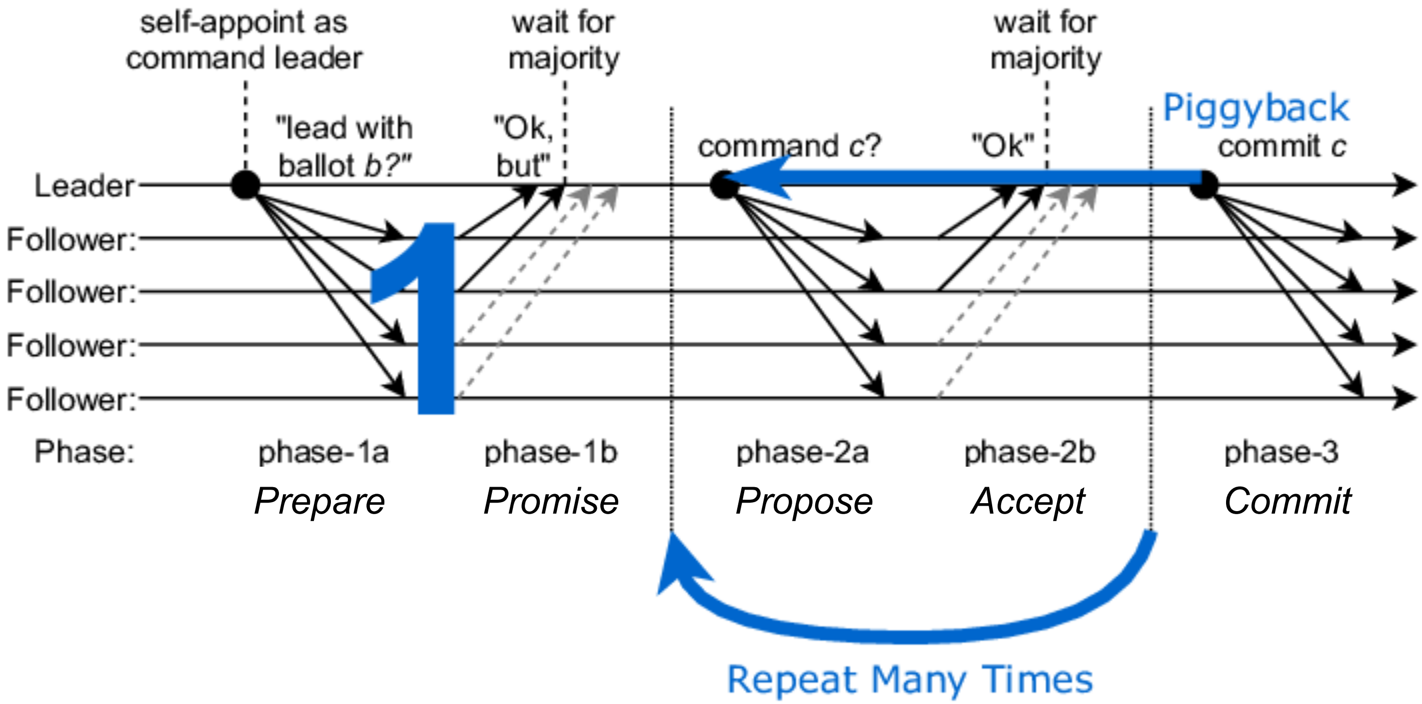
How do we make that happen ?

- Network messages are tied to a box

How do we know the highest box number ?

- Leader-based Paxos, logical clocks, etc.

Multi-Paxos: Leader-based optimization



Source: A. Charapko, PigPaxos

Logical Clocks

Time is ***hard*** (cf. "UTC is enough for everyone…right?")

… and it can't be trusted in a distributed system

… unless you're Google and you control time (in your datacenter)

Can we substitute time with a *logical* alternative ?

→ pass a logical clock time C (a counter) along with events

- Lamport clock — A caused B → C(A) < C(B)

  Each message contains the logical time, receiving updates the local clock

- Vector clocks — C(A) < C(B) → A caused B

  Similar to G-Counter CRDT, one counter per node

- Threshold logical clock — specialized for threshold applications

Threshold Logical Clocks

Paxos

Consensus box: 1

↘️ value X

↘️↘️↘️ TLC =2

↘️↘️↘️ TLC =3

- When a consensus is reached
  → broadcast logical clock advance

- When a threshold (quorum) of clock advances has been received
  → move to the next box, ***even if*** we haven't witnessed the consensus !

Threshold Logical Clocks

Some interesting properties:

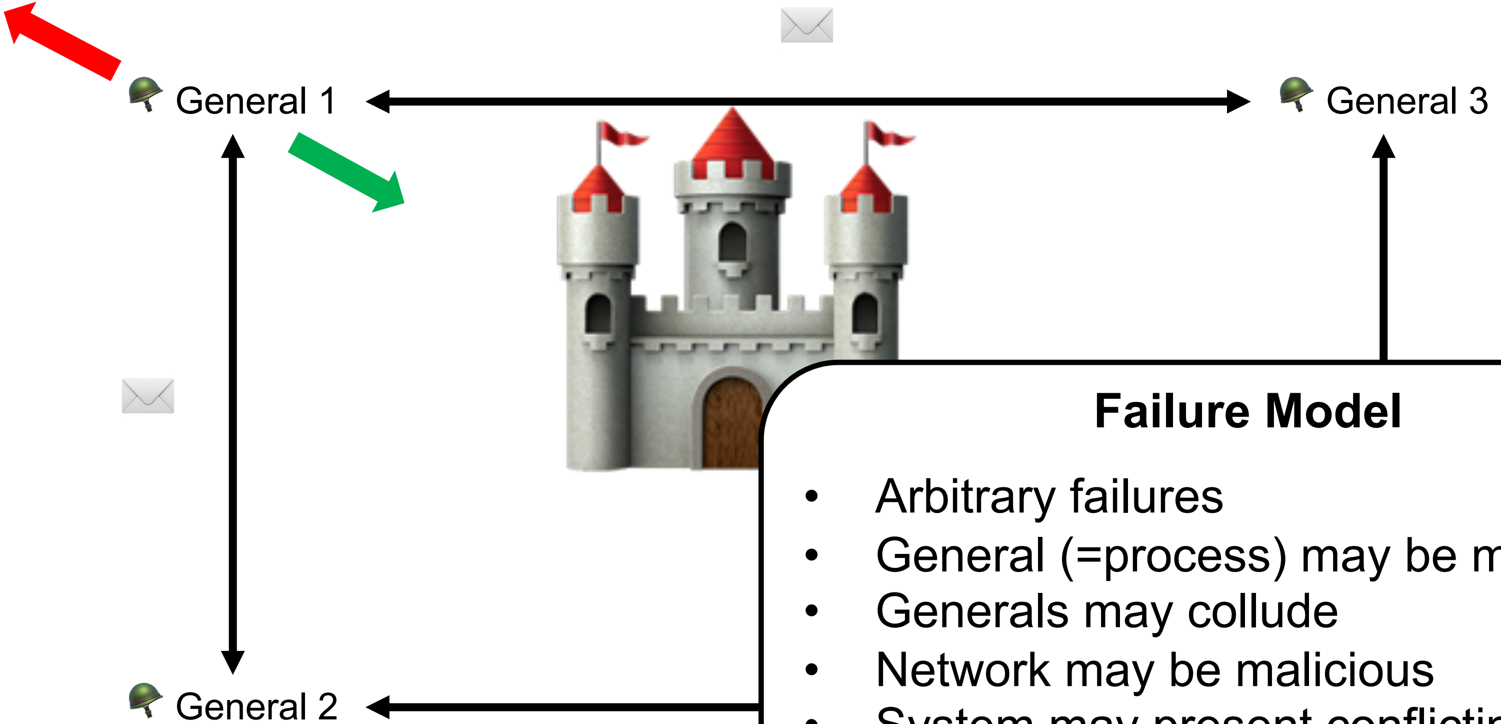
- Works without synchronous assumptions or timeouts
- Works despite malicious nodes (with some added crypto)
- Low latency, low-bandwidth usage
- It makes consensus implementation much simpler !
- It exposes a “synchronous” abstraction to higher layers

Going Byzantine

Faults and how to survive them

Byzantine Generals Problem

- Coordination = Victory
- No coordination = Loss



Failure Model

- Arbitrary failures
- General (=process) may be malicious
- Generals may collude
- Network may be malicious
- System may present conflicting info
- Computations may be incorrect

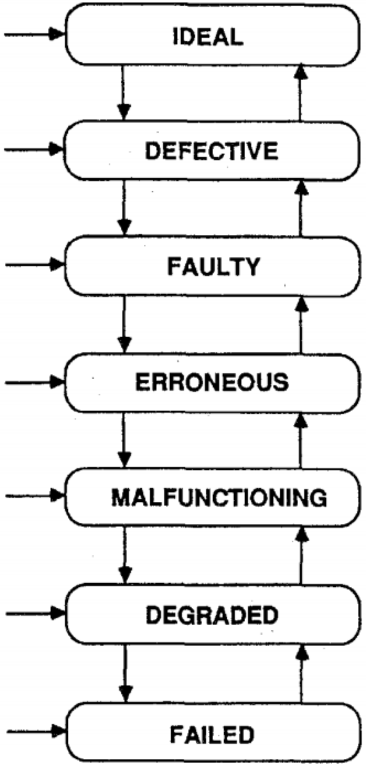
When stuff goes wrong…

- Fault: underlying defect
  → active – injects errors in the system
  → passive – latent

- Failure: system not producing the desired result
  → 1+ fault(s) have made the system useless

- Fault-tolerance:
  building reliability out of unreliable components

- **Denial is not a strategy – things will fail !**

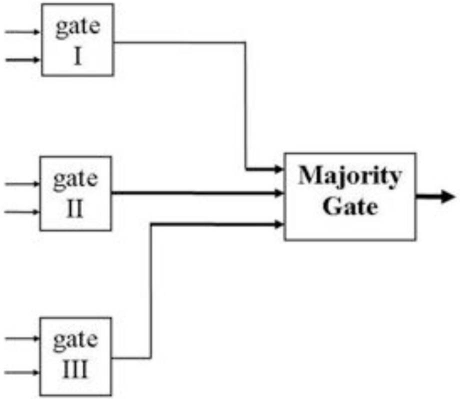


Redundancy

- Fundamental principle to build fault-tolerant systems

- Redundancy in digital design
  - Detect deviations and automatically restore correct behavior
  - Space-redundancy: state
  - Time-redundancy: (re)transmission

- Redundancy in computer systems
  - Coding
  - Data replication
  - N-version programming



Byzantine faults – Causes

- Deliberate tampering
- Software bug
- Hardware failure

What kind of hardware failures could cause a Byzantine fault ?

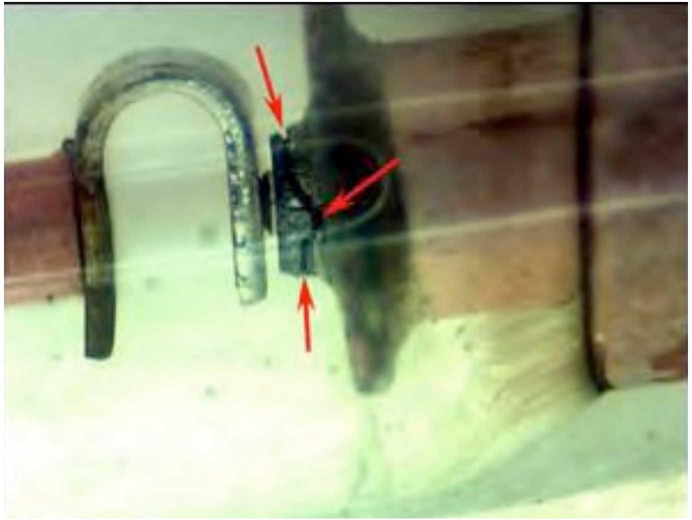
Network partitions

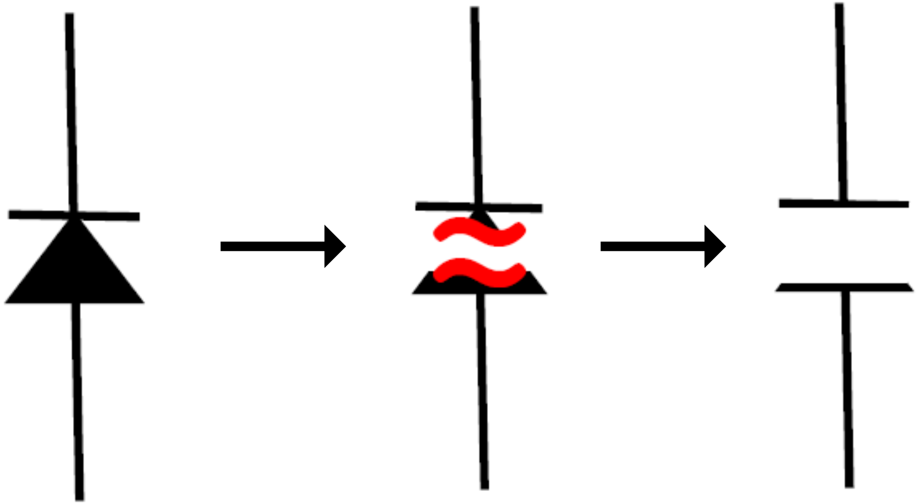
- “Network partitions should be rare but net gear continues to cause more issues than it should.” – James Hamilton, Amazon Web Services (2010)

- Microsoft LAN (2011)
  avg. 40.8 failures/day (95th %: 136)
  5 min median time to repair (up to 1wk)

- HP LAN (2012)
  67.1% of support tickets are due to the network
  Median incident duration 114-188 min

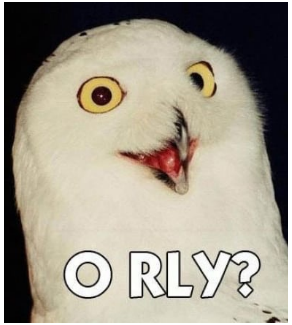
A Space Shuttle Story – STS-124

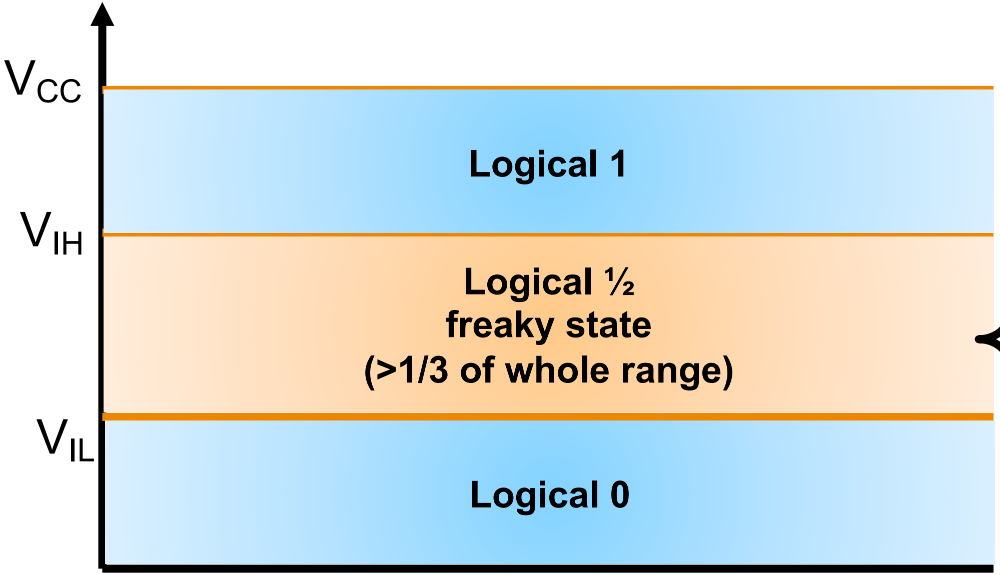




- A crack (fissure) through a diode appeared in the data bus of a Space Shuttle.
- 3-1 split of the four computers that control the Shuttle.
- +3s: split became 2-1-1.
- During troubleshooting: 1-1-1-1

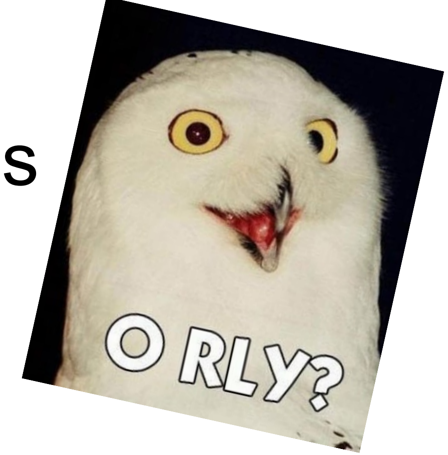
~~Analog~~ Digital systems



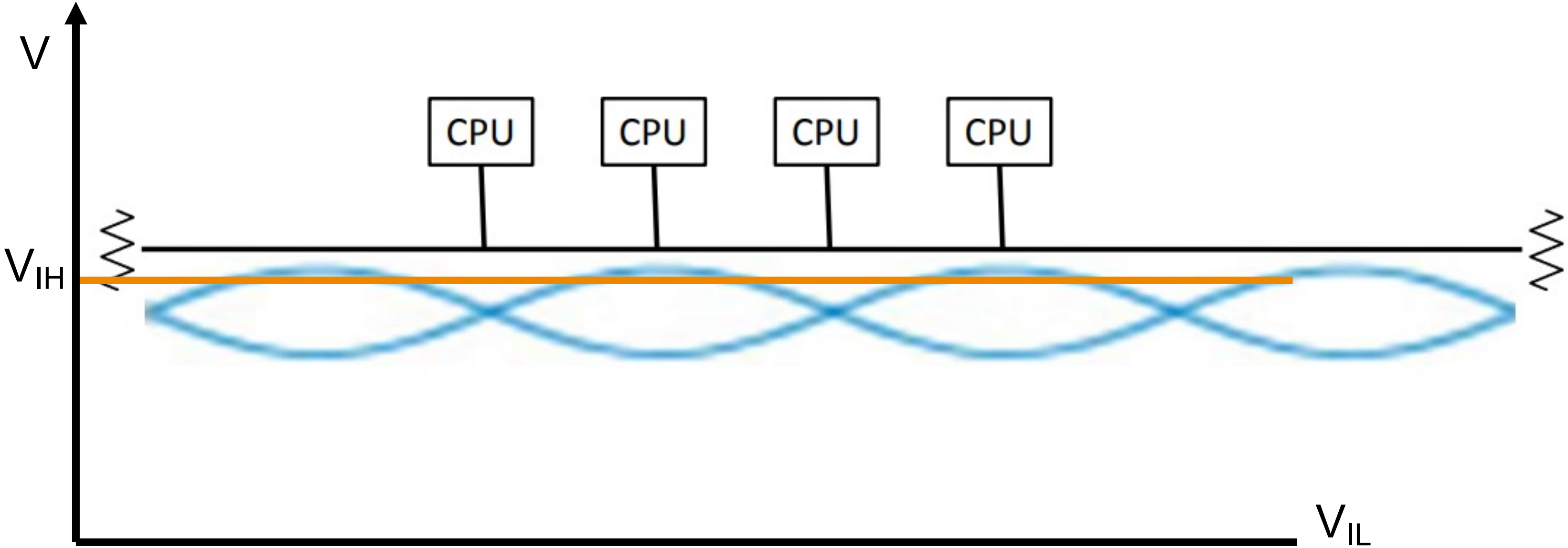


- Interpreted as 0 or 1, depending on RX threshold
- Affected by every environmental +manufacturing factor imaginable

~~Analog~~ Digital systems



Another Space Shuttle story



Failure Modes & Effect Analysis

- A whole area of engineering
- Under-used in software

Approach:

- Analyze system
- Identify what can fail, why, how
- Analyze probability, detectability
- Analyze effects of failure (cascades?)
- Mitigate !

→ Redundancy isn't enough without BFT !

→ BFT can't be solved *exclusively* in software !

BFT Consensus

Key properties:

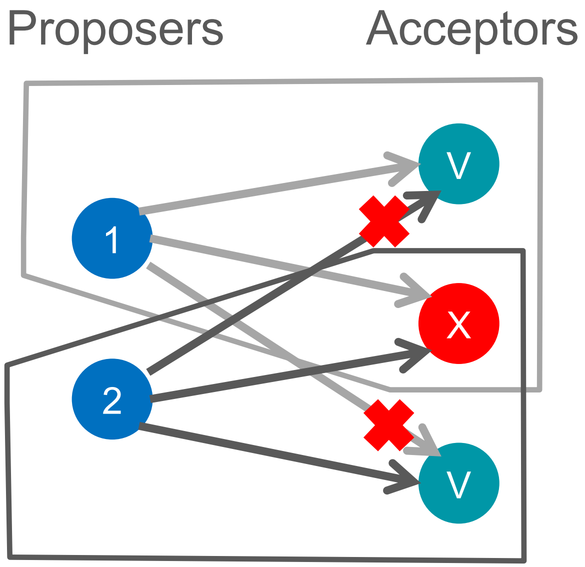
- Safety: all nodes agree on a (single) decision
- Liveness: eventually something is decided

Paxos: n >= 2f + 1

Byzantine model:

- Malicious node “X”
- Malicious network

→ n >= 2f + 1 isn’t strong enough



BFT Consensus

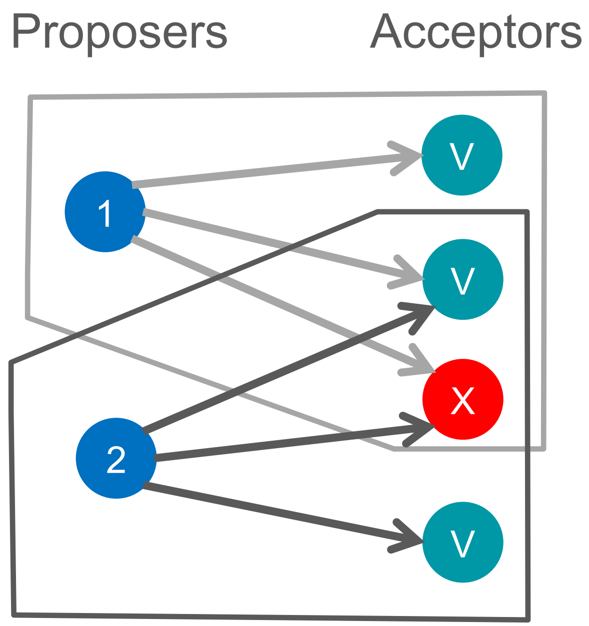
Key properties:

- Safety: all nodes agree on a (single) decision
- Liveness: eventually something is decided

BFT: n >= 3f + 1

→ No assumption about who's honest

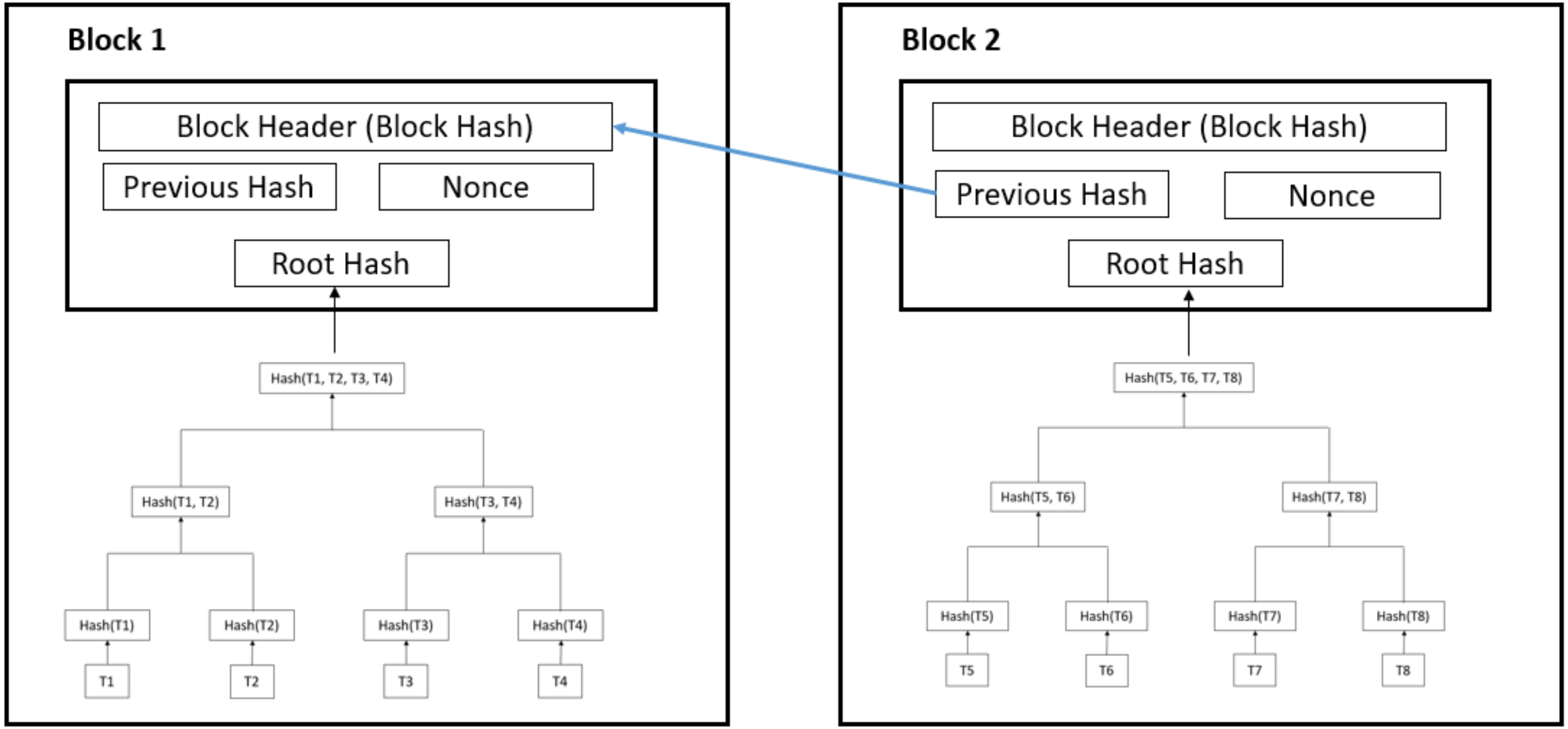
→ 1 honest node in intersection



Permissionless consensus

Bitcoin and Proof-of-Work

Blockchain structure



Bitcoin – Key ideas

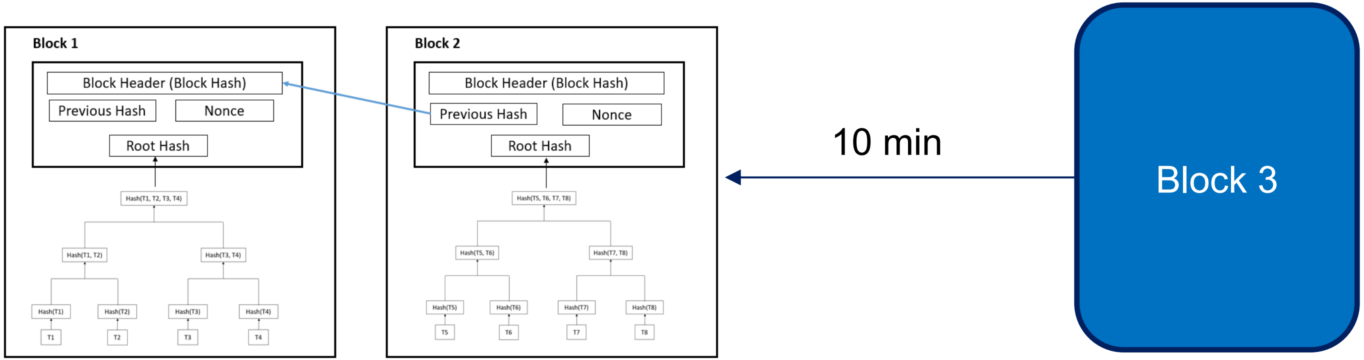
Proof-of-Work

- “Miners” solve computational puzzles (hash with leading N zeros)

    Computational power = Hash rate (H/s)

- Puzzle difficulty is adjusted to keep block rate (roughly) constant

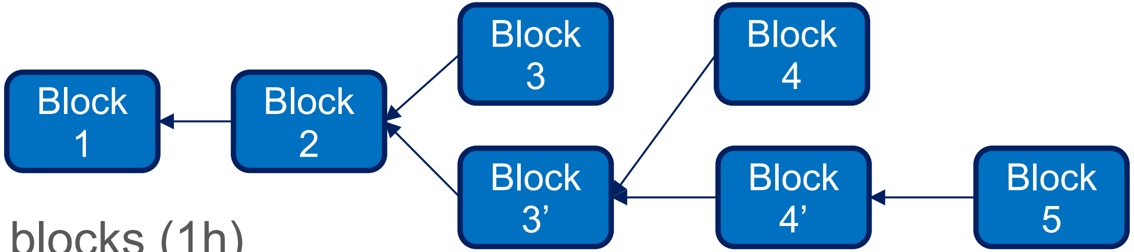
    → compensates for changes in mining power



Bitcoin – Assumptions

- Threshold assumption: majority of mining power is honest
  … independently of the number of nodes

- Longest / heaviest chain rule
  … transient safety violations (e.g. forks, reversed transactions) are OK
  … eventually forks will be resolved (based on expended work) !



- Probabilistic finality – 6 blocks (1h)
- Economic incentive compatibility

Safety ?

- Network connectivity / propagation – synchrony assumption (10 min)