MIDDLEWARE SYSTEMS
RESEARCH GROUP
MSRG.ORG

The Renaissance of Byzantine Fault Tolerance: *Revolutionizing Consensus Architecture for Pervasive Decentralized Systems*

Hans-Arno Jacobsen

Jeffrey Skoll Chair in Computer Networks and Innovation
University of Toronto

July, 9th, 2025 – BTS, Vancouver, Canada

MIDDLEWARE SYSTEMS
RESEARCH GROUP
MSRG.ORG

The Renaissance of Byzantine Fault Tolerance: *Revolutionizing Consensus Architecture for Pervasive Decentralized Systems*

Hans-Arno Jacobsen

Jeffrey Skoll Chair in Computer Networks and Innovation
University of Toronto

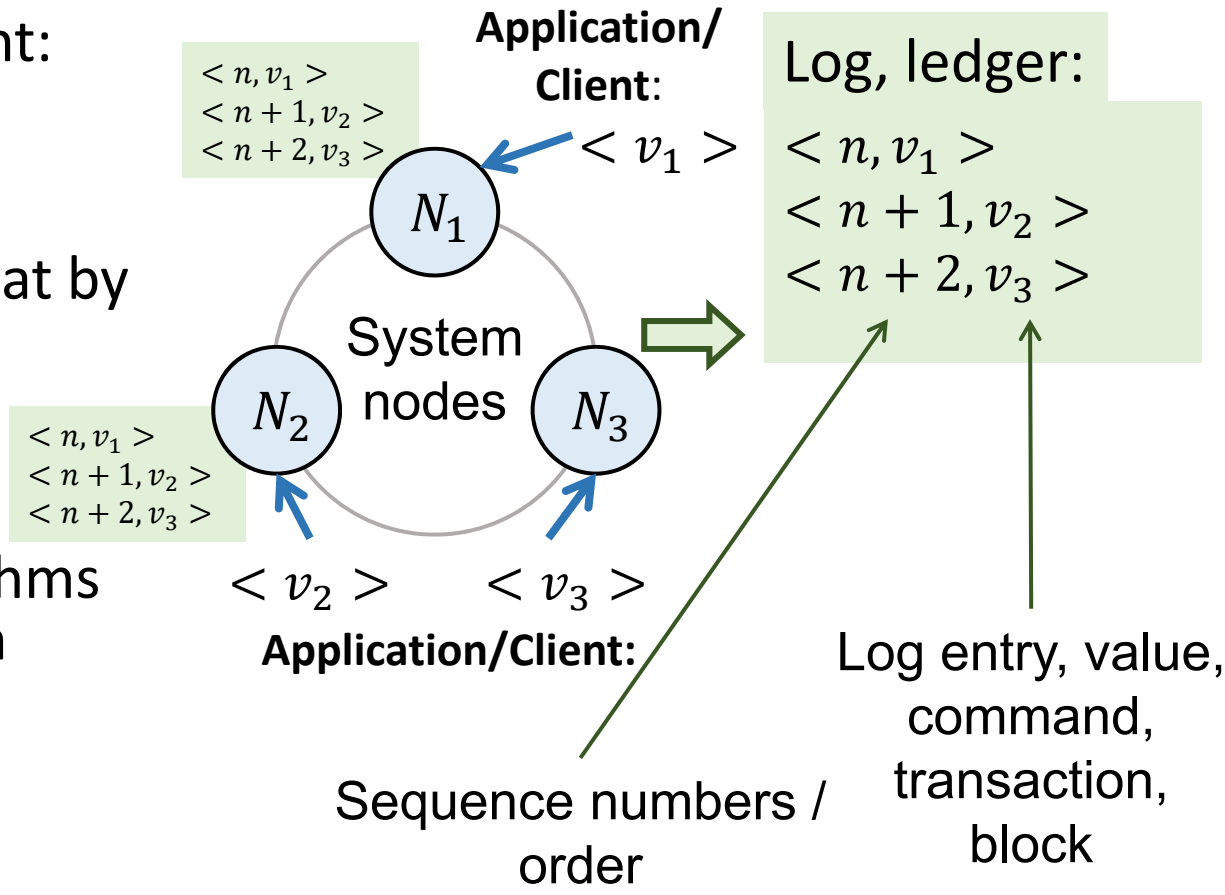
Joint work with Edward (Gengrui) Zhang,

now faculty at Concordia University

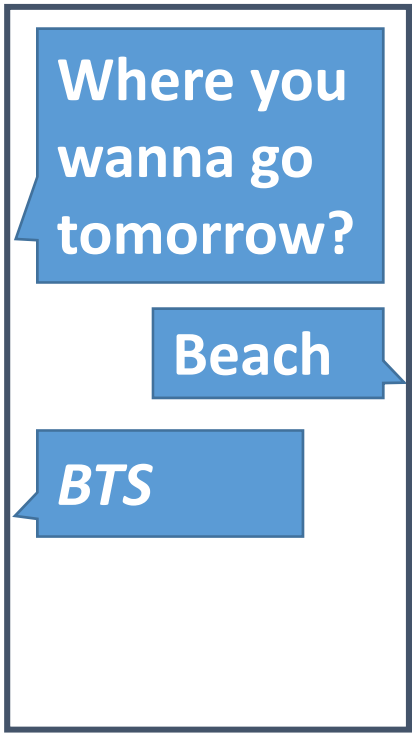
July, 9th, 2025 – BTS, Vancouver, Canada

Recall: *What's Consensus?*

- General agreement: unanimity
- Judgment arrived at by most of those concerned
- Consensus algorithms coordinate system nodes to reach agreement on application input



When and Why Is Consensus Needed?



The total order of events is not maintained

No consensus on order, but *we live with it*.

When and Why Is Consensus Needed?

Agree On …

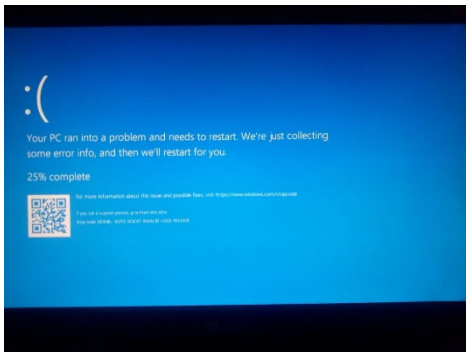
- Order of operations, events
- Content of replicated log
- Determine leader
- Blocks, transactions written to distributed ledger (DL)

Is that all?

Your PC ran into a problem that it couldn't handle, and now it needs to restart.

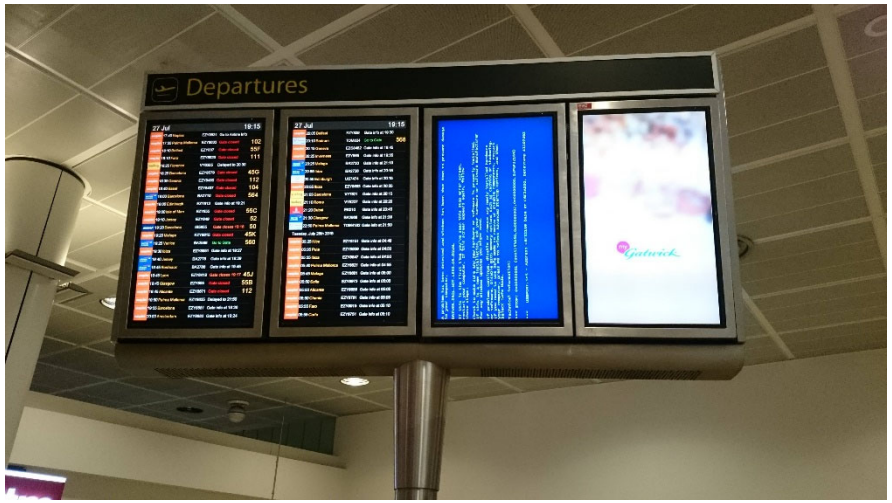
Fault Tolerance

- Systems (nodes) need to tolerate failures
- Meaning: **Function correctly when (some) failures occur**
- *What kind of failures?*



System Failures I

What kind of failures?



CLOUD INFRASTRUCTURE

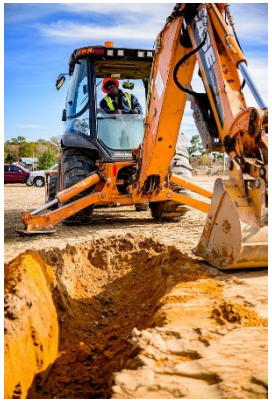
The great 2020 Gmail outage: A tale of two blackouts, and lessons learned

LYNN GREINER DECEMBER 21, 2020

System Failures II

What kind of failures?

- The company said it began investigating "***increased error rates and latencies***"
- Noticed a ***decrease in ads served to west coast***
- Periodic, reoccurring communication disruptions
- "…confirmed that a back hoe, a large excavating machine, ***cut the fiber-optic cable*** in…"
- "A backhoe a real cyber threat"



System "*Failures*" III

What kind of failures?

Ronin Bridge Exploit: The biggest crypto hack

Loss. The attacker got away with 173,600 ETH and 25,500,000 USDC. The loss stood at **$624** million as of March 23, 2022.

Cause. The crypto hack unfolded as a compromise of validator nodes. In particular, the hacker took control over four Sky Mavis and one Axie DAO validators, enough to constitute a 5 of 9 majority. Sky Mavis' validators were

Poly Network: Cross-chain message error

Loss. On August 10, 2021, the cross-chain platform incurred a loss of **$611 million** in various tokens on three different chains, but the attacker returned the stolen funds.

Cause. The loss was triggered by insufficient access control linked to a smart contract vulnerability. The DeFi protocol had a critical error in one of its functions responsible for cross-chain messages. Because of that, anyone was able to trigger

Wormhole: Open the Floodgates of Cryptocurrency Hacking

Damage. The hacker minted 120,000 wETH with no underlying ETH. The financial setback totals **$320 million** lost in Solana's bridge on February 2, 2022.

Cause. The root cause of the exploit was traced back to Solana VAA verification resulting in a failure to validate "guardian" accounts. There were unpatched Rust smart contracts in Solana that prolific cryptocurrency hackers manipulated into deposit credit.

Design issues with protocols

- Validators being sidelined from validator set (protocol flaw)
- Cf. "*Analyzing Geospatial Distribution in Blockchains*," Shashank Motepalli et al. DAPPS04

Quick Online Quiz About Failures

Either scan,

Quick quiz for you

or go to

https://menti.com

Enter the code to join:

8348 5316

Summary of Our Quiz

For Those Who Don't Take It ☹

- Conceptual failure categorization

- What kind of failure …

  - *"Blue Screen of Death"*
  - *Severing a communication line*
  - *Signal attenuation*
  - *Hunting "accident"*
  - *Crypto hacks*

Agenda

- Benign vs. Byzantine failures
- Active view change in BFT
  - Via node reputation
  - **PrestigeBFT**
- Dynamic membership change in BFT
  - **V-Guard**

Benign vs. Byzantine Failures

Benign failures

CFT consensus

- **Failures:** server crash, message loss, reordering, duplication
- Applications (everything distributed):
  - File systems: HDFS and GFS
  - Databases: Spanner and etcd
  - Leader election/coordination: Chubby and Zookeeper

Byzantine failures

BFT consensus

- **Failures**: arbitrary behavior; faulty nodes (intentional, unintentional), node could lie, send erroneous messages (i.e., not follow protocol, bugs)
- Applications:
  - Safety critical systems: command and control systems, airplanes, etc.
  - DLs: Diem, CCF, etc.

Benign vs. ~~Byzantine~~ Arbitrary Failures

Benign failures

CFT consensus

- **Failures:** server crash, message loss, reordering, duplication
- Applications (everything distributed):
  - File systems: HDFS and GFS
  - Databases: Spanner and etcd
  - Leader election/coordination: Chubby and Zookeeper

~~Byzantine~~ Arbitrary failures

BFT consensus

- **Failures**: arbitrary behavior; faulty nodes (intentional, unintentional), node could lie, send erroneous messages (i.e., not follow protocol, bugs)
- Applications:
  - Safety critical systems: command and control systems, airplanes, etc.
  - DLs: Diem, CCF, etc.

Benign vs. Arbitrary Failures

Benign failures

CFT consensus

- **Failures:** server crash, message loss, reordering, duplication
- Applications (everything distributed):
  - File systems: HDFS and GFS
  - Databases: Spanner and etcd
  - Leader election/coordination: Chubby and Zookeeper

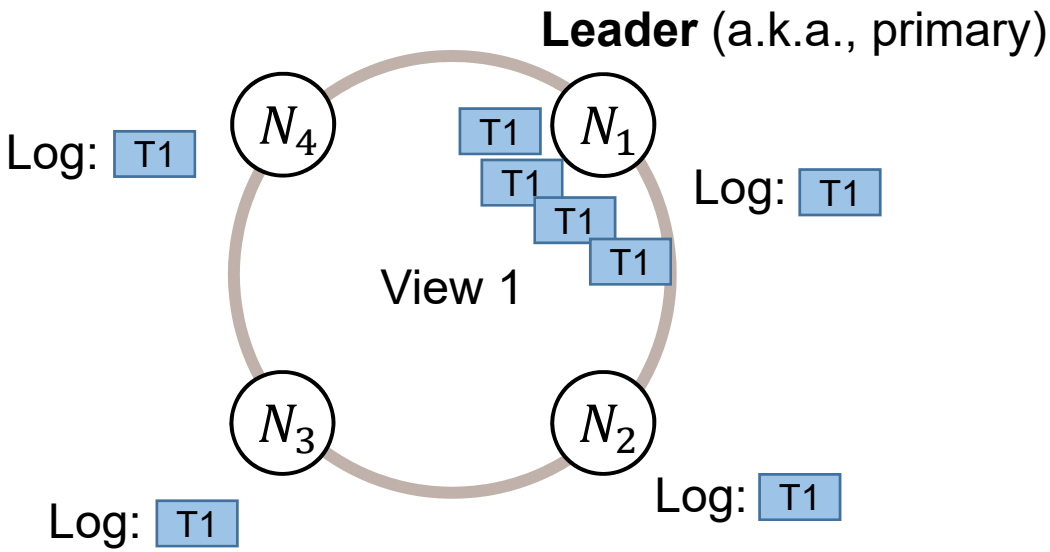
Arbitrary failures

BFT consensus

- **Failures**: arbitrary behavior; faulty nodes (intentional, unintentional), node could lie, send erroneous messages (i.e., not follow protocol, bugs)
- Applications:
  - Safety critical systems: command and control systems, airplanes, etc.
  - DLs: Diem, CCF, etc.

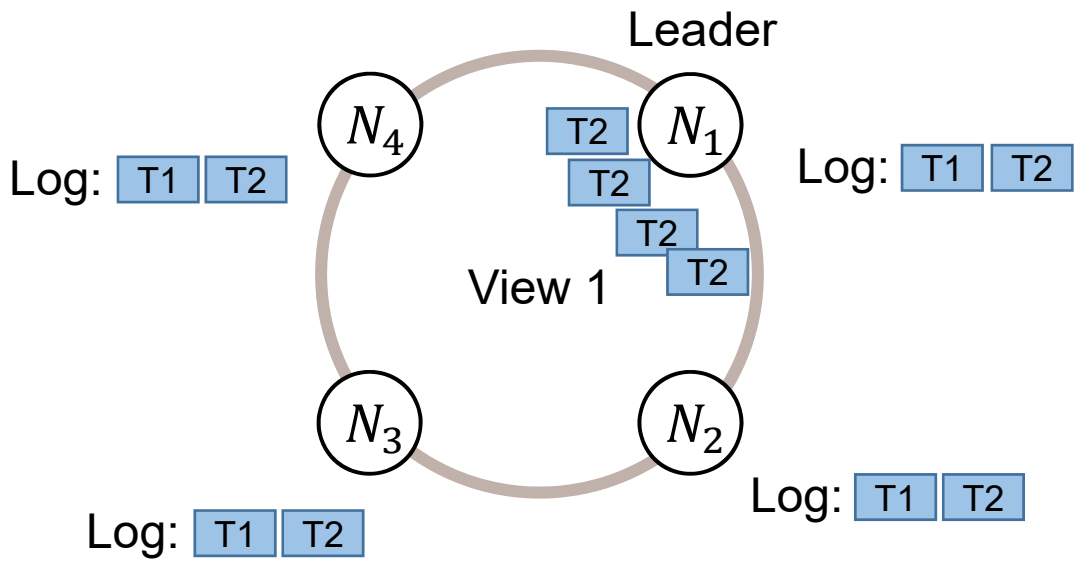
Leader-based BFT Consensus

Replication (Consensus on Committing Client Requests)



Leader-based BFT Consensus

Replication (Consensus on Committing Client Requests)

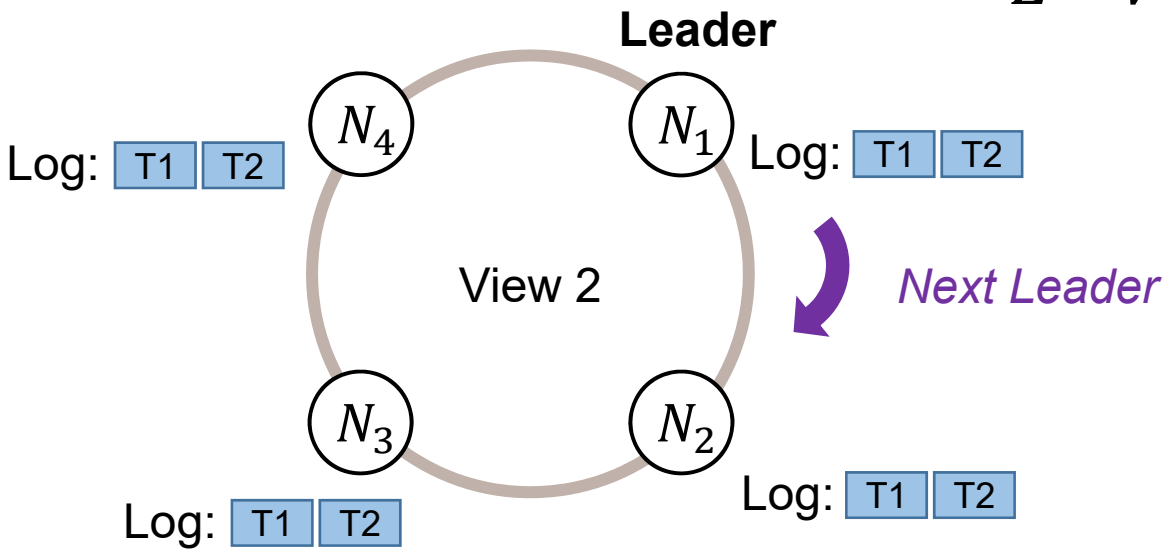


Leader-based BFT Consensus

Passive View Change (Consensus on Leader)

Leadership rotation:

$$L = V \ mod \ n$$

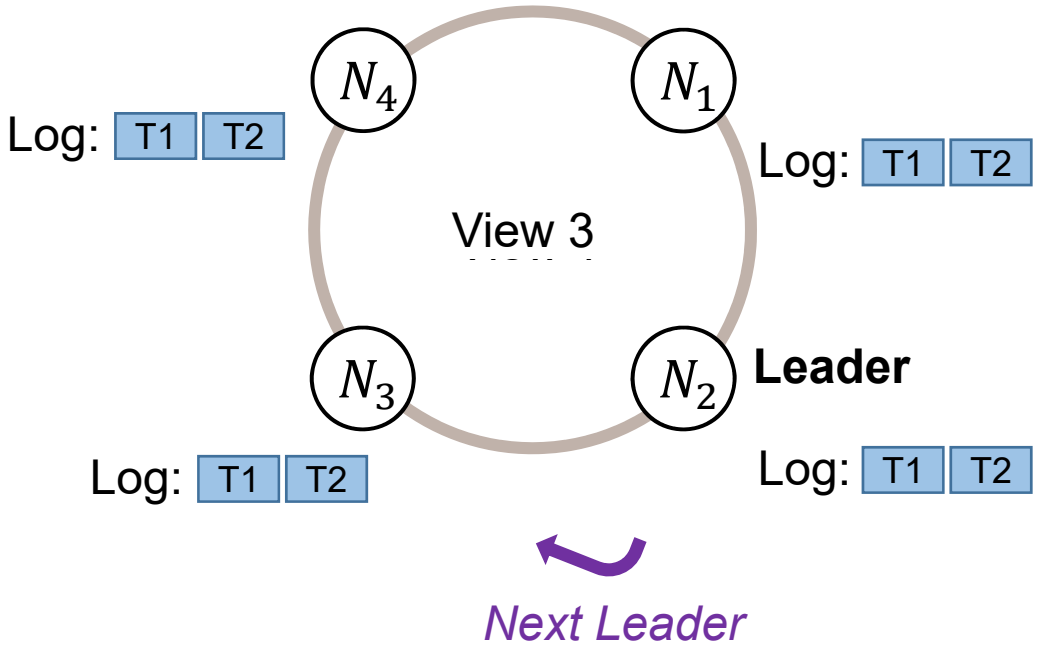


Example:

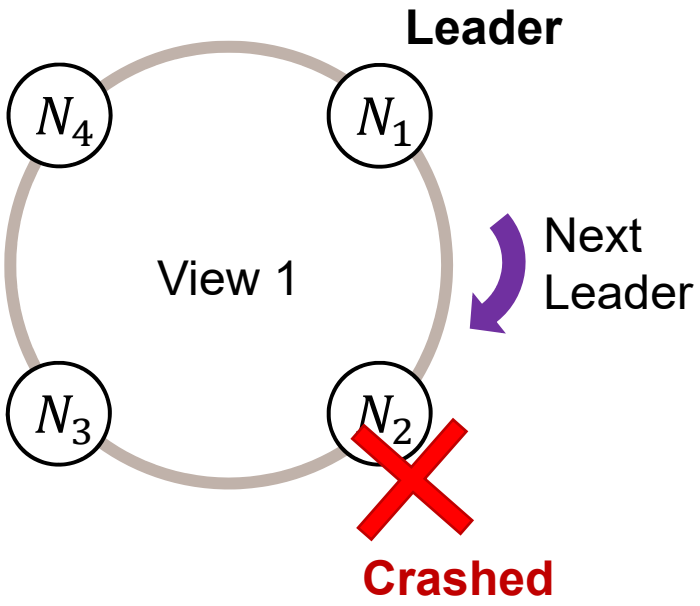
- L = 1 mod 4 = 1 (in View 1)
- **L = 2 mod 4 = 2 (in View 2)**
- L = 3 mod 4 = 3 (in View 3)

Leader-based BFT Consensus

Passive View Change (Consensus on Leader)



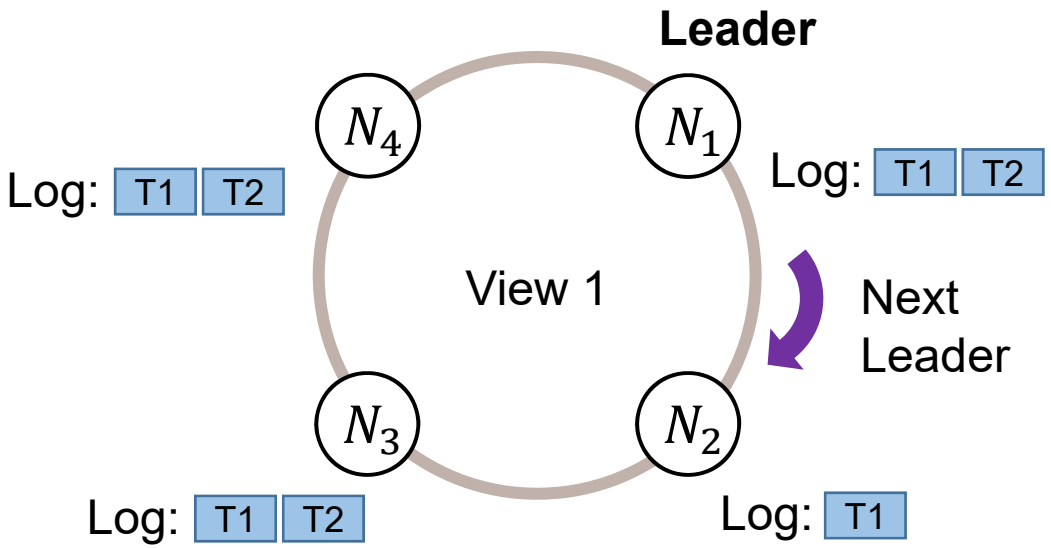
What's wrong with passive view change?



Wait for a timeout to realize N_2 is down!

Worst case: Wait for *f-1* timeouts (after initial failure detection)

What's wrong with passive view change?



N_2's log lags behind other nodes

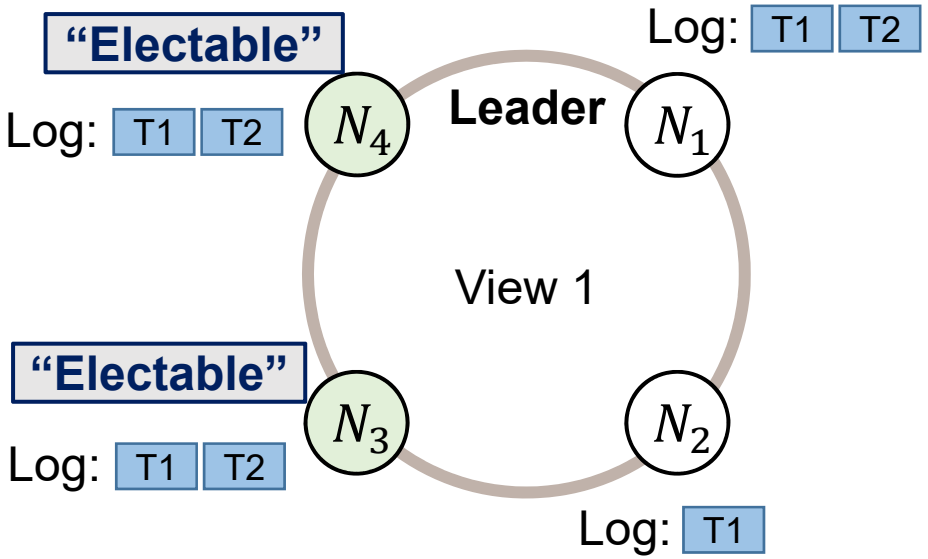
Further increase in latency!

Must synchronize log from others to be up-to-date first.

Goal: Enable Active View Changes in BFT

Nodes Actively Campaign for Leadership

- No leader schedule



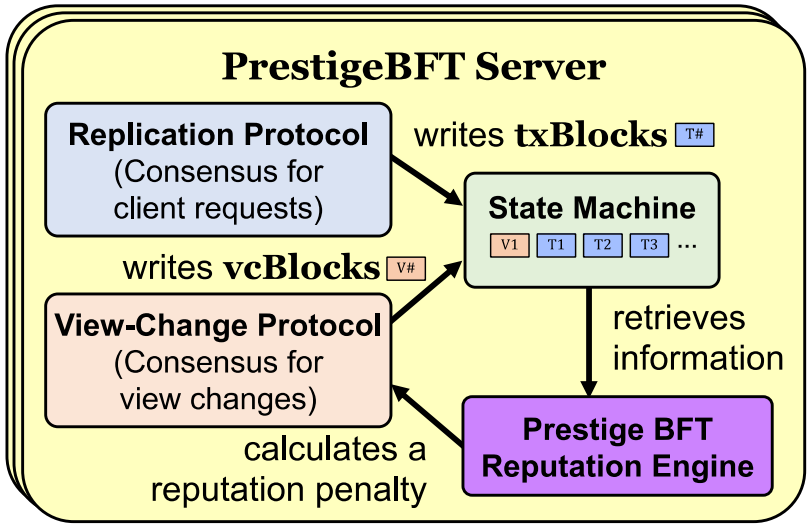
- Nodes detecting leader failure, start election
- Only elect nodes that are up-to-date
- Done under CFT (Raft)
- More efficient but not sufficient!
  - Byzantine servers could lie and start election campaigns

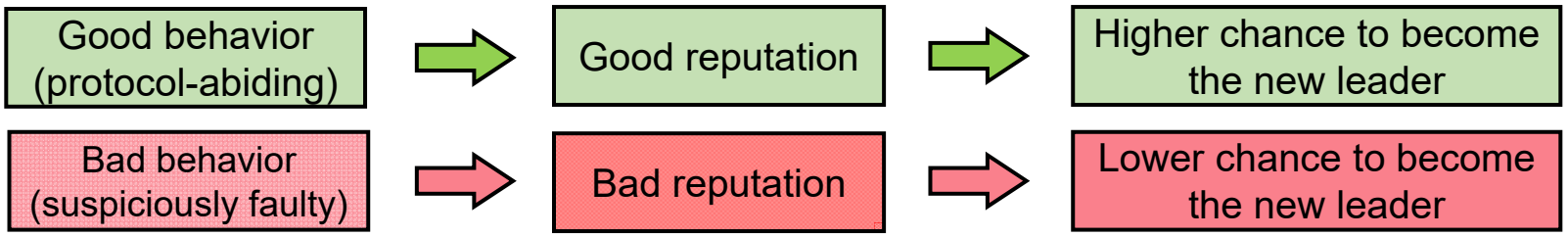
How can we reduce the chance that Byzantine servers become leaders?

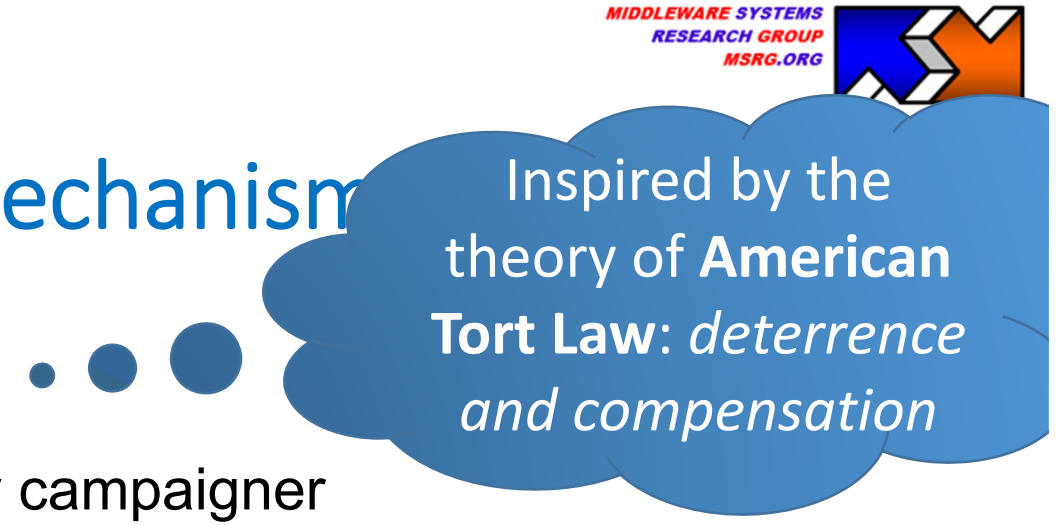
PrestigeBFT: A Reputation-based Approach

Enabling Active View Change in BFT

- Reputation mechanism translates node behavior into a **reputation score** that reflects node's chance of being correct
- Likelihood of node being selected as new leader







Our Reputation Mechanism

Inspired by the theory of **American Tort Law**: *deterrence and compensation*

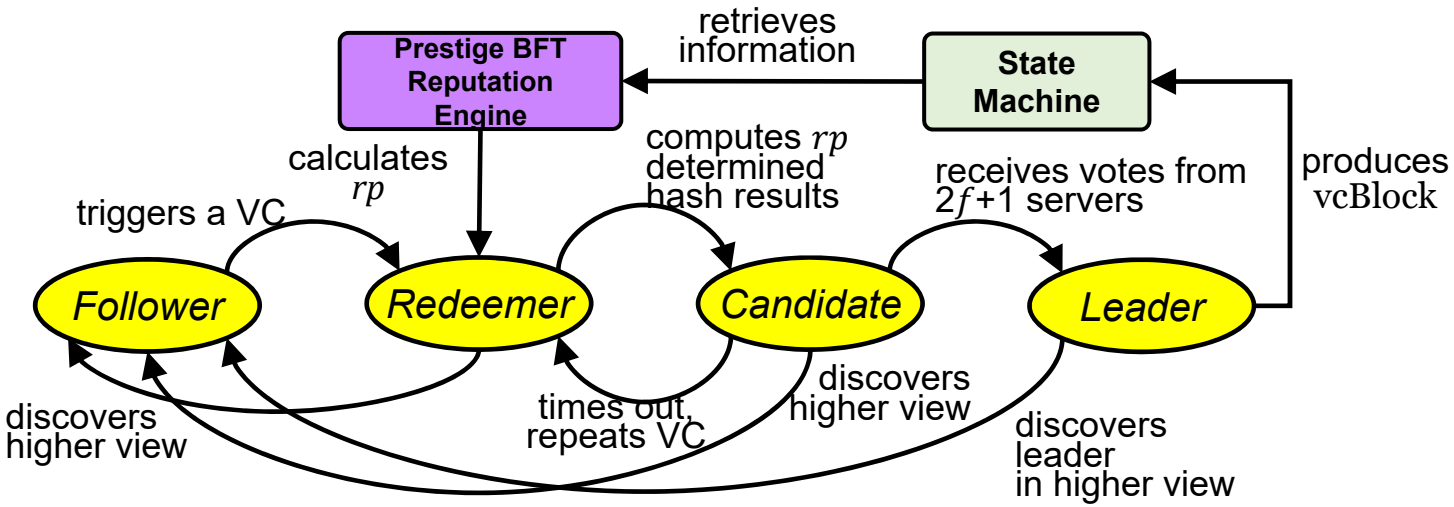
- Step 1 **Penalization**
  - Apply penalty to every campaigner
- Step 2 **Compensation**
  - Reward up-to-date log replication
    - Calculate incremental log responsiveness (δ_{tx})
  - Reward no or only gradual increase in past penalties
    - Calculate the Z-score of all penalties in past view changes (δ_{vc})
  - Deduct penalty applied in Step 1

Entice servers to replicate more transactions

Incentivize servers to stop repossessing leadership

Active View Change Protocol

State and Transitions Nodes Trigger as Leader Elections Unfold



As compared to passive view changes

- Not affected by crash failures or slow servers
- Gradually distinguish Byzantine servers from correct servers
- Suppress faulty servers over time by imposing computational work

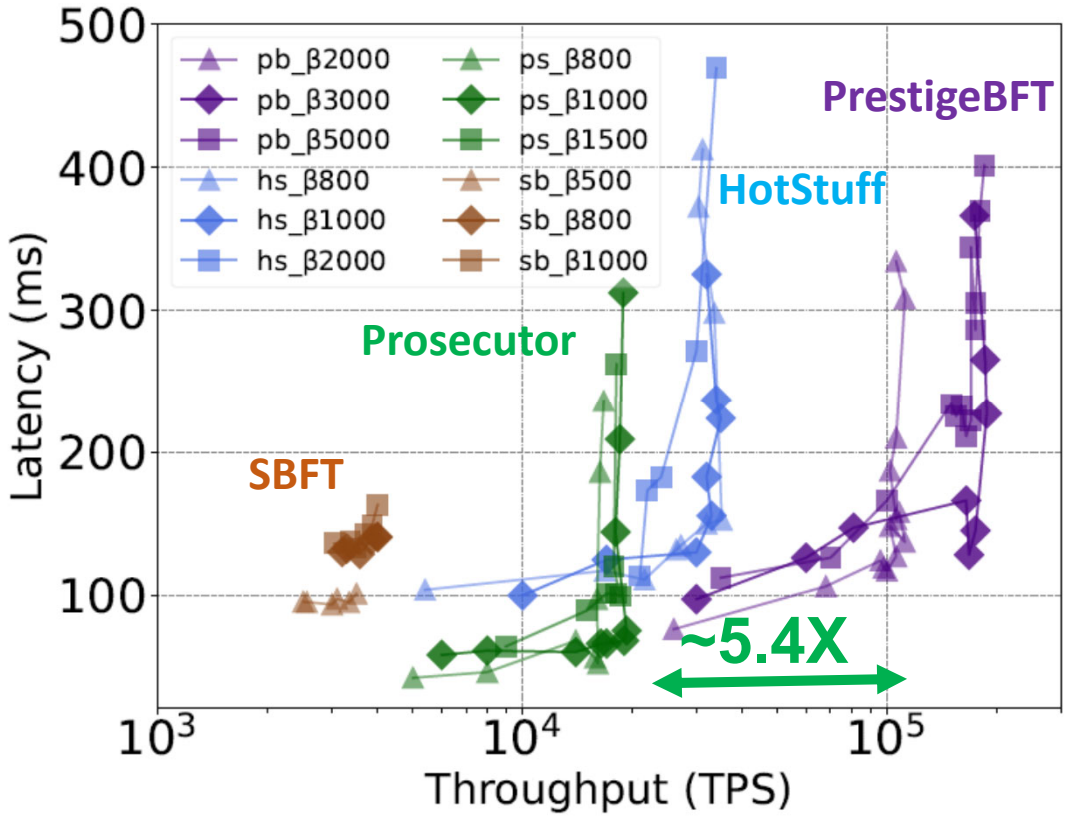
Performance Under Normal Operation I

No Failures – Determine Peak Performance

Compute Canada Cloud over $4, 16, 31, 61, 100$ VMs

Increase TPS until latency shoots up (transactions queued)

Different batch sizes (β)

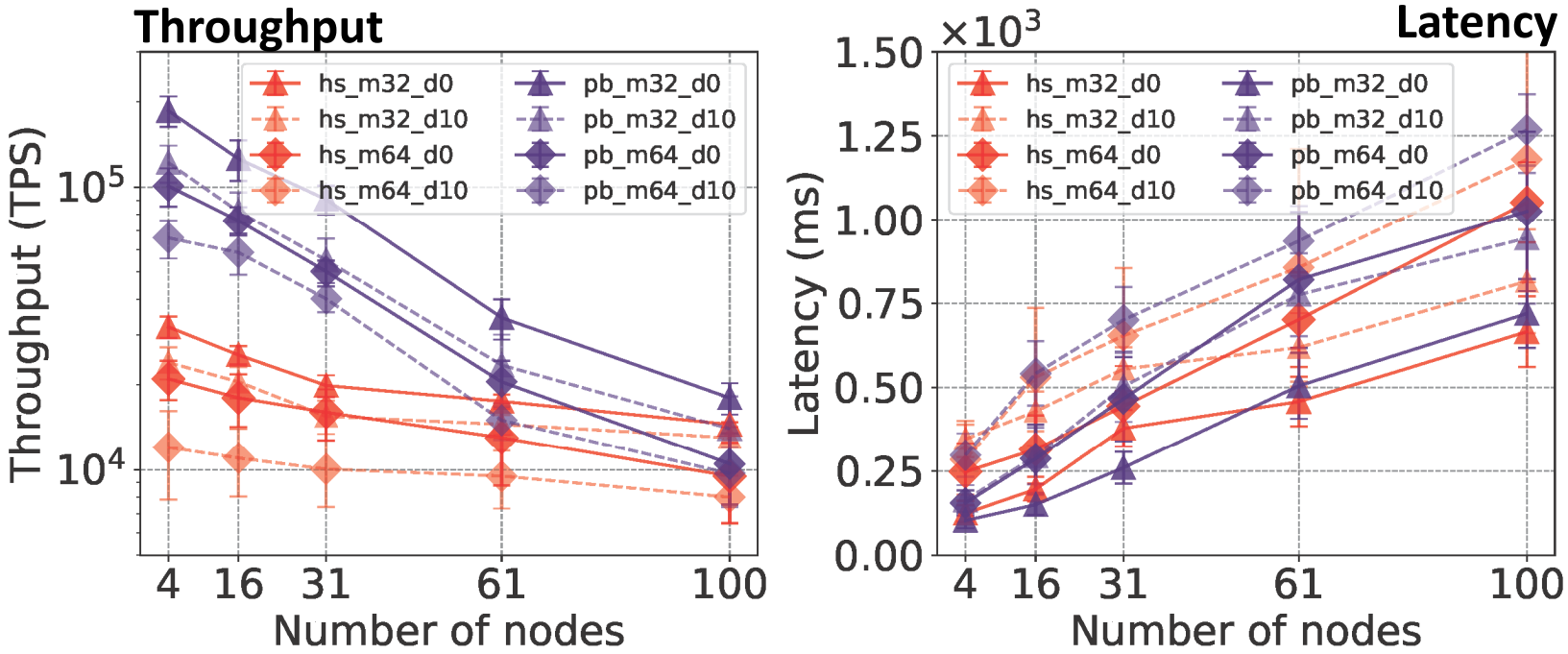


Three baselines: SBFT (sb), HotStuff (hs), Prosecutor (pr)

Performance Under Normal Operation II

No Failures – Impact of Nb. Of Nodes on Throughput/Latency

Across different message sizes and inter-node delays.

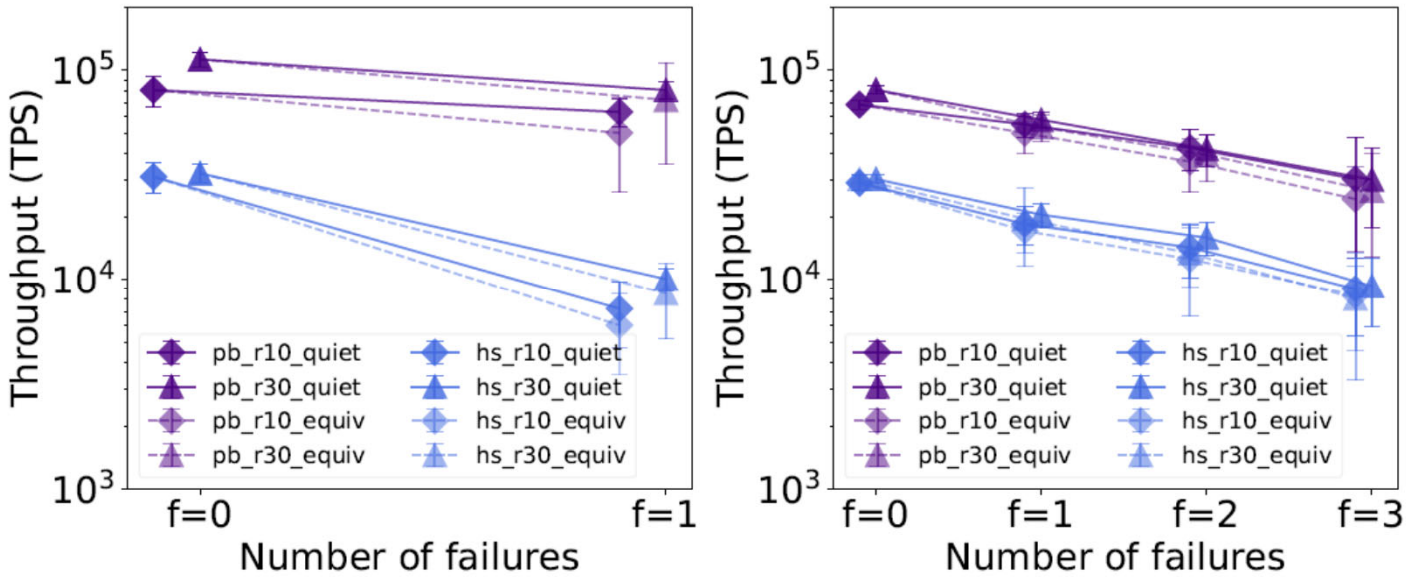


Performance Under Failures

Attack scenarios (failures):

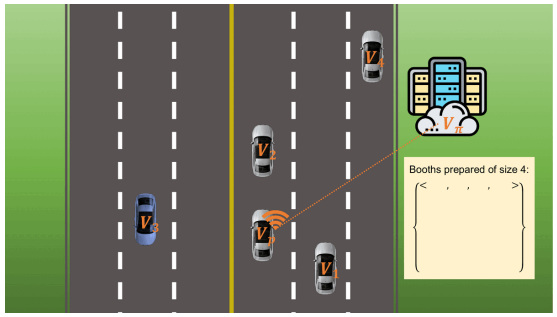
- **Quiet participants**: Do not respond to any request (similar to crash failures)
- **Equivocation**: Reply to query by sending back erroneous messages
- **Repeated view-change attacks**: Campaign for leadership when not leader

Throughput **Throughput**



PrestigeBFT - Summary

- First **active view change protocol** for BFT consensus
  - Servers can proactively campaign for leadership, preventing unavailable or slow leaders
- Convert node's behavior history into a **reputation score** representing node's **likelihood of being correct**
- Unique combination of **robustness** and **efficiency**
  - Faulty servers are suppressed and cannot usurp leadership **after they perform attacks that relegate their reputation**



V-Guard

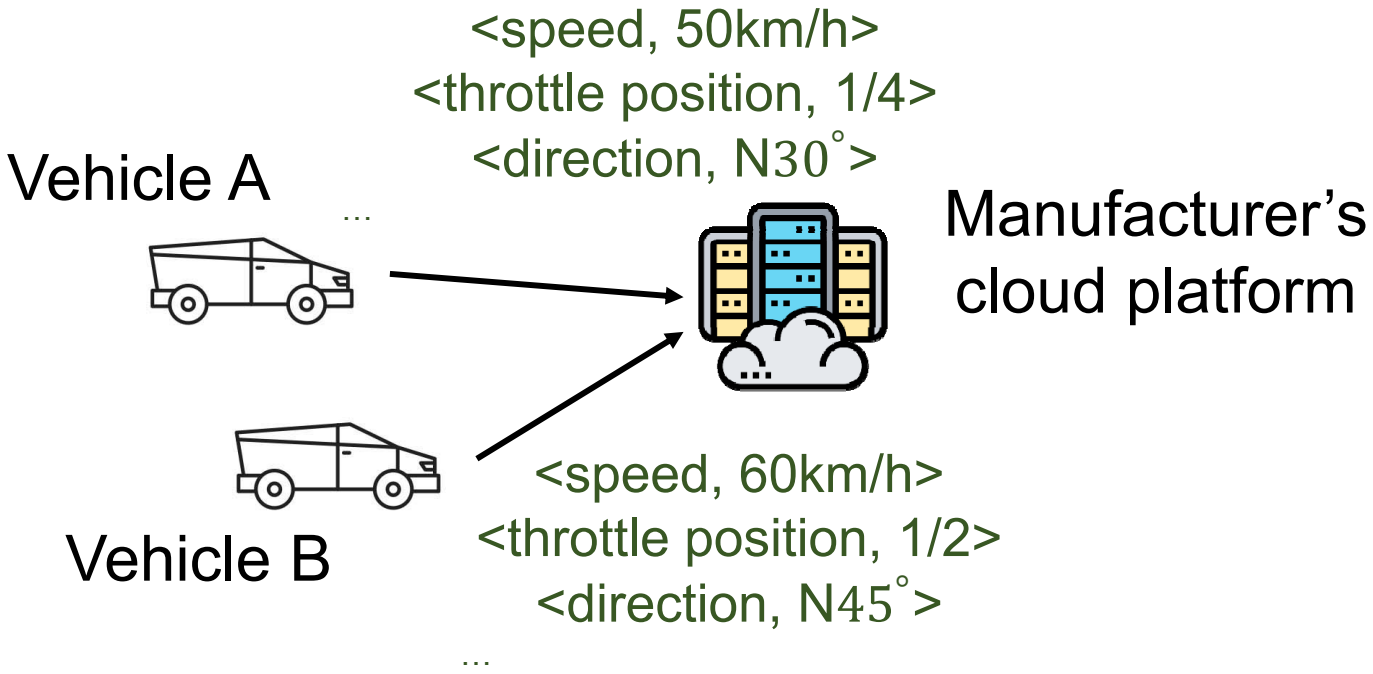
How about dynamic membership?

Nodes coming and going

But, why?

Dawn of Vehicular Automation

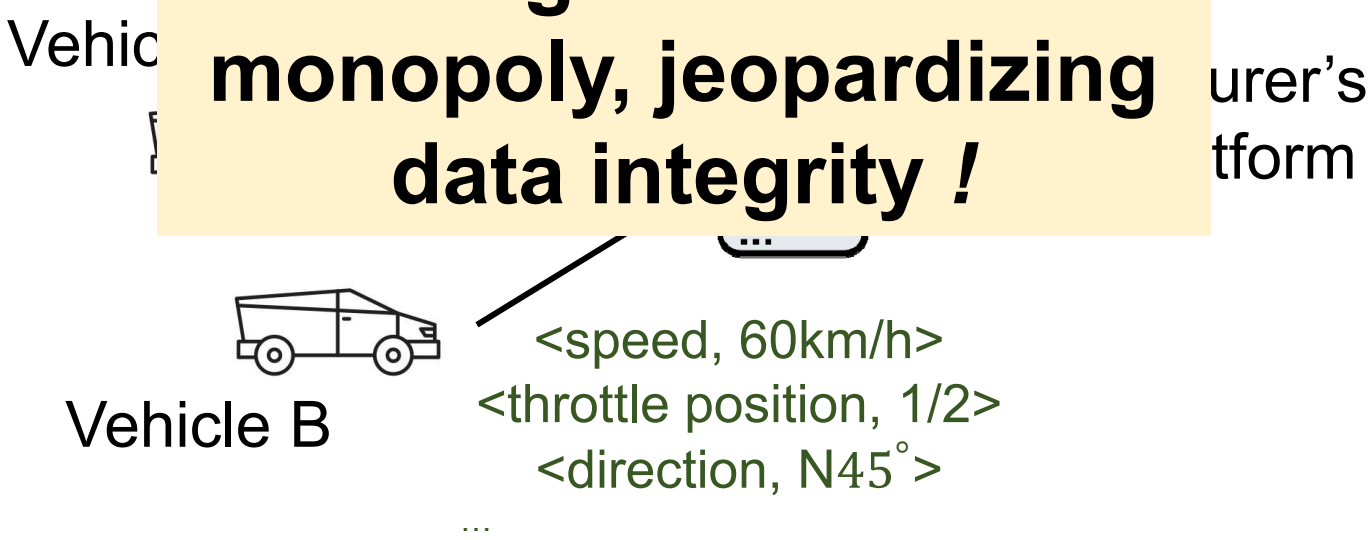
- Vehicle manufacturers' Autopilot, SuperCruise
  - Data collected by manufacturer only – centralized model
  - **Consumers have only limited access to their own data**



Dawn of Vehicular Automation

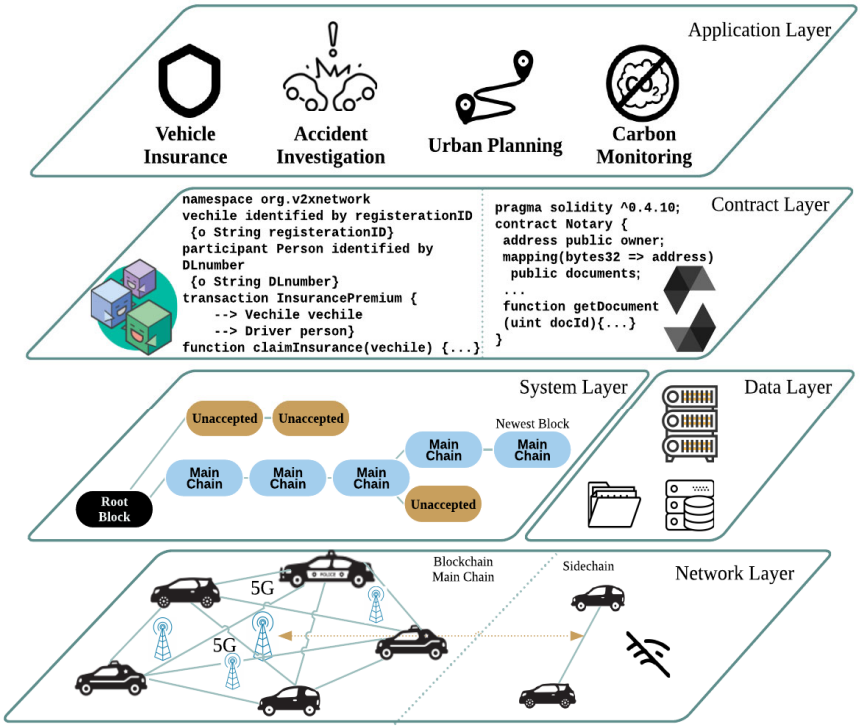
- Vehicle manufacturers' Autopilot, SuperCruise
  - Data collected by manufacturer only – centralized model
  - **Consumers have only limited access to their own data**

Centralized data handling creates a **data monopoly, jeopardizing data integrity *!***



Busting Data Monopolies via DLs

- V2X demands for DL-centric designs
- **Data integrity** via **consensus**
- Vehicles inherently identifiable, thus, **permissioned DLs**
- **Many DL initiatives by car manufacturers**



NB: *V2X* – "Vehicle to ***Everything***" – mostly in terms of communication.

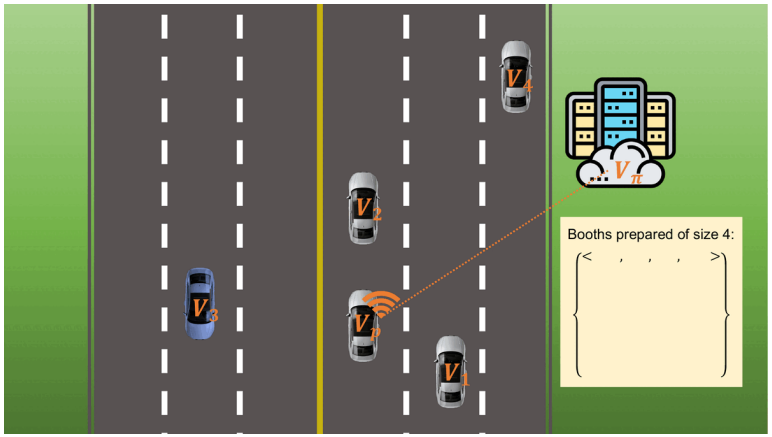
Why a New DL Design?

- BFT consensus algorithms assume a **stable environment**
- I.e., an a priori fixed, static set of nodes
- V2X networks **cannot guarantee a stable environment** – vehicles come/go, are online/offline

V2X DLs must be able to operate in a **dynamic environment**, - frequently changing memberships

Binding Configurations and Transactions

- A **booth** is a **descriptor** that designates a set of membership configurations of participating vehicles.

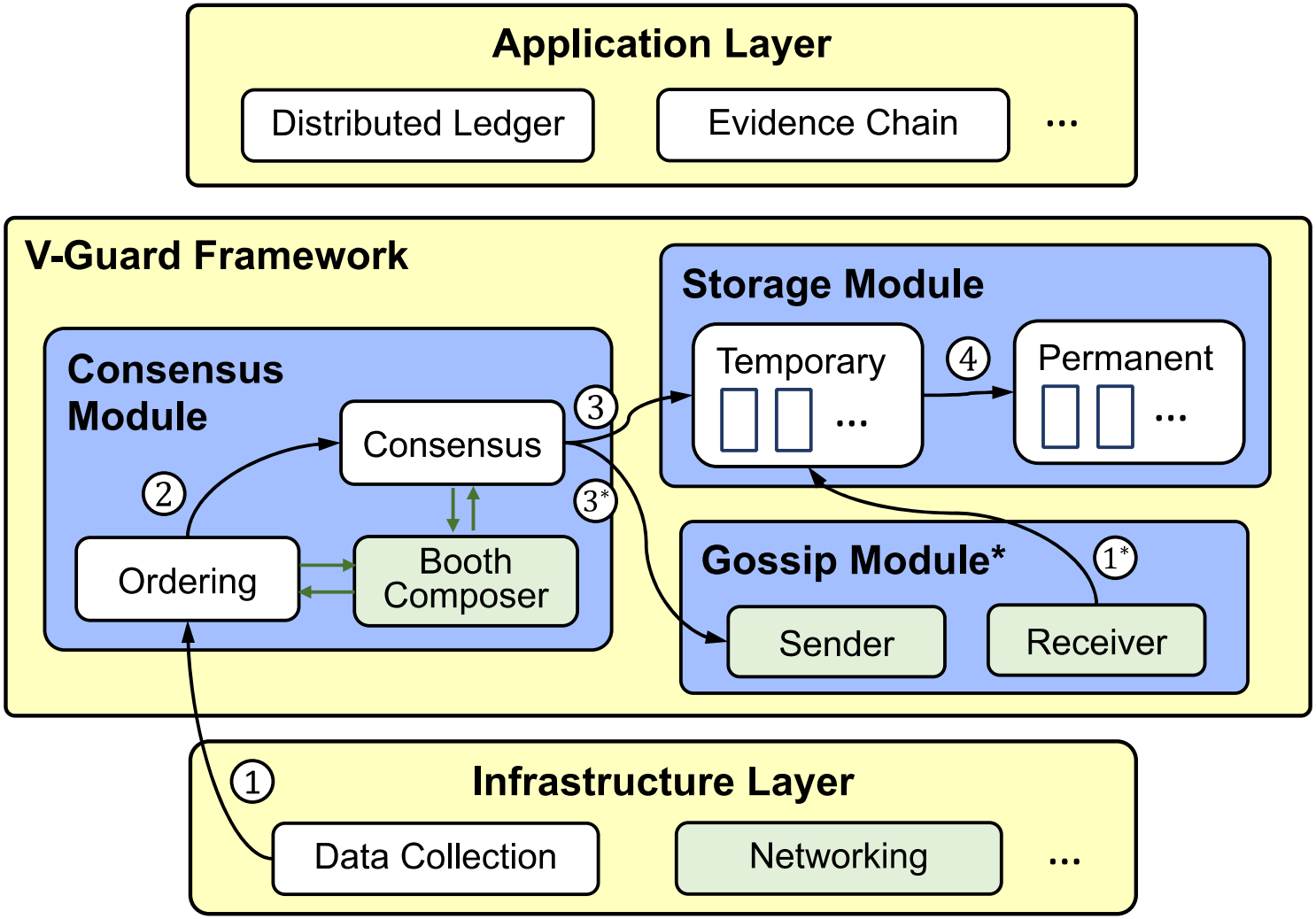


Booth composer
Preparing a queue of valid booths

Consensus target
$<t_1, B_1, O_1, Q_1>$

Transaction
Order
Booth
Quorum

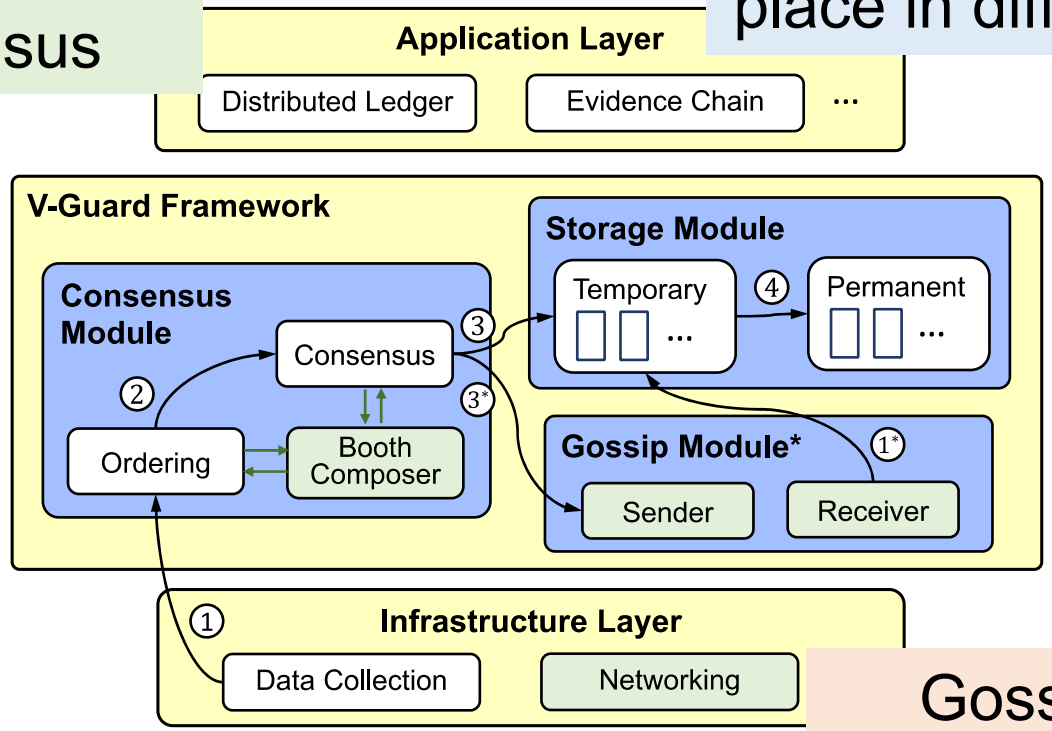
V-Guard Architecture



V-Guard Architecture

Ordering separated from consensus

Ordering and consensus can take place in different booths



Gossip to further disseminate committed transactions

Peak Performance Stationary

Ordering, Consensus

	Best batch size # of transactions	Throughput (TPS)	Latency (ms)
HotStuff	(β_b=1000)	34,015	155.39
ResilientDB	(β_b=500)	80,580	4367.87
Narwhal	(β_b=1,000,000*)	423,058	510.6
V-Guard (o)	(β_b=3000)	871,248	13.1
(c)		765,930	143.72

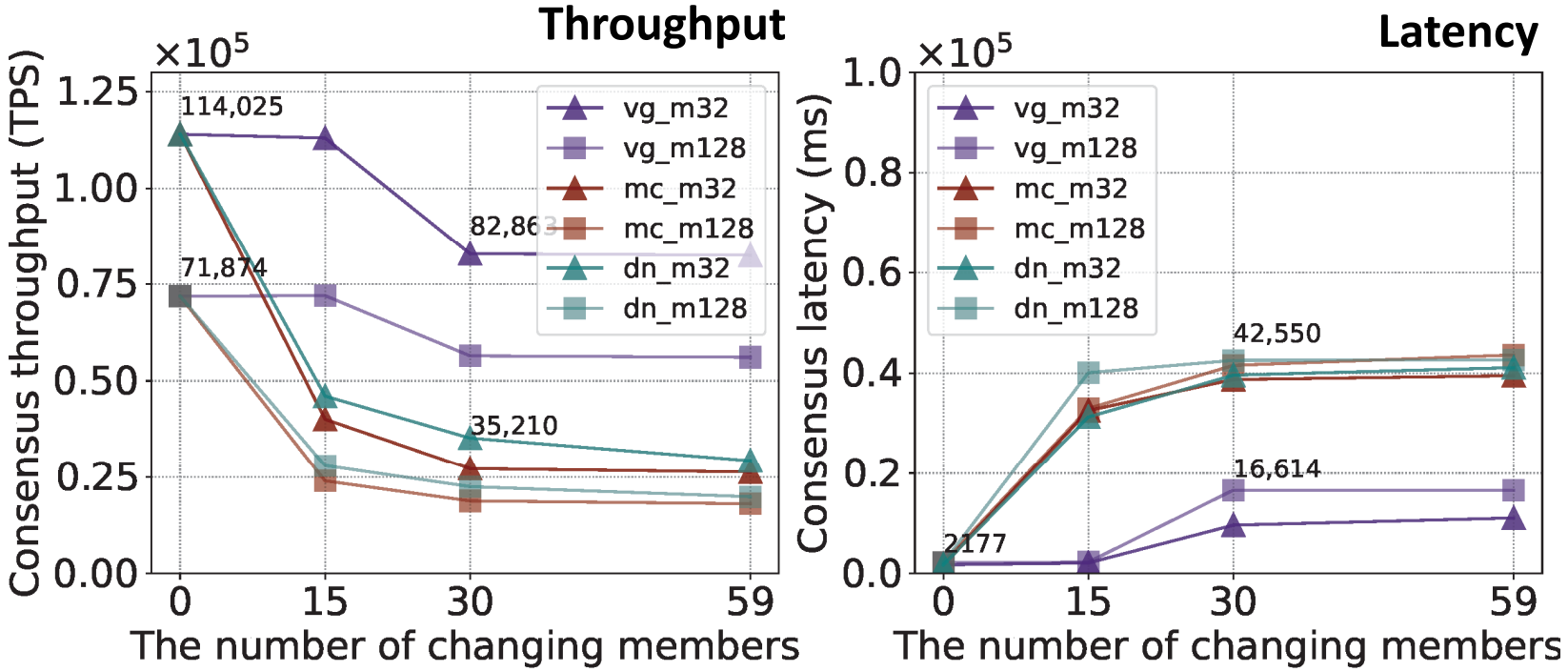
* Narwhal uses fixed-bytes buffers as batches, so its buffer size = $m \times \beta_b$.

V-Guard's peak throughput is

- **22** × higher than HotStuff [PODC`19]
- **9.5** × higher than ResilientDB [VLDB`21]
- **1.8** × higher than Narwhal [Eurosys`22]

Throughput/Latency under Dynamicity

Consensus



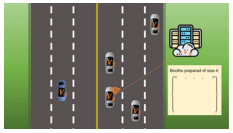
V-Guard - Summary

- V-Guard is **fast**
  - Separates ordering from consensus
- V-Guard is **dynamic**
  - Operates under dynamically changing memberships
- V-Guard is **versatile**
  - Suitable to applications operating under unstable networking, i.e., intermittent connectivity

Wanna Know More?

Edward

- **PrestigeBFT** (IEEE ICDE'2024)
  - https://arxiv.org/abs/2307.08154
- **V-Guard**
  - https://github.com/vguardbc/vguardbft/
  - https://arxiv.org/abs/2301.06210



- "***Reaching Consensus in the Byzantine Empire**: A Comprehensive Review of BFT Consensus Algorithms*" (ACM CSUR)
  - https://arxiv.org/abs/2204.03181
- "**Cabinet**: *Dynamically Weighted Consensus Made Fast*"
  - https://www.vldb.org/pvldb/vol18/p1439-zhang.pdf

Appendix

- Correctness argument PrestigeBFT
- Cf. https://arxiv.org/abs/2307.08154, Section 5 for proof sketches
  - Theorem 1 (Validity). In each consensus instance, if all servers have received the same tx, then any tx committed by a non-faulty server must be that common tx.
  - Theorem 2 (Liveness). After GST, a non-faulty server eventually commits a proposed client request.
  - Theorem 3 (Safety). Non-faulty servers do not decide on conflicting blocks. That is, non-faulty servers do not commit two txBlocks at the same sequence number n.
- Correctness argument V-Guard
  - Cf. https://arxiv.org/abs/2301.06210, Section 4.4

Audience Response (BTS)

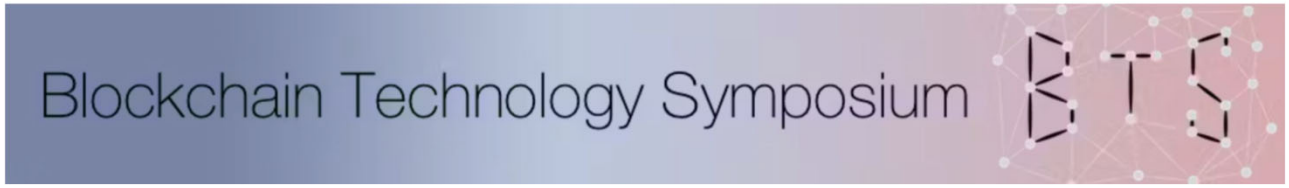
Failure Quiz

Menti Results, BTS 2025

Mentimeter

Blockchain Technology Symposium: Audience Participation Quiz

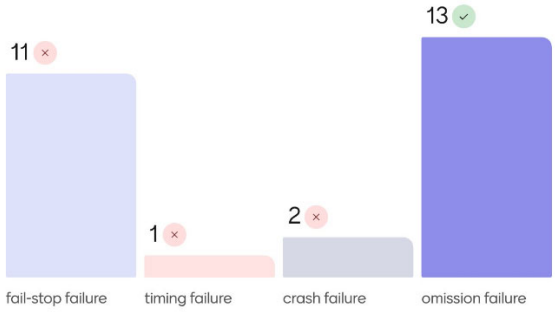
Six Quick Questions About Failures



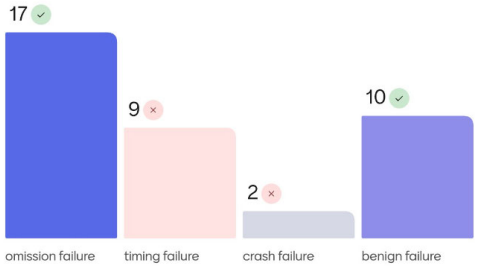
Conceptually speaking, what kind of failures do you know?

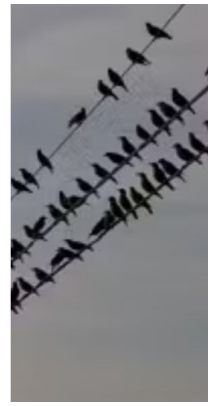
The "*blue screen of death*" is a (an)

Severing a communication line is a (an)

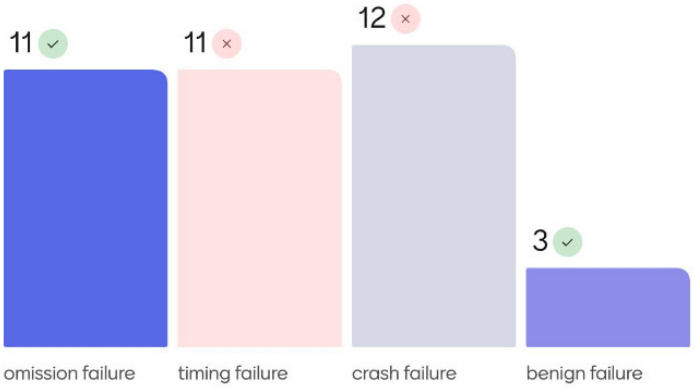


Signal attenuation is a (an)





The "hunting" accident is a (an)



The "crypto hacks" we have seen are