Fairness in Byzantine Consensus

Gengrui Zhang, PhD Candidate
University of Toronto

Seminar talk at

Today's Agenda

- The permissioned vs. permissionless
- Consensus protocols in the age of blockchains
  - Tolerating benign failures
  - Tolerating Byzantine failures
- State-of-the-art algorithms at a glance
- Our work:
  - Suppressing Byzantine behavior
  - Fairness in Byzantine consensus

Categorizing blockchains based on consensus

Permissionless

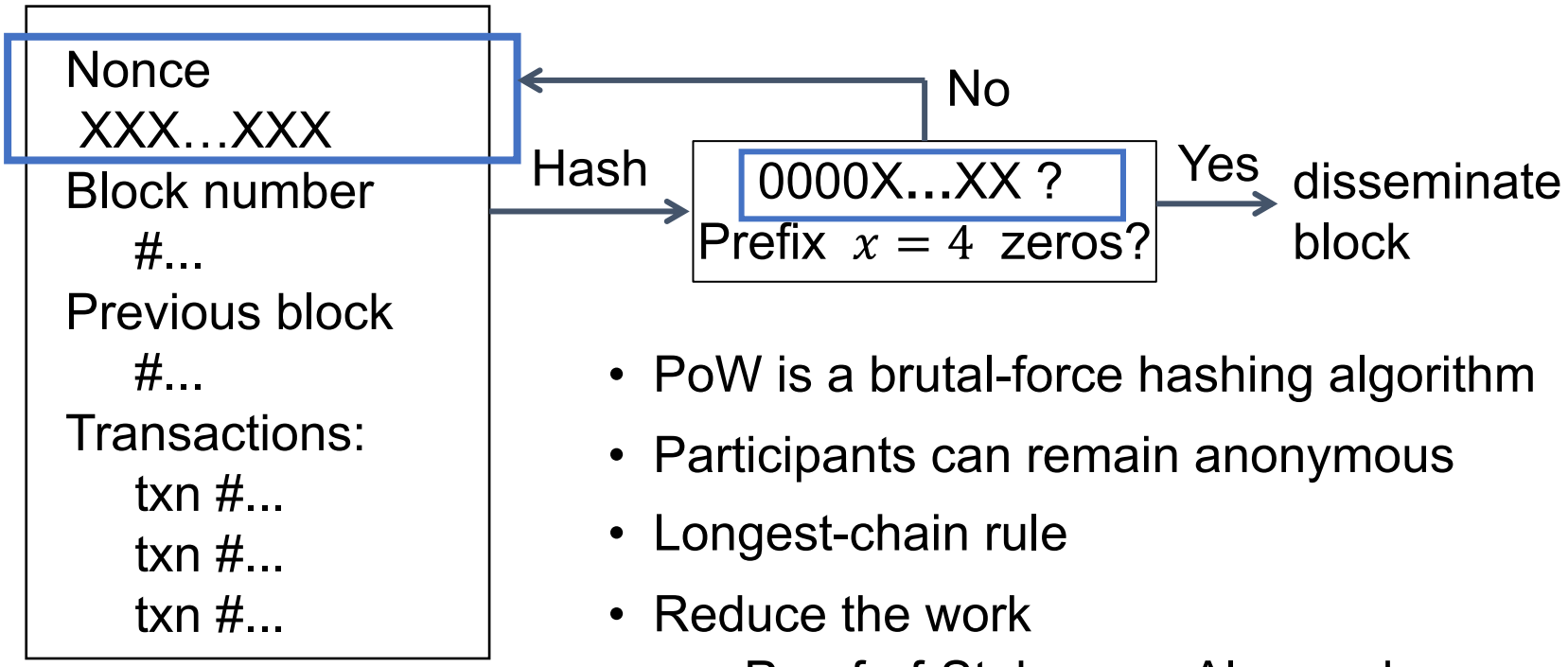
- Freely join or leave without node management
- Open distributed ledger
  - Bitcoin
  - Ethereum
- Proof-of-X protocols
  - Proof-of-Work
  - Proof-of-Stake [OSDI '17]

Permissioned

- Join by permission, requiring node management
- Shared distributed database
  - HyperLedger (IBM)
  - Libra (Facebook)
- BFT consensus
  - PBFT [OSDI '99]
  - Zyzzyva [SOSP '07]
  - HotStuff [PODC '19]
  - Pompē [OSDI '20]

The Proof-of-X family

- Proof-of-Work (PoW)
  - Hard to solve, but easy to verify

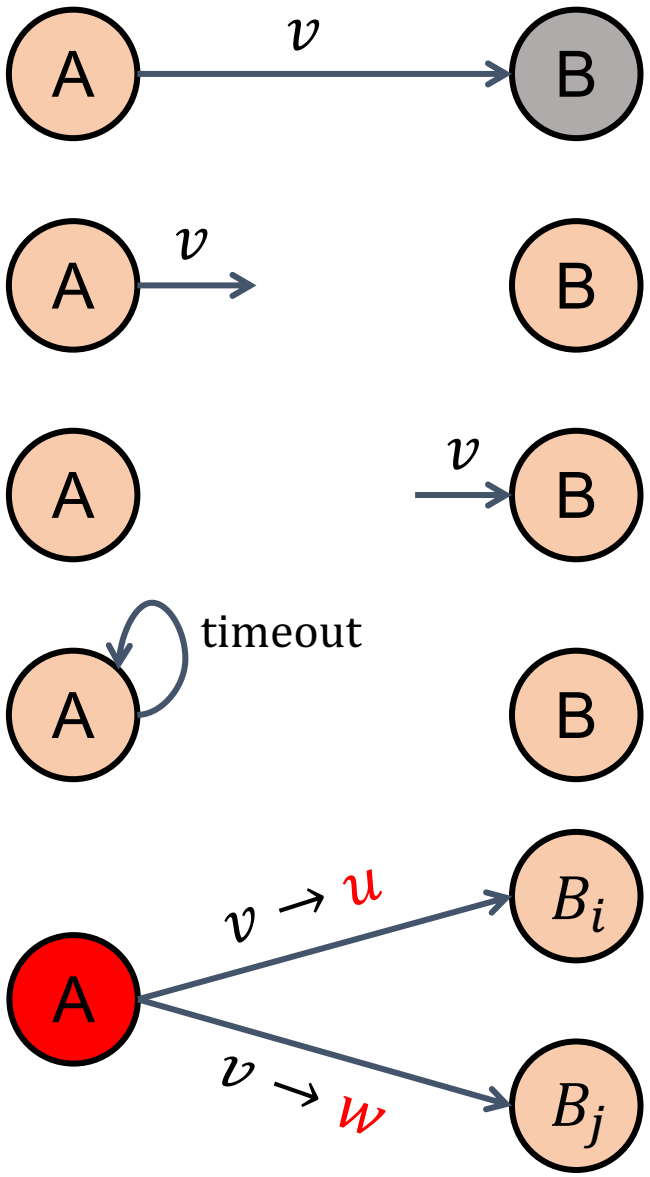


- PoW is a brutal-force hashing algorithm
- Participants can remain anonymous
- Longest-chain rule
- Reduce the work
  - Proof-of-Stake: e.g, Algorand [OSDI '17]

Failure model

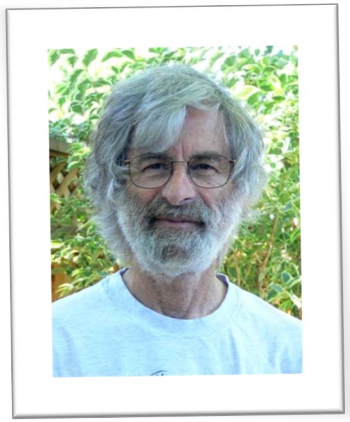
- Benign failures
  - Crash failures
  - Omission failures
    - Send omission
    - Receive omission
  - Timing failures

- Byzantine failures
  - Arbitrary failures
  - Malicious attacks



Consensus in the presence of failures

- Benign failures
  - Paxos
    - Viewstamped replication [PODC '88]
    - Raft [ATC '13]



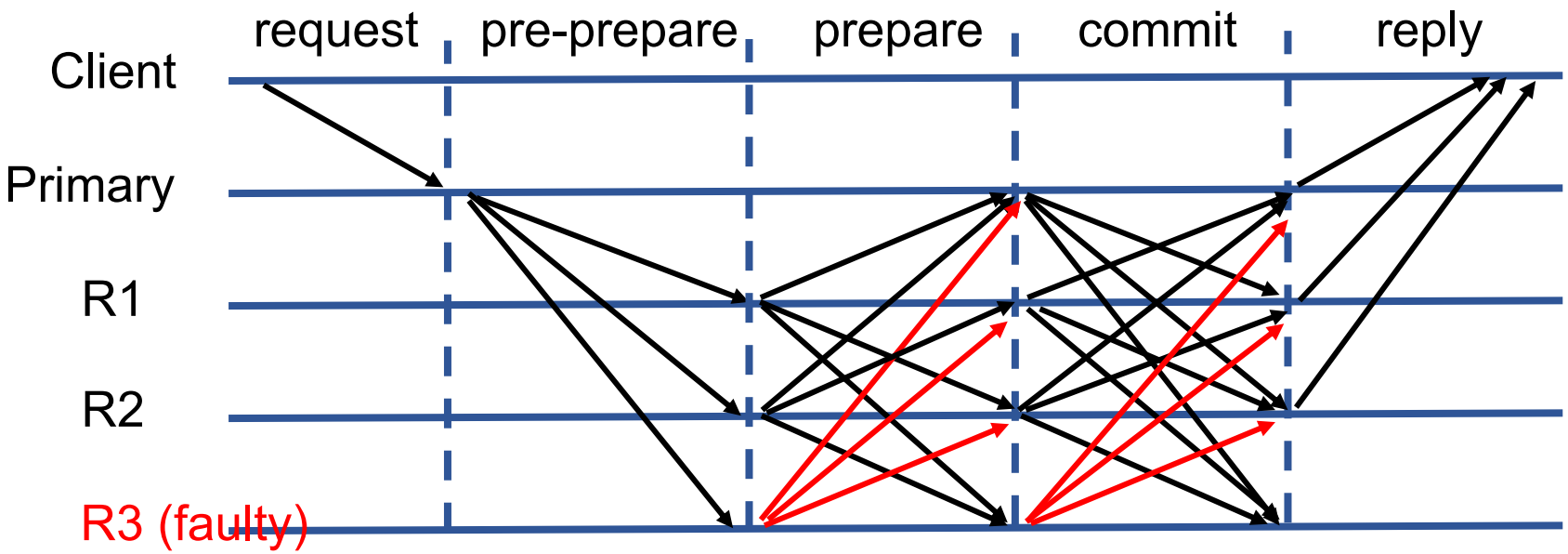
Quorum: $2f + 1$
Simple majority

http://www.lamport.org
e.g., Latex, logical clock, Byzantine general problems

- Byzantine failures
  - PBFT [OSDI '99]
    - Zyzzyva [SOSP '07]
    - BFT-SMaRt [ATC '13]
    - HotStuff [PODC '19]
    - Pompē [OSDI '20]

Quorum: $3f + 1$
Byzantine agreement
Byzantine broadcast

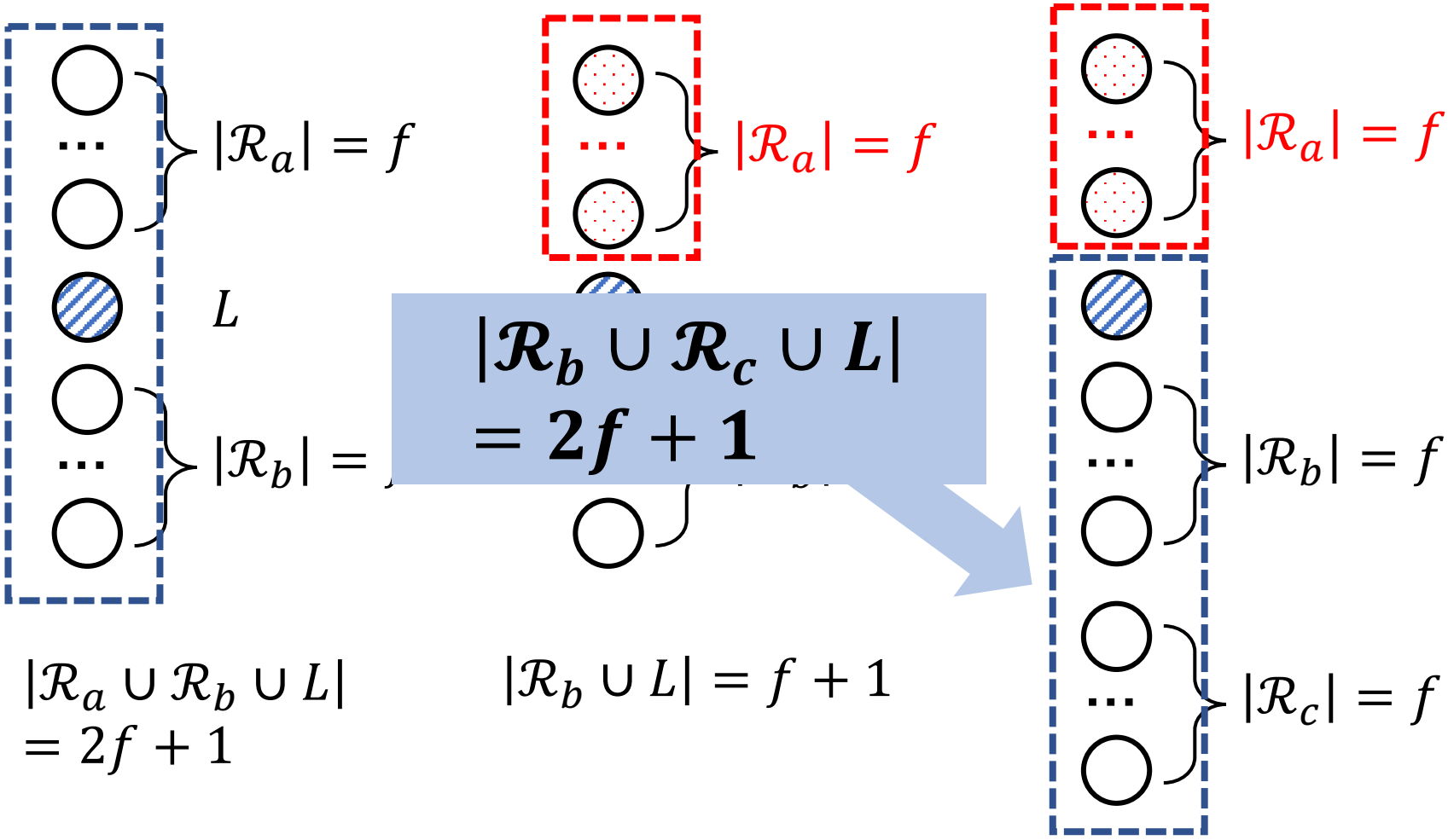
The celebrated PBFT



- A client sends a request to invoke a service operation to the primary
- The primary multicasts the request to the backups
- Replicas execute the request and send a reply to the client
- The client waits for $f + 1$ replies from different replicas with the same results

Lower bound of $3f + 1$

Consider a simple 2PC process tolerating f Byzantine failures



BFT consensus for permissioned blockchains

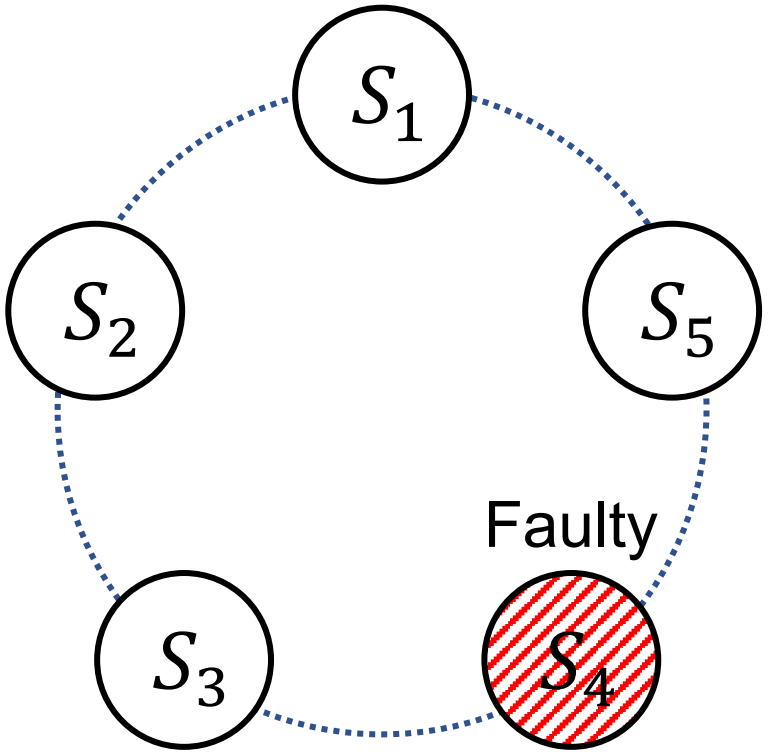
Along with blossomed blockchain applications, numerous BFT consensus have been designed and deployed

BFT algorithms	Normal operation	Leader failure	f Leader failure
PBFT (OSDI '99)	$O(n^2\|M\|)$	$O(n^3)$	$O(fn^3)$
Zyzzyva (SOSP '07)	$O(n\|M\|)^*$	$O(n^3)$	$O(fn^3)$
BFT-SMaRt (ATC '13)	$O(n^2\|M\|)$	$O(n^3)$	$O(fn^3)$
SBFT (DSN '19)	$O(n\|M\|)$	$O(n^2)$	$O(fn^2)$
HotStuff (PODC '19)	$O(n\|M\|)$	$O(n)$	$O(fn)$

*only in optimal path

So, what next?

Round-robin leader selections: $\mathcal{P} = view \ mod \ |\mathcal{R}|$



- Always leaves a window for correct servers to take the leader duty

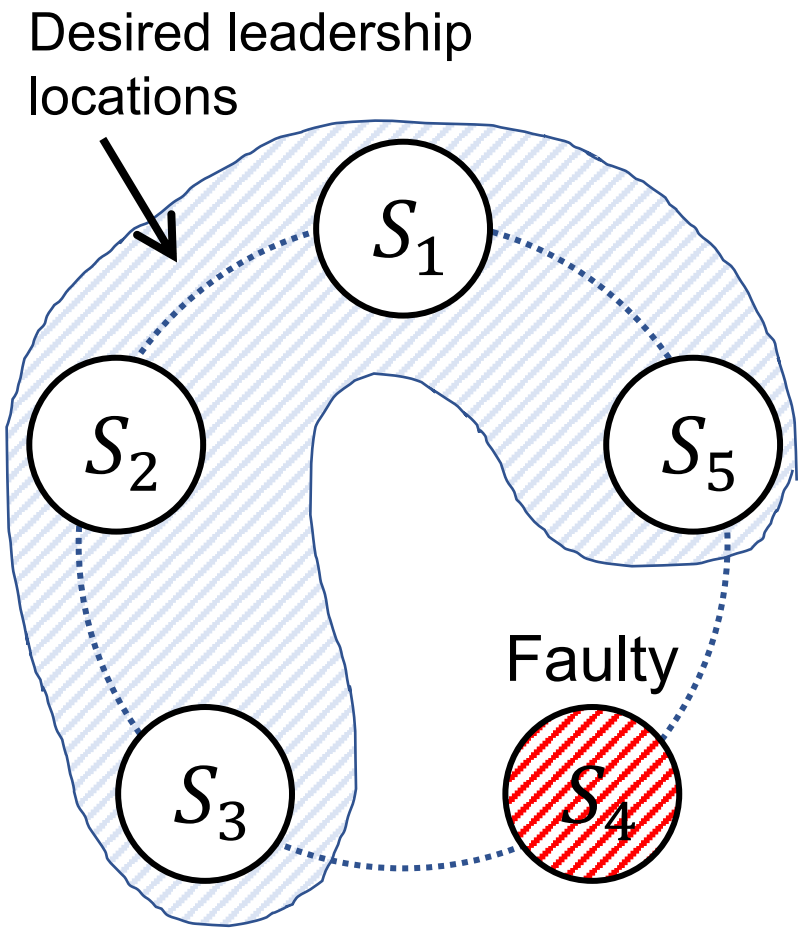
fair, simple , and *painful*

- Faulty servers have the chance to be assigned as next leader

Pr of unavailable leaders:

$$Pr = \frac{f}{|\mathcal{R}|} \approx 20\%$$

Is fairness the ultimate option for selecting leaders?

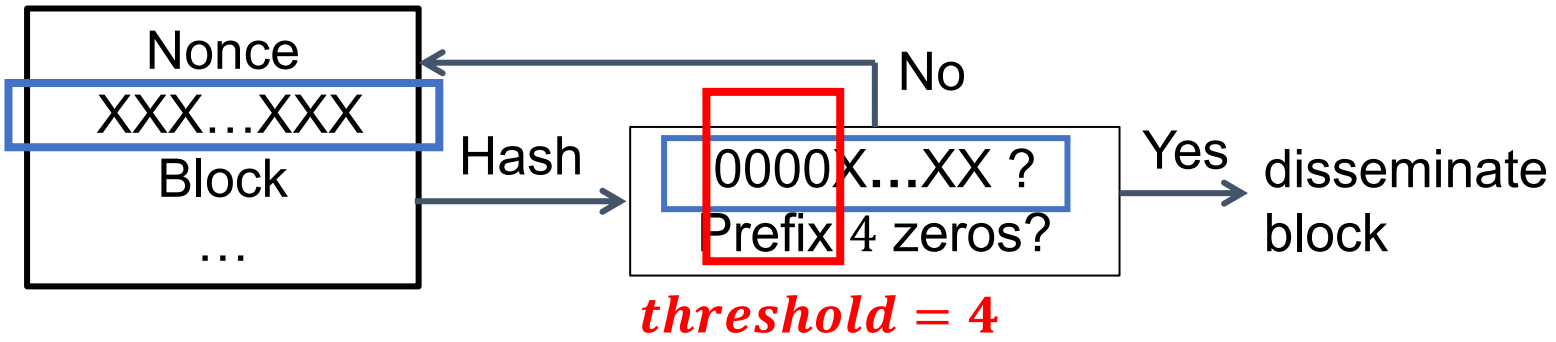


Question arises:

*How can we design efficient BFT algorithms that **not only tolerate** Byzantine failures, but also **alleviate systems from being impaired by failures** to enhance availability?*

Penalize suspicious servers by detecting faulty behaviors, thereby pushing leader duty to correct servers

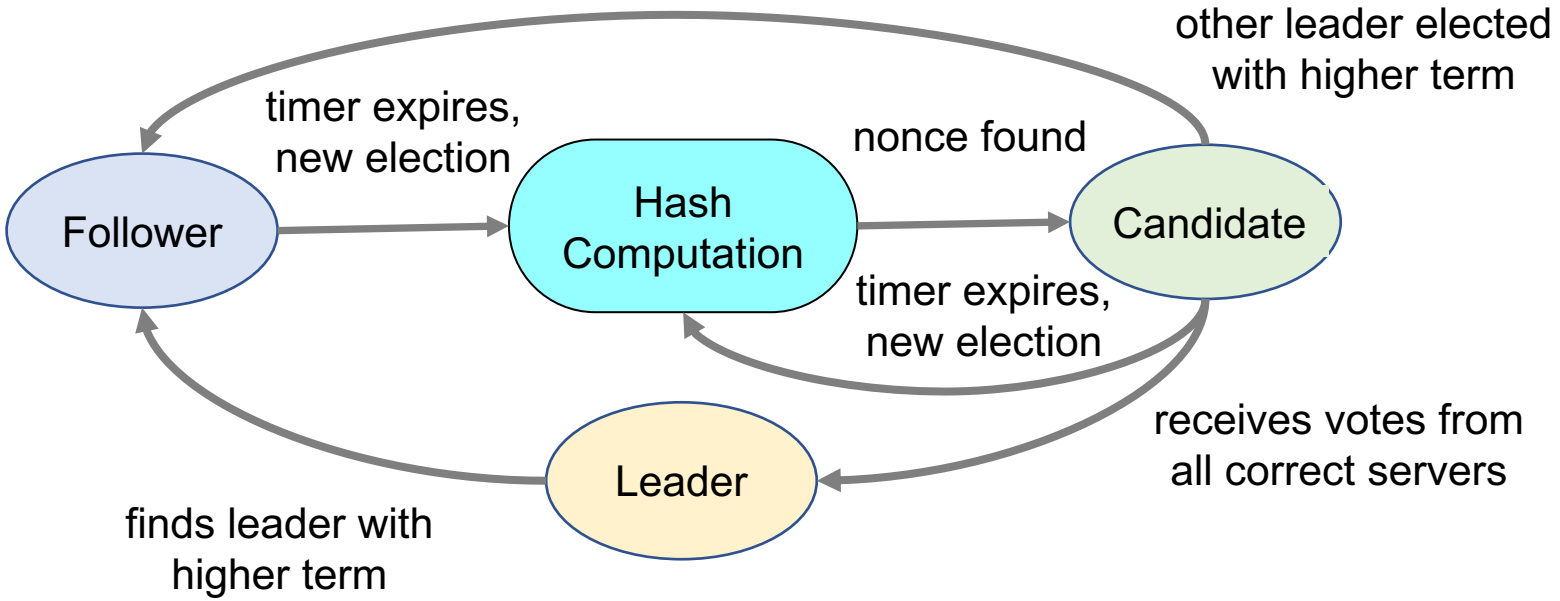
Leveraging PoW into BFT consensus



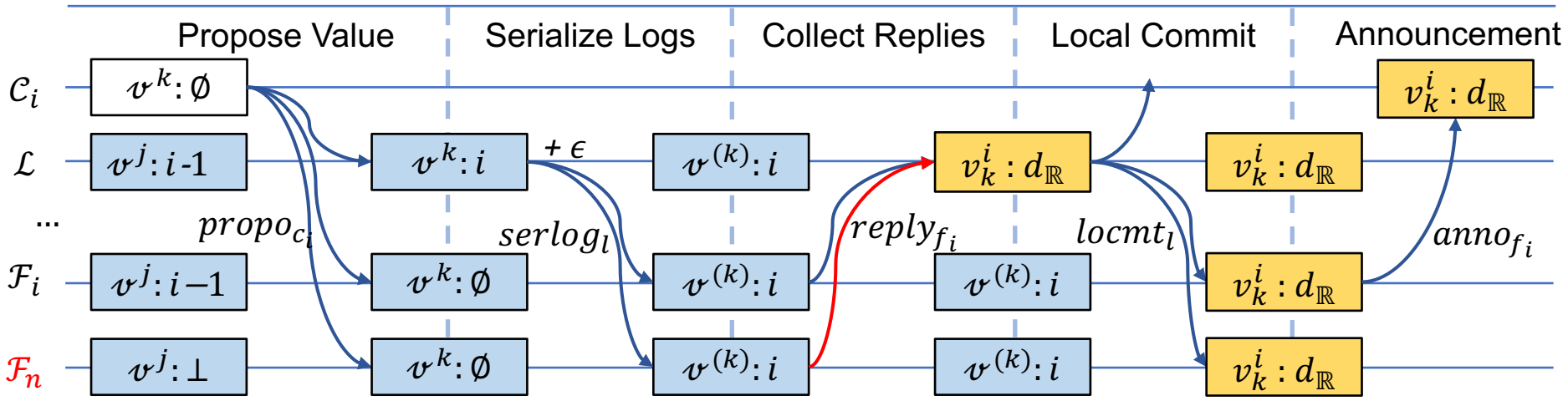
- The more zeros a result prefixes, the more iterations the hashing process requires
- Cost of hash computation can be dynamically adjusted by changing thresholds
- Utilizing PoW as a tool to discourage misbehaved servers can marginalize Byzantine servers out of participating consensus

The Proof-of-Commit leader election

- Becoming a leader engages performing computation
- The more election initiated, the more time-consuming the computation will be (**penalization!**)



Linear message transmission in log replication



- Utilizes leaders as a bridge for leader-followers communication instead of using $n - to - n$ followers-to-followers communication
- Holds properties of Byzantine agreement
- Reduce message delivery from $O(n^2)$ to $O(n)$

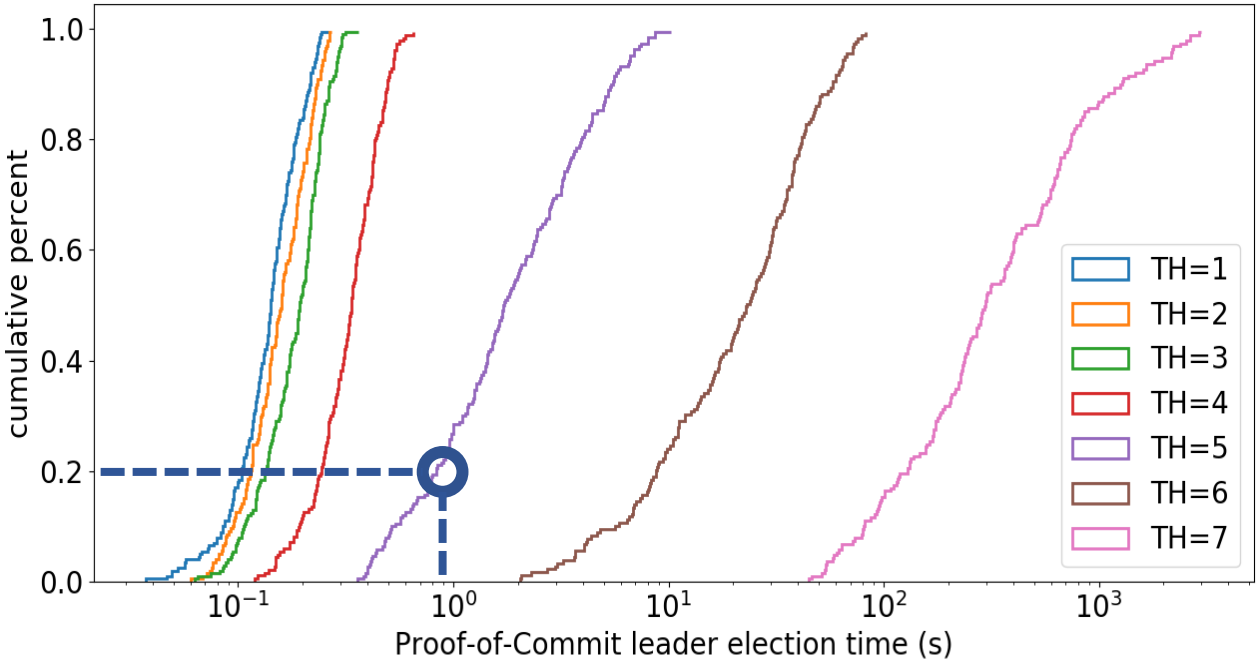
Compliant-based client interaction

- Every server has a **proposal timer** to limit the time for a proposal to be committed
- First, if the proposal timer keeps expiring, clients send $clientComplain$ messages to connected servers
- Next, servers forward $clientComplain$ to the leader and broadcast $secondComplain$ messages
- Then, upon receiving $f + 1$ $secondComplaint$ messages, a server starts to count down its **election timer**.
  - If the proposal is committed within election timeout, server resets election timer
  - If election timer expires, the server considers leader is faulty and starts its leader election campaign

Suppressing Byzantine servers

- Suppose Byzantine servers as a cohort have a computation ability, δ,
  - Before Byzantine servers exhaust δ, correct leaders may appear in between attacks
  - After Byzantine servers exhaust δ, Proof-of-Commit leader election guarantees correct leaders.
- After difficulty exceeds δ, Byzantine servers will vanish into repeatedly performing demanded computation, converging systems to failure-free operations

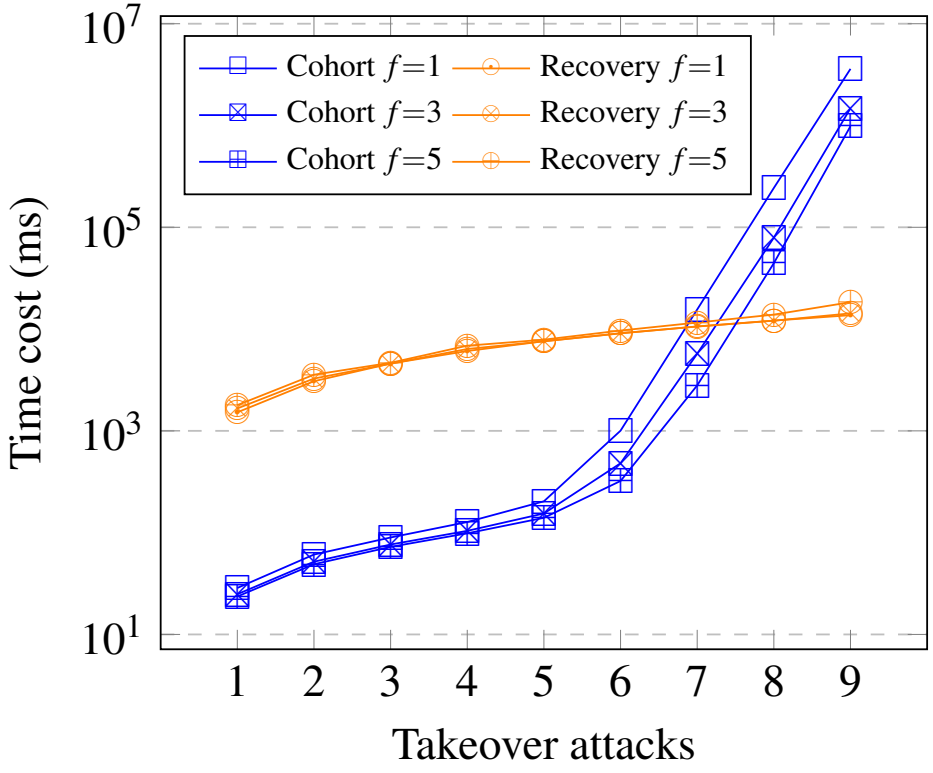
Preliminary results in PoC leader election



- ❖Cluster of 16 servers
- ❖1,000 independent runs under varying thresholds

- When threshold < 5, election time differs by a narrow margin
- When threshold > 5, election time soars, and when threshold > 7, election time goes skyrocketing

Byzantine attacks and recovery



In a 16-server cluster, where $f_{max} = 5$

- Timeout is set to 1s
- Strategy for Byzantine servers:
  - Take over the leadership whenever they are not leaders
- Strategy for correct servers:
  - Follow timing requirement; initiate new elections based on timeouts

Lessons learned

- Penalty can only be imposed based on suspicion [FLP]
  - Ambiguity could lead to false penalization

- Applying PoW may threaten system liveness
  - Byzantine servers with omnipotent computation power (in theory)
  - Byzantine servers pretend to be correct
  - Correct servers may compete for leadership

Thank you! Questions?

Gengrui Zhang (张耕瑞)
gengrui.zhang@mail.utoronto.ca