Reputation-based Consensus Algorithms: Binding Efficiency and Robustness

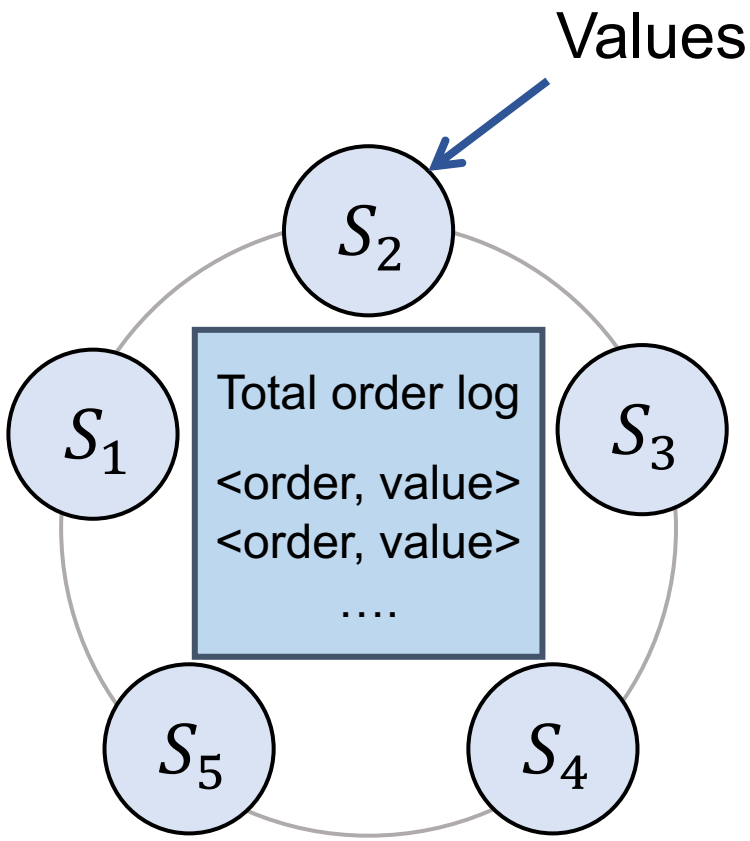
Gengrui (Edward) Zhang, PhD Candidate

Advisor: Hans-Arno Jacobsen

University of Toronto

Consensus algorithms and fault tolerance

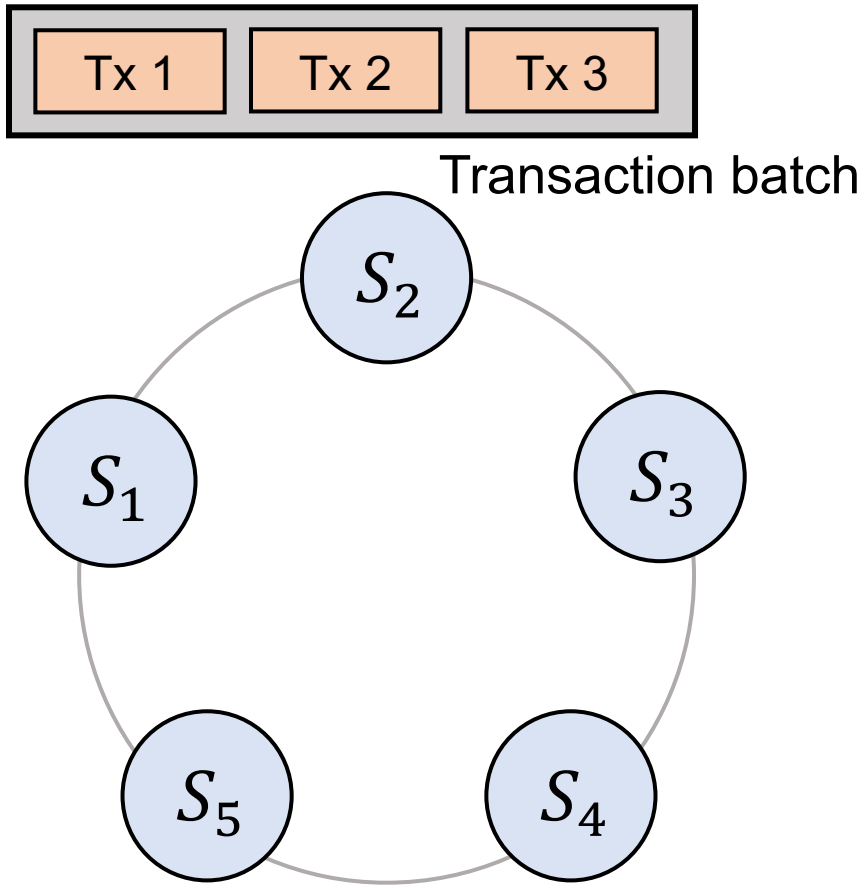
- State machine replication (SMR)
  - A collection of servers compute identical copies of the same state
- Byzantine failures
  - Faulty servers can exhibit arbitrary and malicious behavior; they can also collude to perform attacks
  - Faulty behavior can be intentional



The new state of Byzantine faulty servers and the content of the messages sent are completely unconstrained

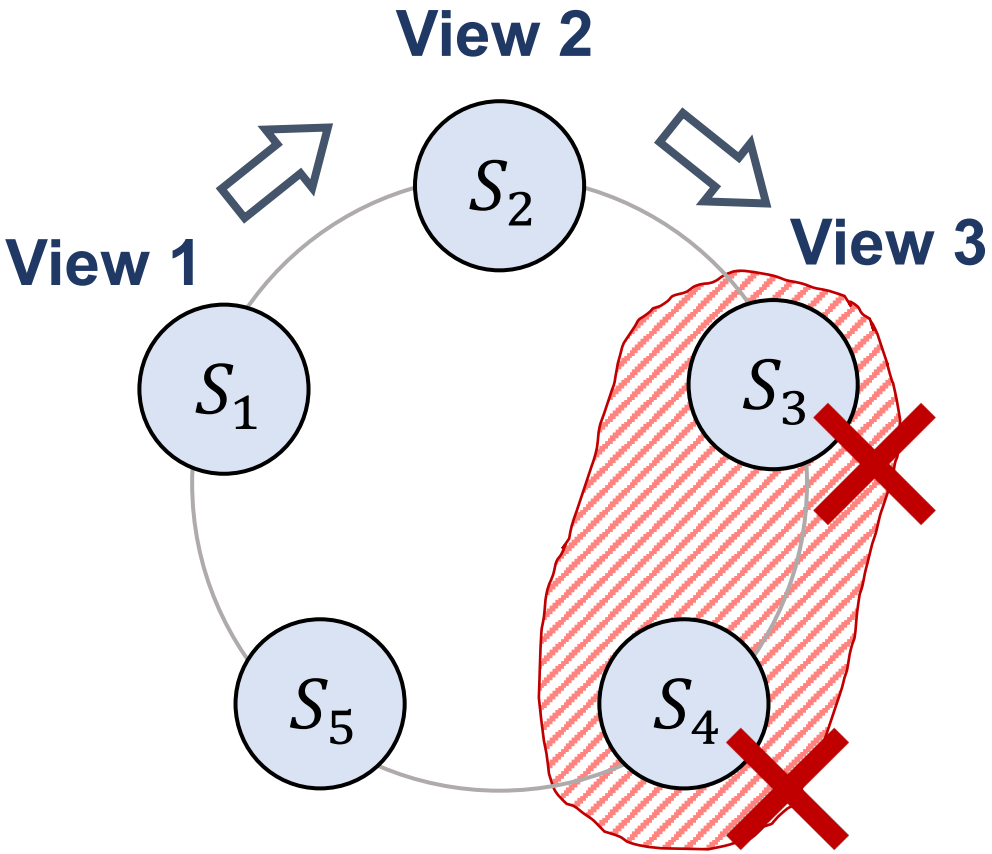
How's Byzantine fault-tolerance doing?

- **Explicit faults:** detectable behavior
  - Stop responding
  - Send erroneous messages
- **Implicit faults:** hard to detect; sometimes undetectable
  - Manipulate transaction orders
    - Democratic ordering:
    - E.g., Pompe [OSDI`21]
  - Exploit timer timeouts
    - Performance monitor and frequent leadership rotation
    - E.g., Aardvard [NSDI`08], RBFT [ICDCS`13], DiemBFT



Passive leadership rotation

- All servers follow **a pre-defined leader schedule** to rotate leadership
  - LeaderID = View *mod* # of servers
    - I.e., leadership is assigned to $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_1 \rightarrow \cdots$
- Pros:
  - Simple; easy to understand and implement
- Cons:
  - Cannot avoid scheduled but crashed servers assigned with leadership



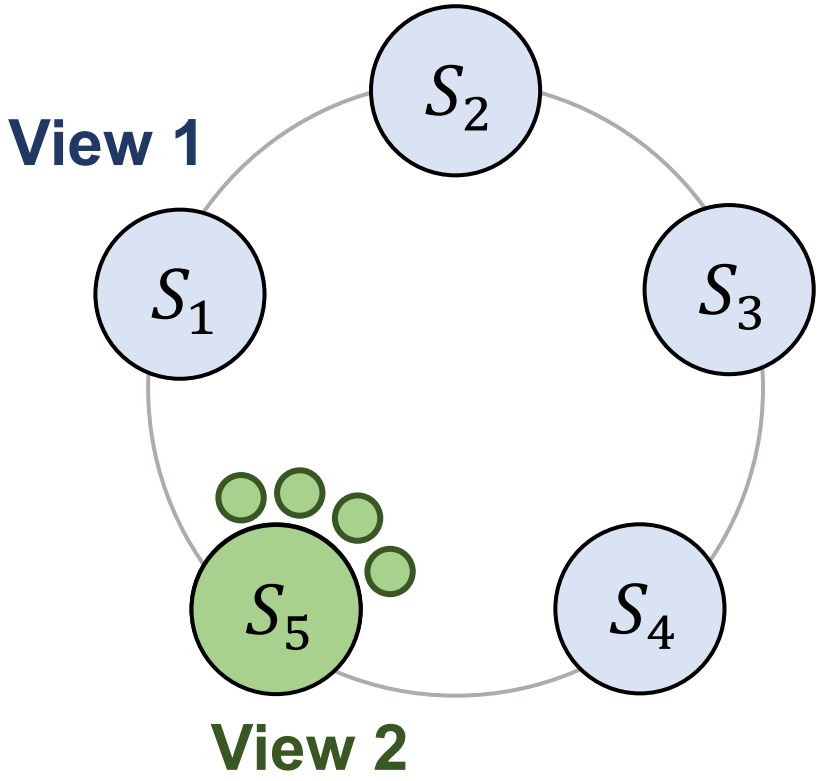
Active vs. passive view changes

- Active view change
  - No leader schedule
  - Whoever detects a leader's failure proactively campaigns for leadership

	Passive	Active
Normal operation	–	–
Crash failures	↓	–
Byzantine failures	↓	↓↓ (?)

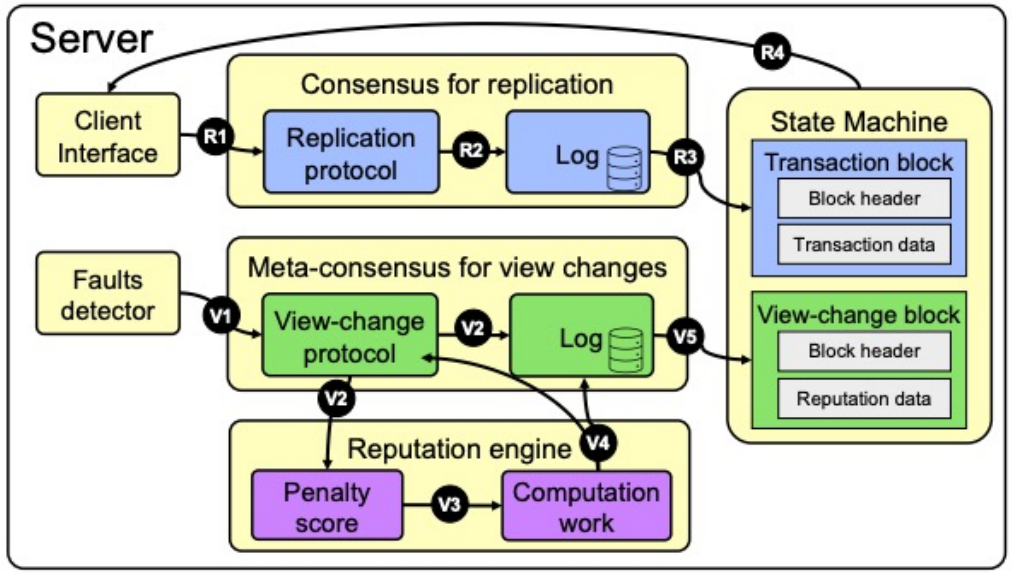
Design goals:

1. Suppress faulty servers to be elected
2. Let attackers pay (misbehavior comes at a cost)

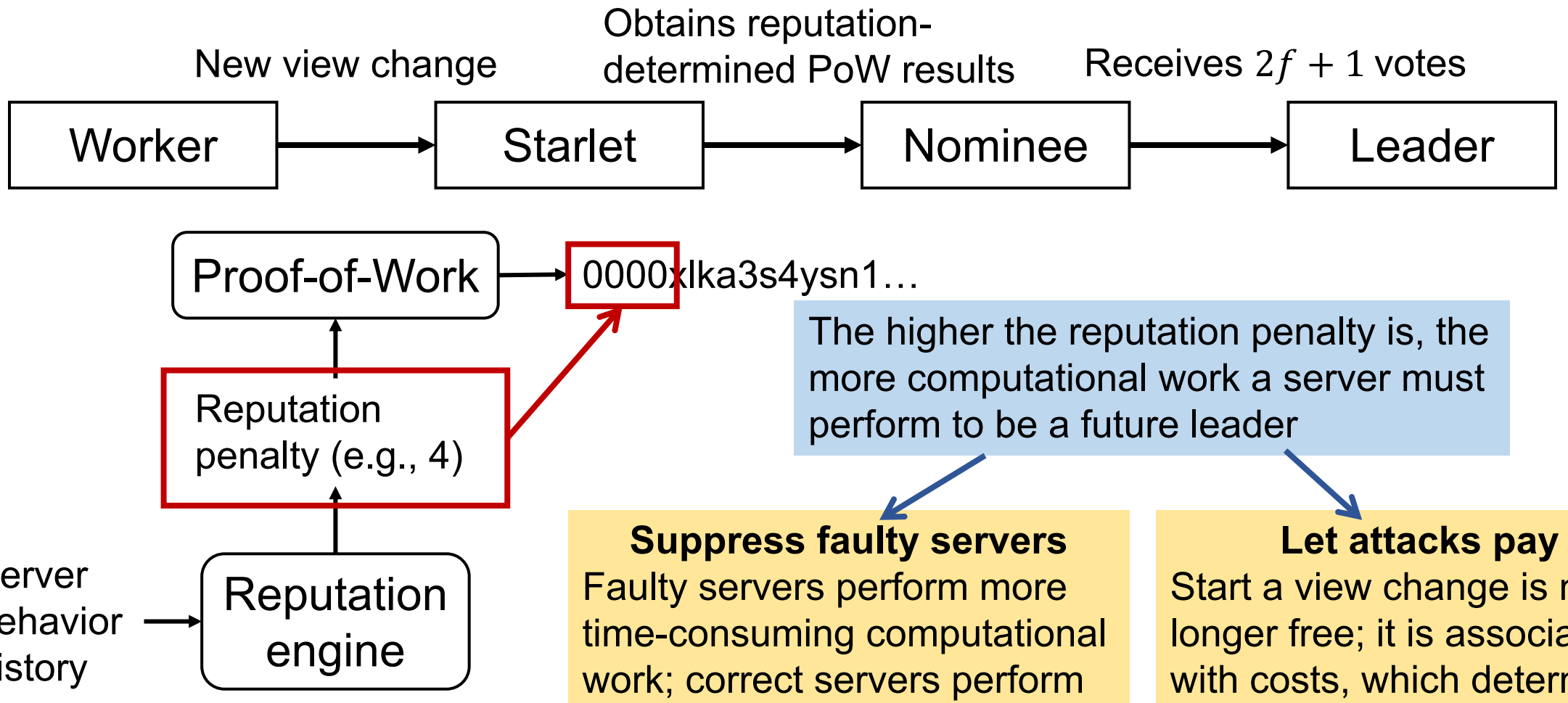


Reputation-based BFT consensus algorithms

- Extending state machine replication properties to a **reputation state**
  - A server's reputation state is indictive of the server's correctness, i.e., being correct or malicious
  - The reputation state is calculated based on servers' past behavior



Reputation-determined state transition



The higher the reputation penalty is, the more computational work a server must perform to be a future leader

Suppress faulty servers

Faulty servers perform more time-consuming computational work; correct servers perform less work

Let attacks pay

Start a view change is no longer free; it is associated with costs, which determined by reputation

UNIVERSITY OF TORONTO

Reputation engine: translating behavior to reputation penalty

Good behavior

Short link of VC blocks
Unzealous leadership competition; servers are indifferent to becoming the next leader

Long link of TX blocks
Up-to-date replication of transactions; servers are eager to participate in consensus

Reputation penalty

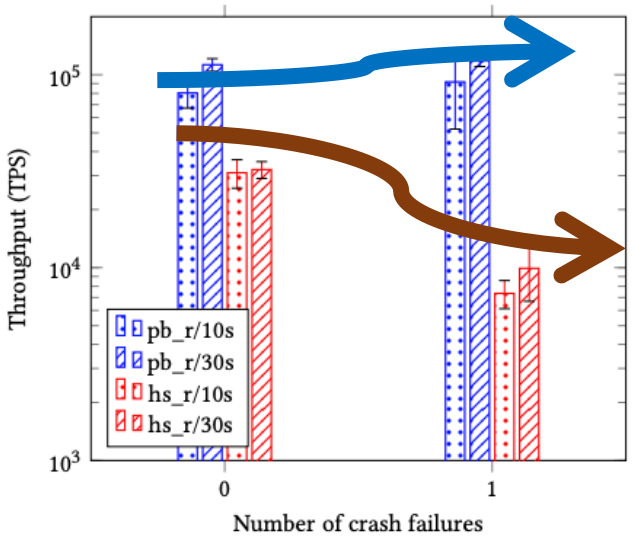
Translation of behavior in replication (TX blocks)

$$p^{(v')} = p^{(v')} - \max\{0, \lfloor \frac{|\mathcal{T}_{left}|}{|\mathcal{T}_{all}|} \times \frac{p^{(v)} - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}}} \rfloor\}$$

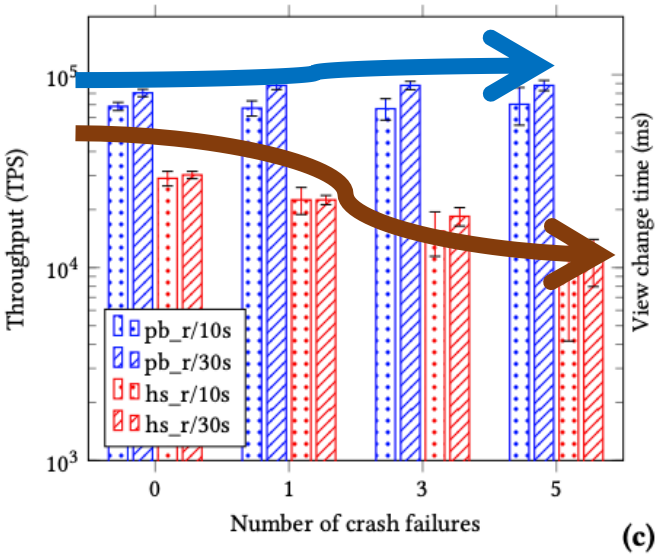
Translation of behavior in view changes (Z-core of all past penalties in VC blocks)

UNIVERSITY OF TORONTO

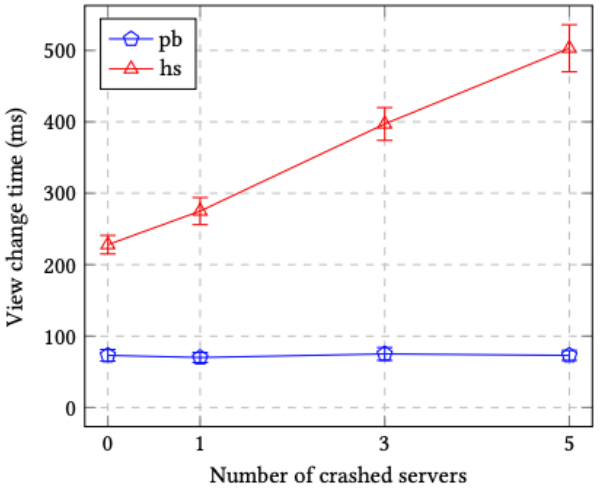
Performance under crash failures



(a) $n = 4$.



(b) $n = 16$.

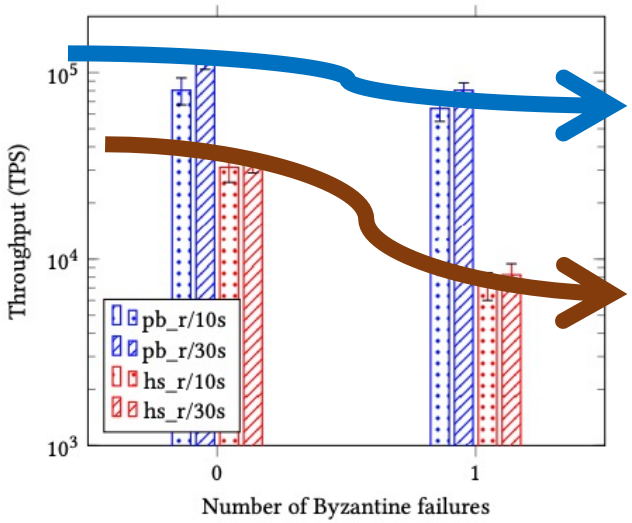


(c) Successful view change time (n=16), excluding timeouts.

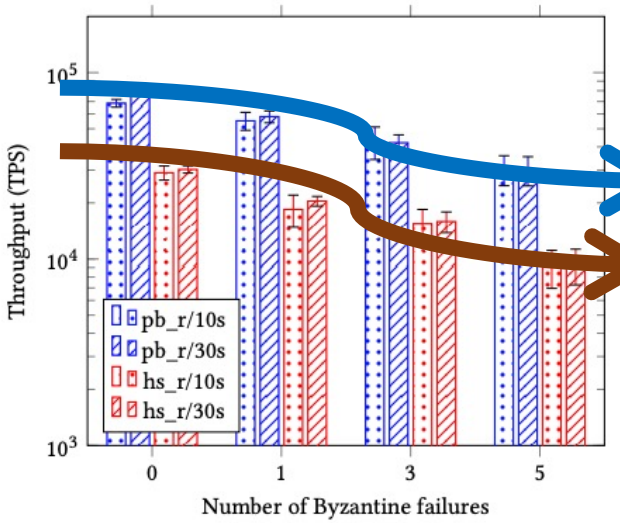
- Baseline: HotStuff; 4 and 16 nodes
- Rotate leaders per 10s (r/10s) and per 30s (r/30s)

Reputation-based active view changes can completely avoid crashed servers in leadership changes

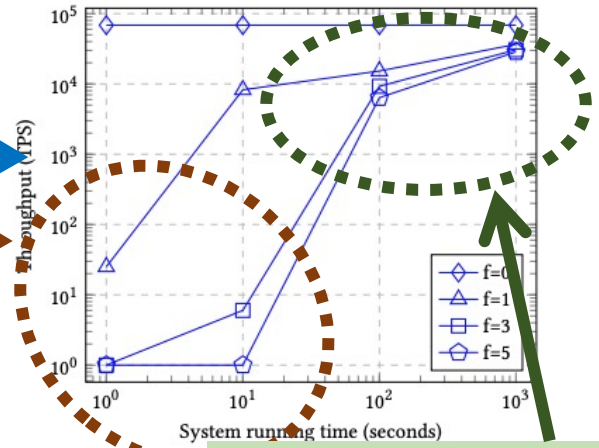
Performance under Byzantine failures



(a) $n = 4$.



(b) $n = 16$.



(c) Throughput (uous Byzantine

Reputation-based active view changes can gradually suppress faulty servers with a moderate performance drop

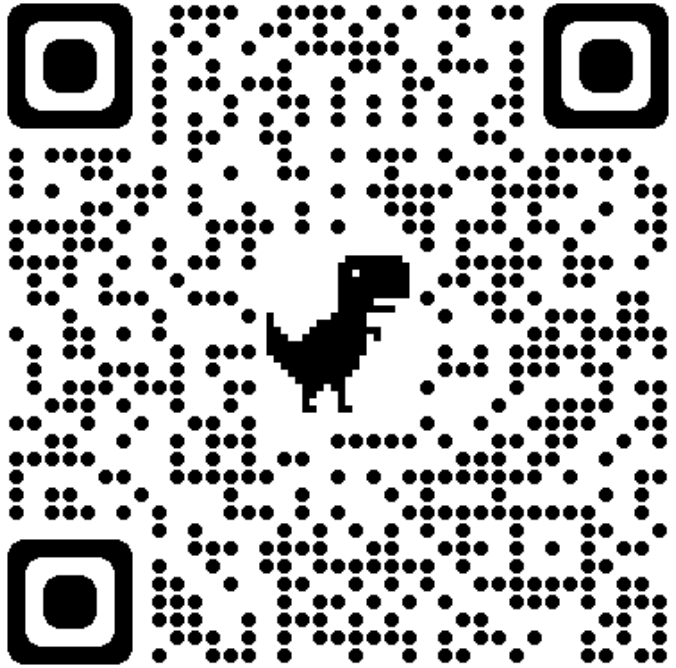
Byzantine servers perform attacks

Byzantine servers have a high reputation penalty after performing attacks and are suppressed

UNIVERSITY OF TORONTO

Questions?

Gengrui (Edward) Zhang, PhD Candidate
University of Toronto
Website: gengruizhang.github.io



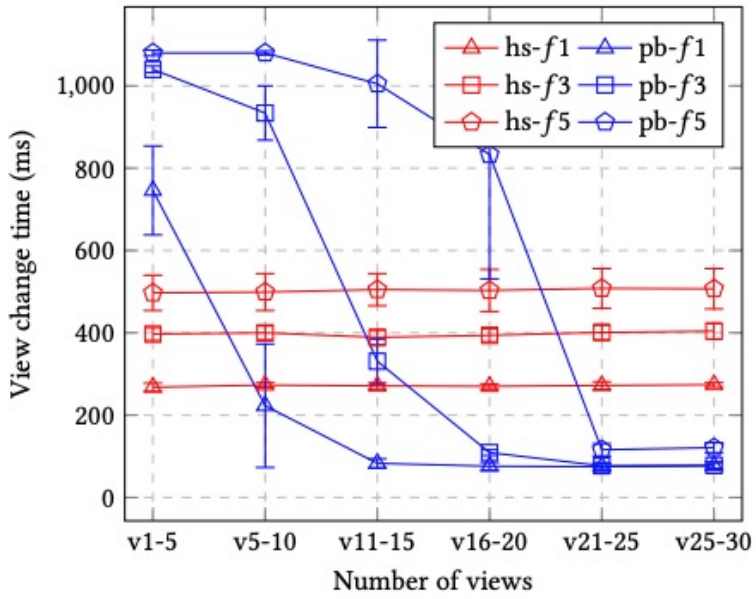
UNIVERSITY OF TORONTO

Tolerating failures vs. reconfigurations

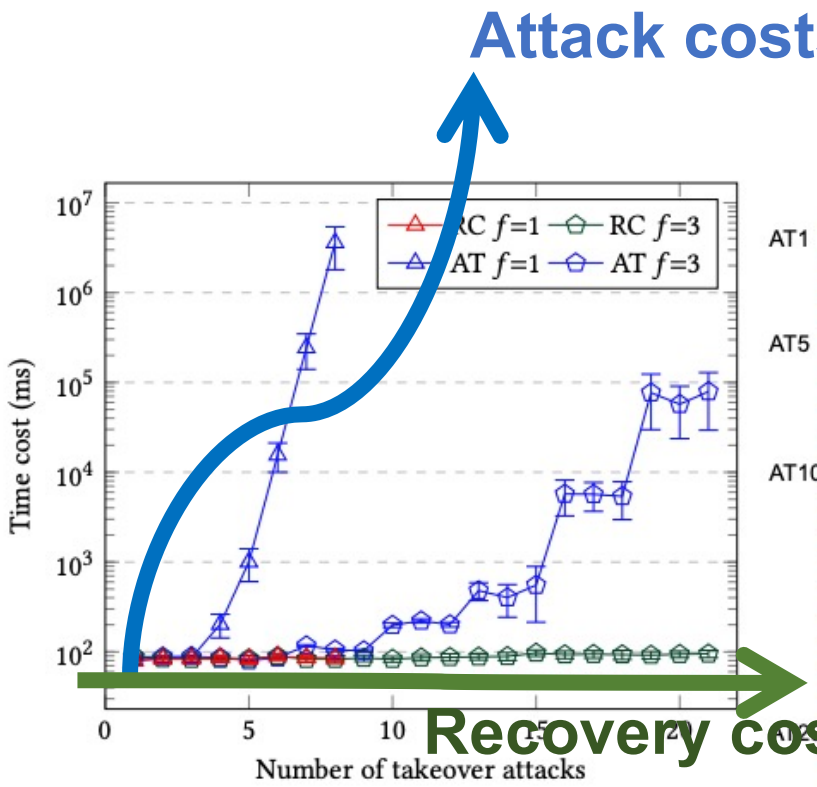
- It is impossible to pass an absolute judgement of which server is Byzantine faulty in the presence of asynchrony

- Do we want to reconfigure the system each time a server fails to exhibit an expected behavior?
- Fault tolerance is not kicking out faulty servers (i.e., reconfiguration)

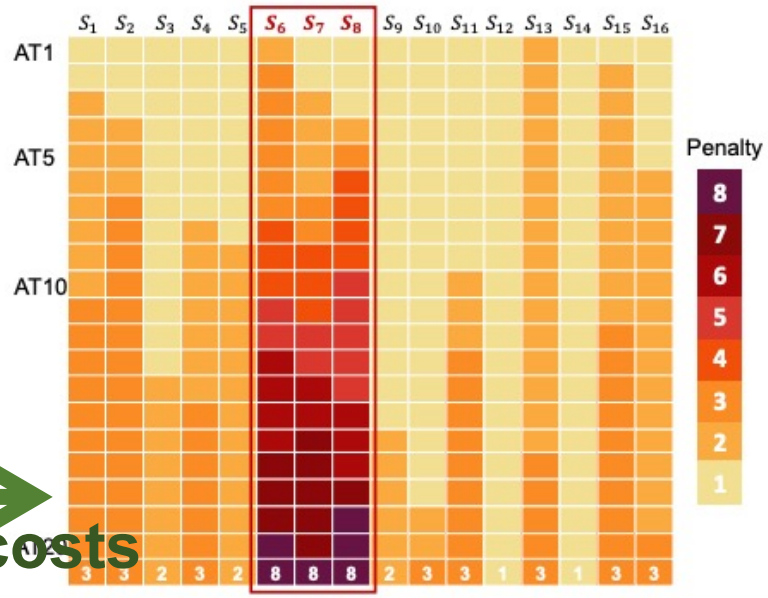
Performance of the reputation engine



(a) The progression of view change time under repeated Byzantine failures.



(b) The time cost of attacks (AT) from Byzantine servers vs. recoveries (RC) from correct servers.



(c) Penalty changes of all servers when f=3 and n=16 under increasing number of attacks (AT).

UNIVERSITY OF TORONTO