SBFT: a Scalable and Decentralized Trust Infrastructure

By Gueta et al.,2018

Presented By

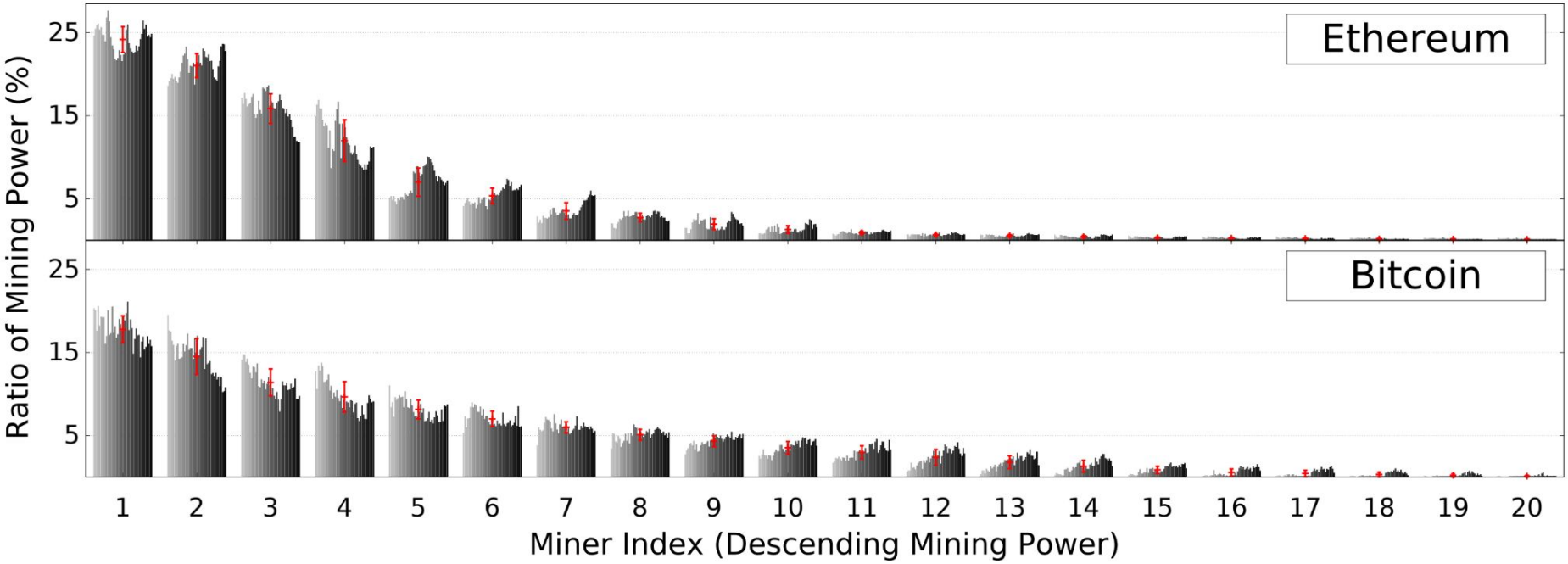
Vasudha Jha, Anusha Kulkarni, Parichaya Chatterji, Preethi Muruganandam and Michael Yang

Outline

1. Introduction
2. SBFT Fast-Path Protocol
3. Linear PBFT Protocol
4. View Change Protocol
5. Performance Evaluation

Why another BFT protocol?

Centralization of power



Gencer et al., “Decentralization in Bitcoin and Ethereum Networks”, 2018

At a global scale, we need to **tolerate tens of malicious nodes**

SBFT has been experimentally evaluated on a deployment of **209 active replicas**, withstanding a malicious adversary controlling **f = 64** replicas.

SBFT provides **~2x better throughput** and **~1.5x better latency** relative to a highly optimized PBFT system

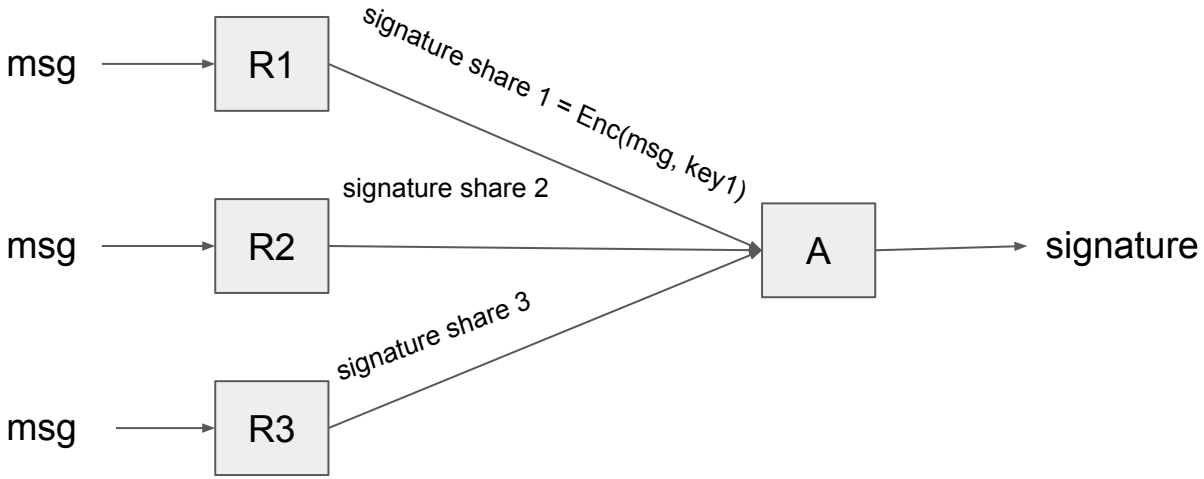
System Model

- Asynchronous
- Possible Faults:
  - Delayed Messages
  - Dropped Messages
  - Byzantine Faults
- $n = 3f + 2c + 1$, where $c <= f/8$
  - n = total number of replicas
  - f = number of faulty nodes
  - c = number of crashed or straggler nodes

Protocol Properties

- **Safety**
- **Liveness**
- **Linearity**

Cryptography



Threshold signatures

Threshold Signature Scheme

BLS Signatures

- **Robust** - signers can filter out invalid signature shares
- **Shorter** than RSA signatures (33B vs 256B)
- **Computationally cheaper**
- Supports **batch verification**

The 4 ingredients of SBFT for Performance Improvement

Ingredient 1: From PBFT to linear-PBFT

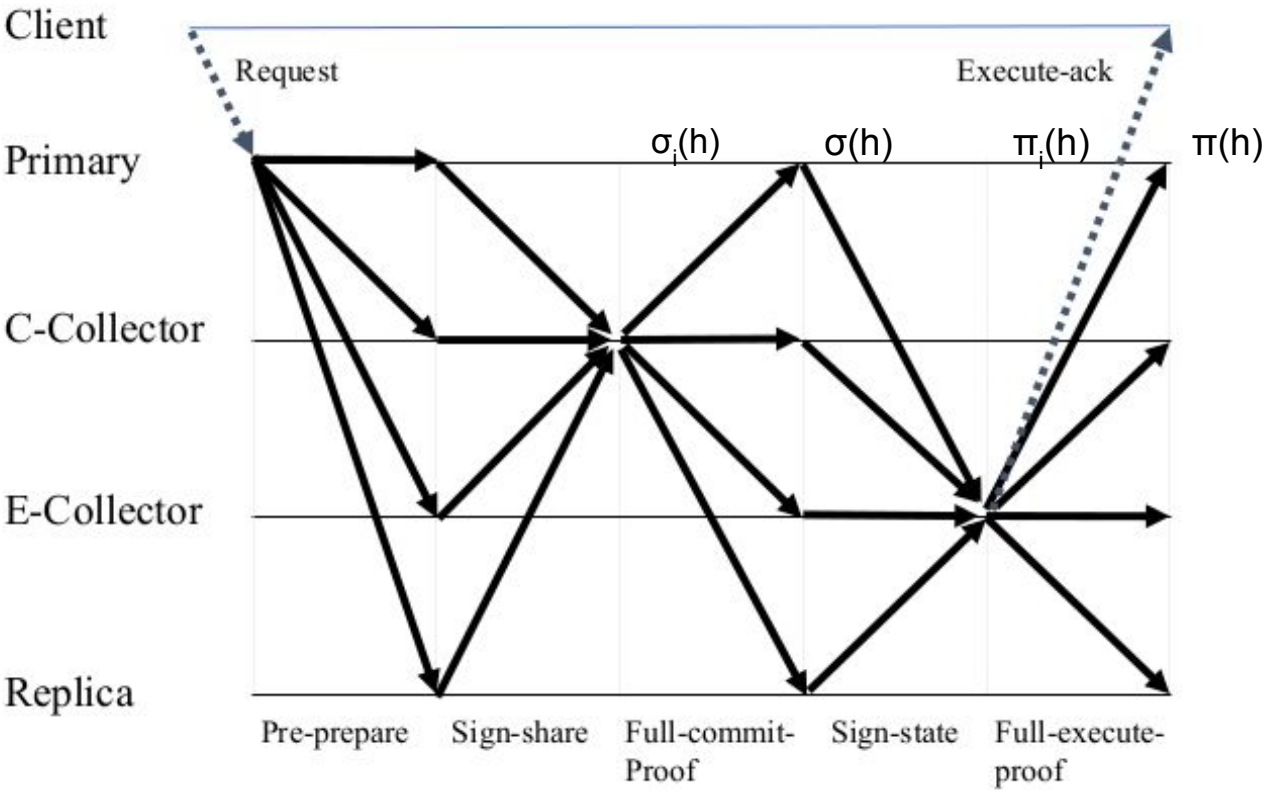
Ingredient 2: Adding a fast path

Ingredient 3: Reducing client communication from f+1 to 1

Ingredient 4: Adding redundant servers to improve resilience and performance

SBFT Fast Path

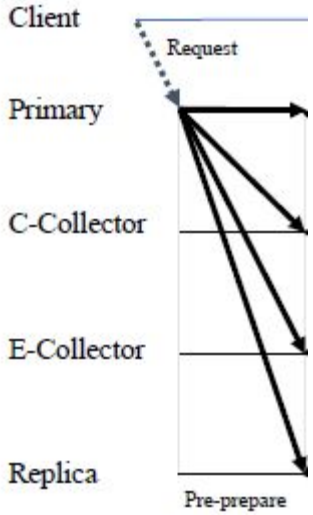
$n = 3f + 2c + 1$



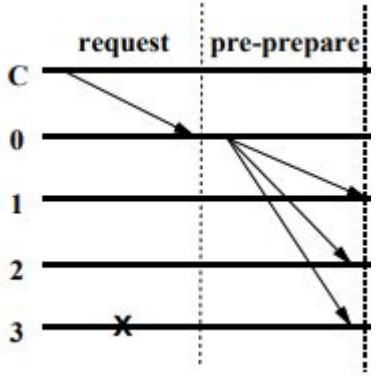
Threshold Signatures

- **Fast path sign-share σ** with threshold (3f + c +1)
- **Slow path sign-share τ** with threshold (2 f + c + 1)
- **Execution sign-share π** with threshold (f + 1)

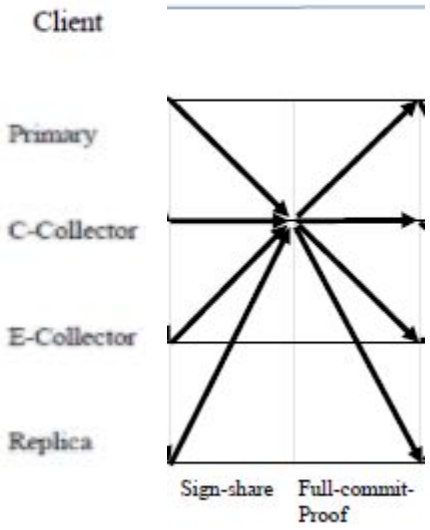
SBFT


Client
Request
Primary
C-Collector
E-Collector
Replica
Pre-prepare

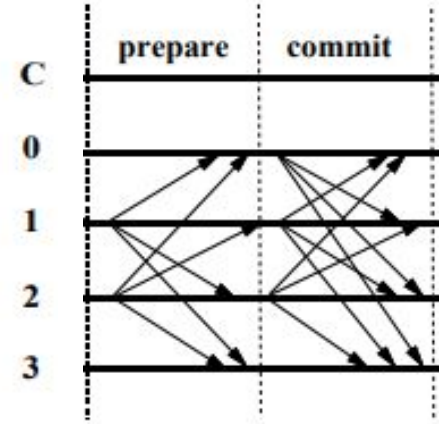
PBFT


request
pre-prepare
C
0
1
2
3

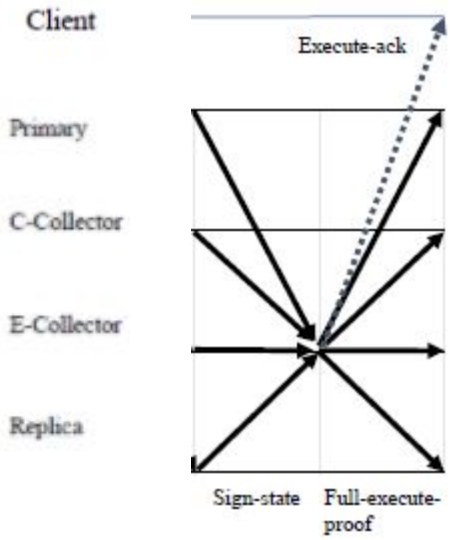
SBFT


Client
Primary
C-Collector
E-Collector
Replica
Sign-share
Full-commit-Proof

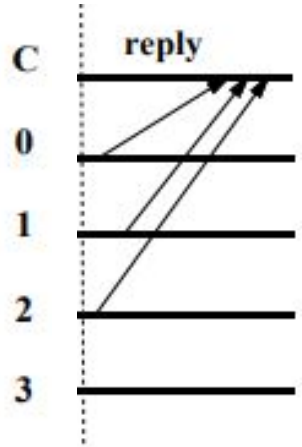
PBFT


prepare
commit
C
0
1
2
3

SBFT


Client
Execute-ack
Primary
C-Collector
E-Collector
Replica
Sign-state
Full-execute-proof

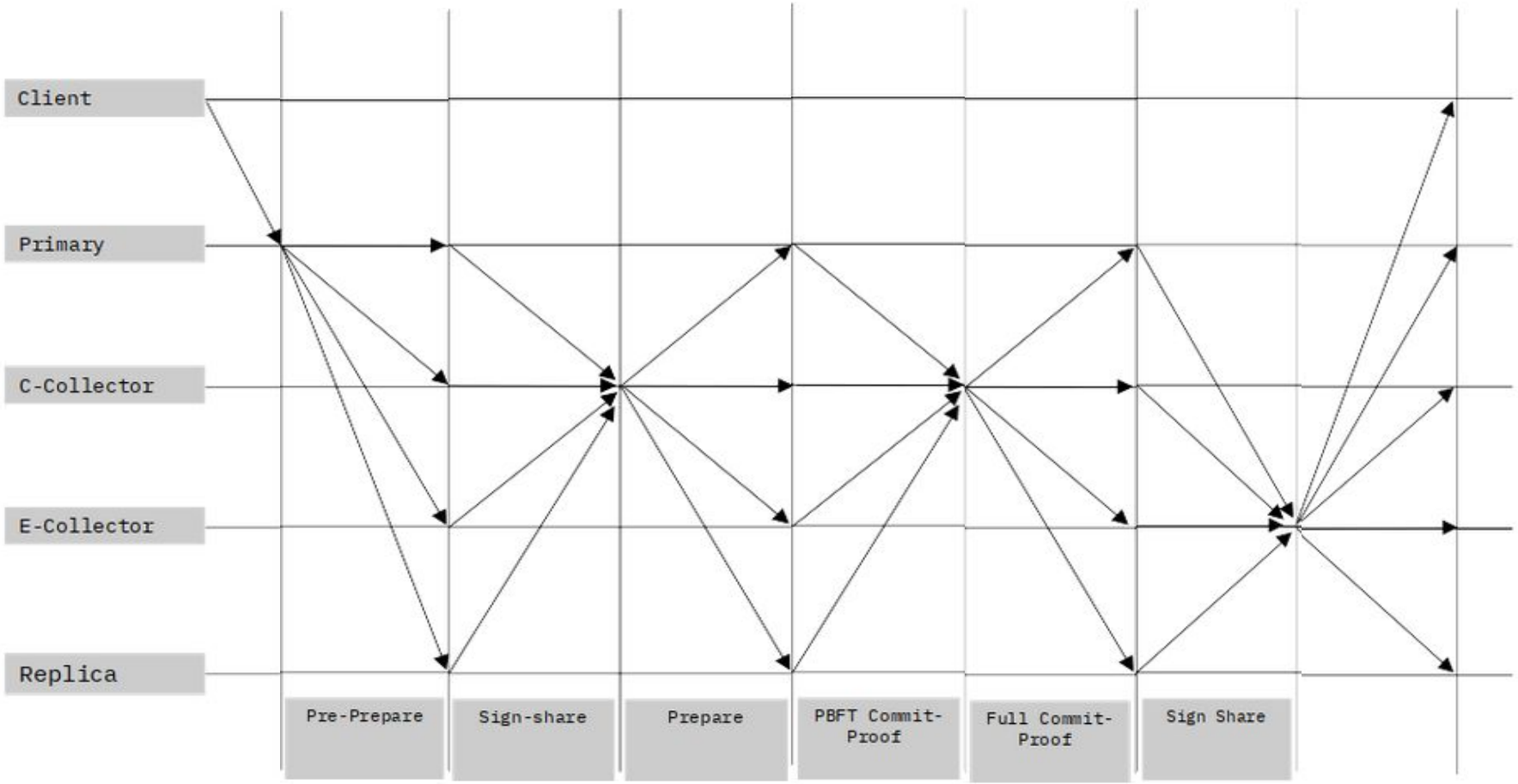
PBFT


C
reply
0
1
2
3

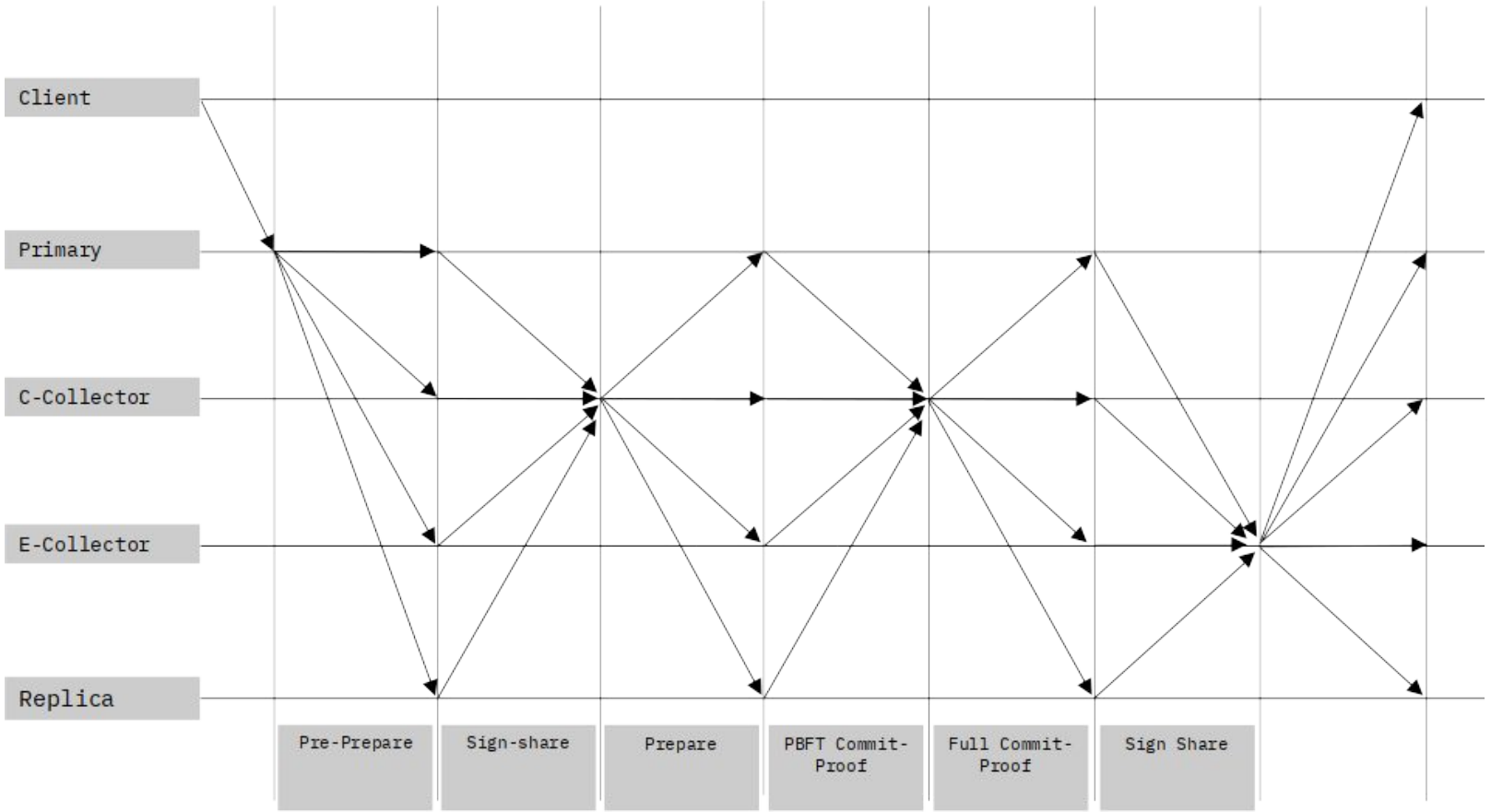
Linear-BFT

- This is the **fall-back protocol** and used when Fast Path is unavailable
- Adaptation of PBFT that uses threshold signatures and linear communication
  - Avoiding all-to-all communication
- Threshold: 2f + c + 1
  - Less than 3f+c+1 but >= 2f+c+1
  - Timer

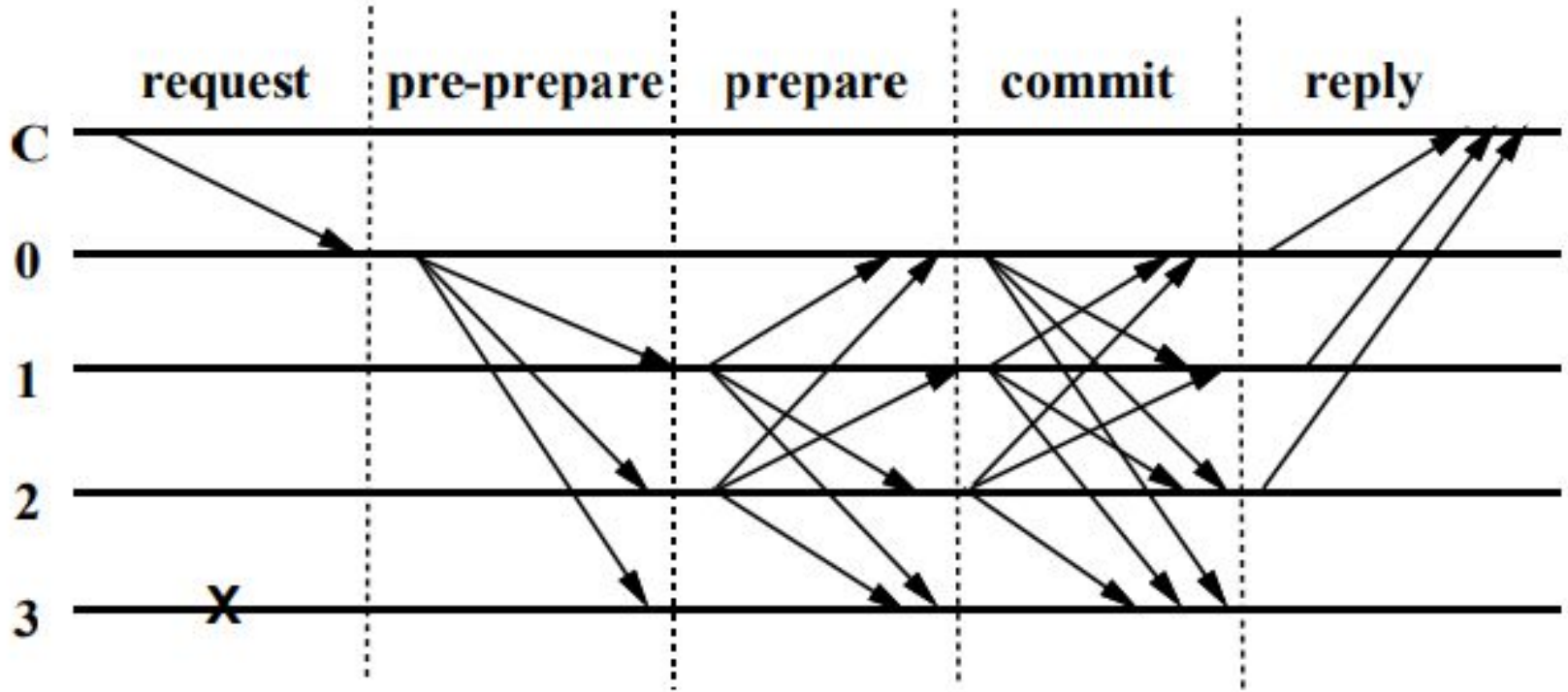
Schematic Message Flow for LBFT



Schematic Message Flow for LBFT



Reference: PBFT



View Change

View change is triggered when the primary is faulty. This can be decided when f+1 replicas complain

Three phases:

1. View-change Phase
2. New-view Phase
3. Accepting a New-view Phase

View-change Phase

Each replica i sends to the new primary of view v+1 the <view-change> message

where v is the current view number.

New-view Phase

The new primary gathers 2 f + 2c + 1 view change messages from replicas. The new primary initiates a new view by sending a set of 2 f +2c+1 view change messages.

Accepting a New-view

When a replica receives a set I of |I| = 2f +2c+1 view change message, it processes slots one by one. For each such slot, a replica either decides it can commit a value, or it adopts it as a pre-prepare by the new primary, according to the algorithm:

IF a full-commit-proof accepted :
THEN DO: commit a value
ELSE:
THEN DO: adopts a safe value

Safety and Liveness

SBFT is safe because the view change protocol takes the maximum view over both fast-path and linear-PBFT path.

SBFT lacks liveness in asynchronous mode due to FLP [3]. In the synchronous mode, liveness is obtained by striking a balance between making progress in the current view and moving to a new view.

Performance Evaluation

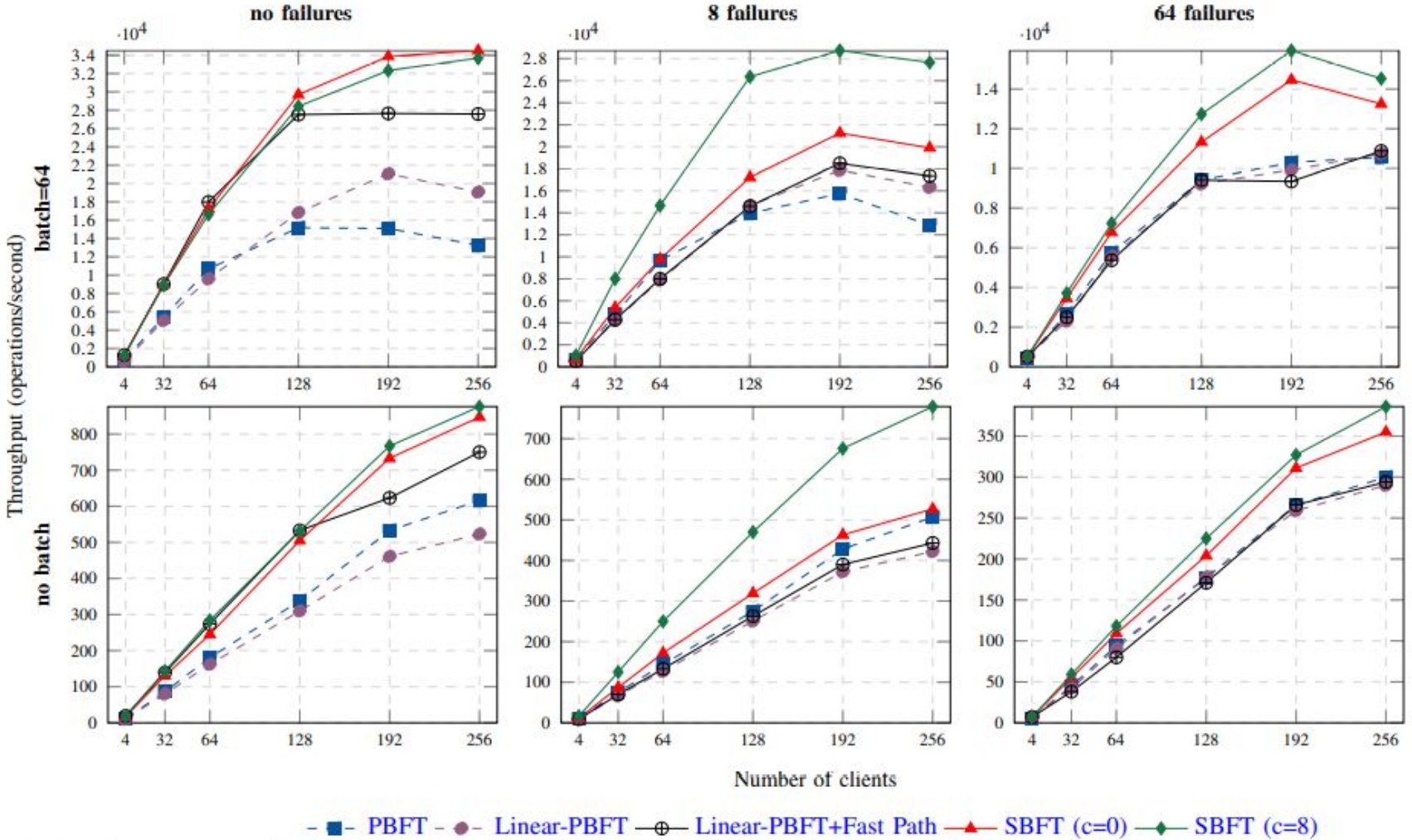


Fig. 2. Throughput per clients for key-value store benchmark on Continent scale WAN.

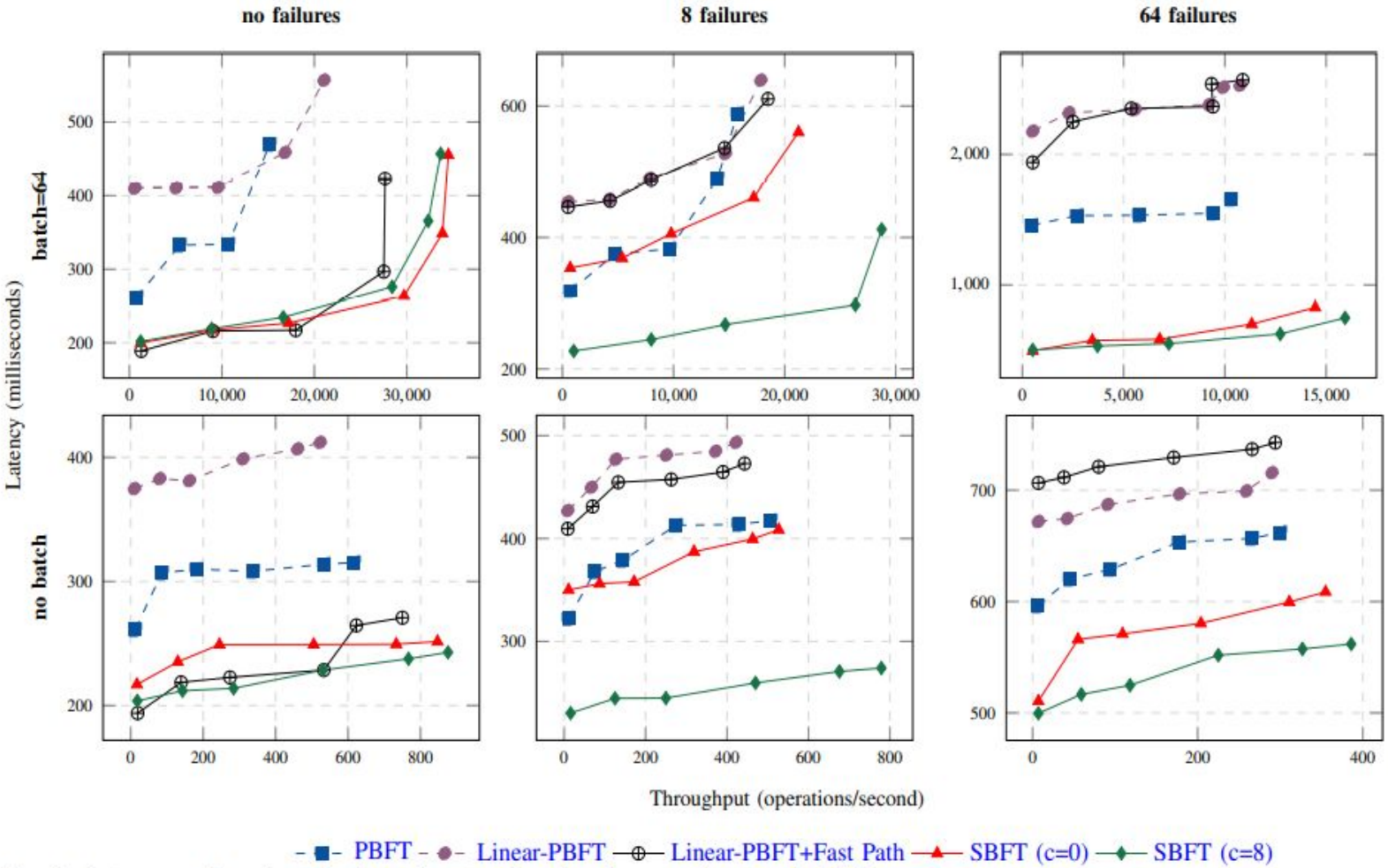


Fig. 3. Latency vs throughput for key-value store benchmark on continent scale WAN.

Thank you

References

[1] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin G¨un Sirer. Decentralization in bitcoin and ethereum networks. Financial Crypto, 2018.

[2] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, OSDI ’99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[3] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. J. ACM,
32(2):374–382, April 1985.