Separation is Good: A Faster Order-Fairness Byzantine Consensus

Ke Mu
Southern University of Science and Technology

Bo Yin
Changsha University of Science and Technology

Alia Asheralieva
Loughborough University

Xuetao Wei*
Southern University of Science and Technology

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

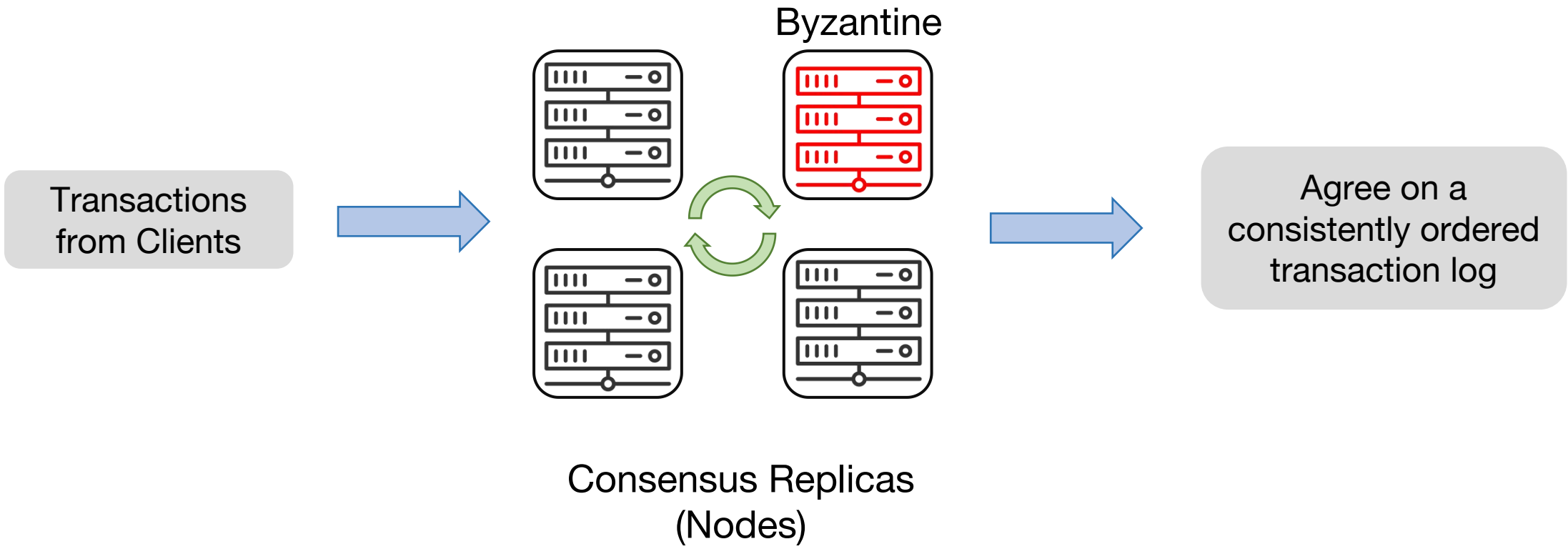
NDSS 2024

Outline

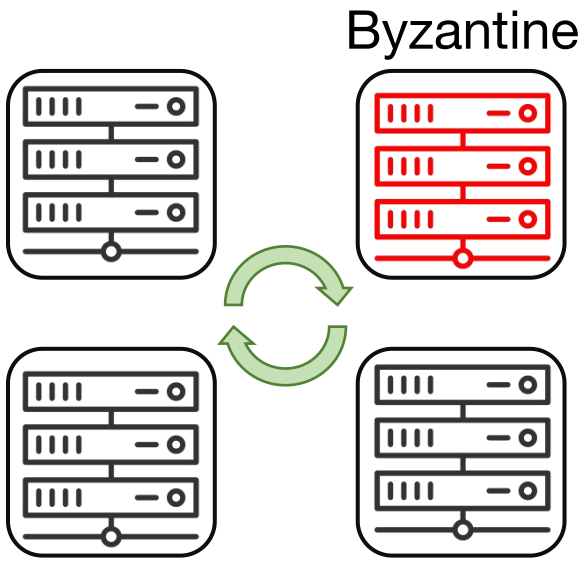
- Background
- Motivation
- Our Solution: SpeedyFair
- Performance Results
- Conclusion

Background

Byzantine Fault Tolerant (BFT) Consensus



Background



BFT Consensus

Safety or **Consistency**

Correct replicas output the same sequence of transactions

Liveness

Valid transactions are eventually delivered in a reasonable time

However

No restrictions on how to order or which order is chosen

Transaction Order Manipulation Problem

Background

- Transaction ordering is crucial in **decentralized finance (DeFi)**
  - Adversarial transaction ordering can cause **unexpected financial losses** for users, systematic **bribery**, or even **protocol instability**

- Currently, most permissioned BFT consensus protocols are leader-based:
  - PBFT, Hotstuff, etc.

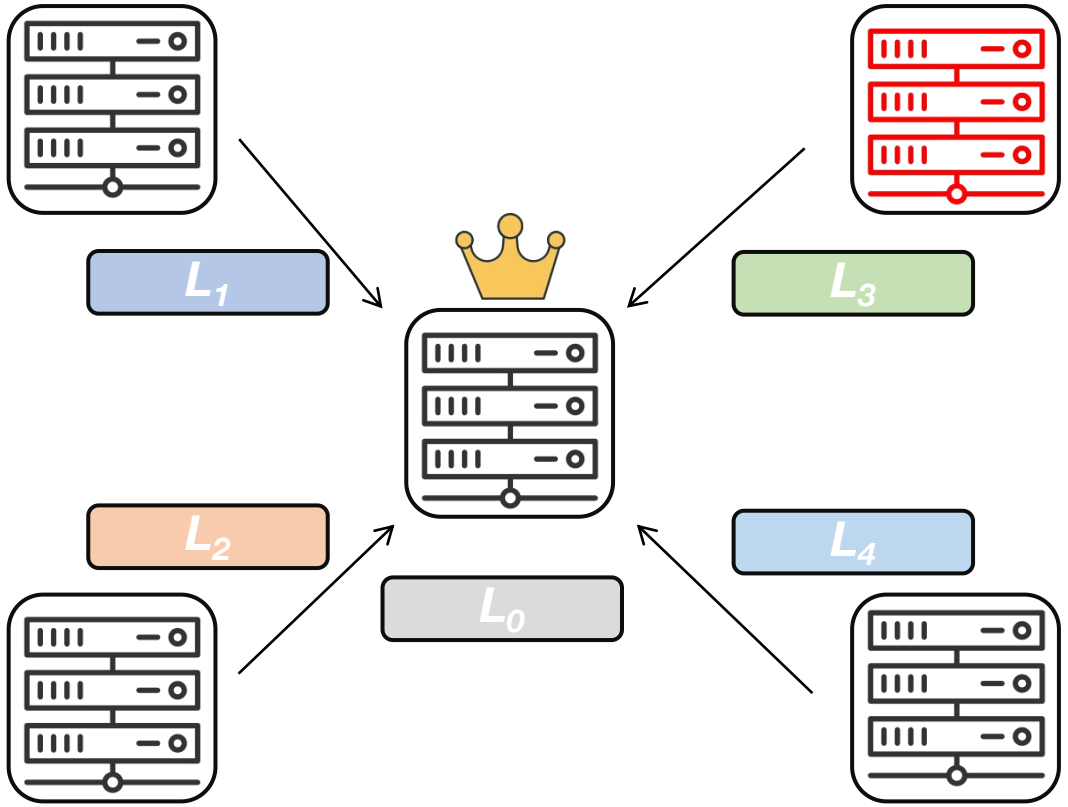
- Leader node **can easily control the transaction ordering**:
  - Adversarial leaders can arbitrarily manipulate ordering without violating safety or liveness
  - Lack of scheme checking fairness in consensus protocol

- Recent works introduce a new property called order-fairness in BFT consensus to prevent adversarial order manipulation

  [Kelkar et al, Crypto, 2020; Kursawe et al, AFT, 2020; Cachin et al, FC, 2022, Kelkar et al, CCS, 2023]

Motivation

Leader-based order-fairness consensus

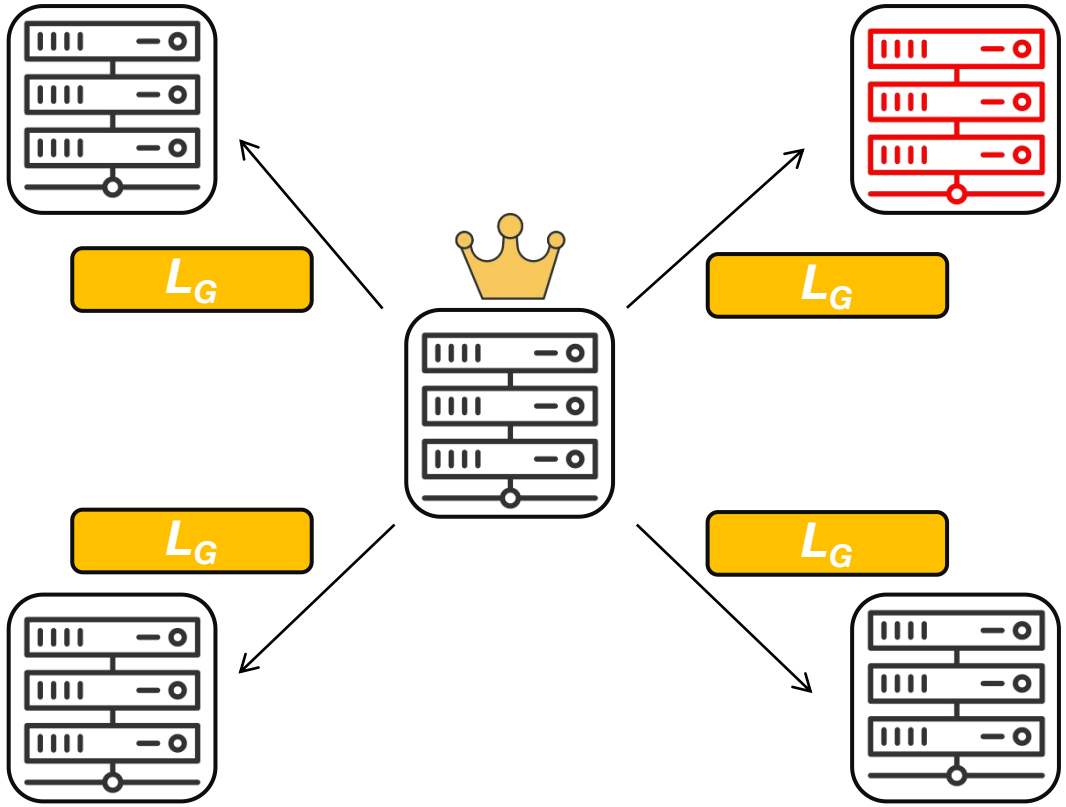


(i) Collect Local Ordering

- Replicas send local orderings to the leader

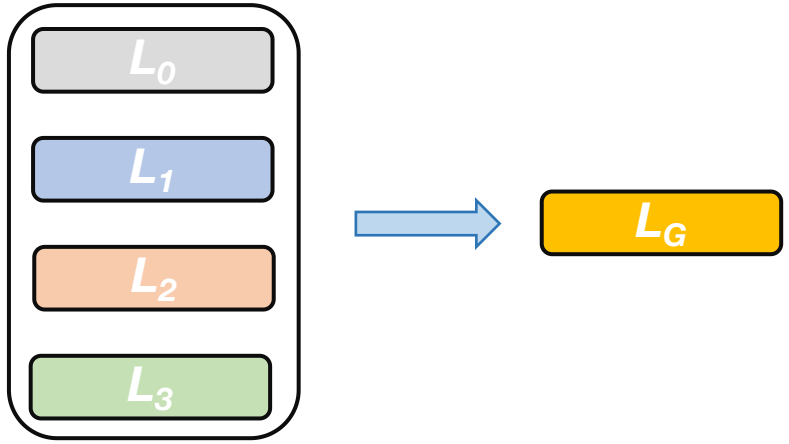
Motivation

Leader-based order-fairness consensus



(ii) Fair Ordering & Propose

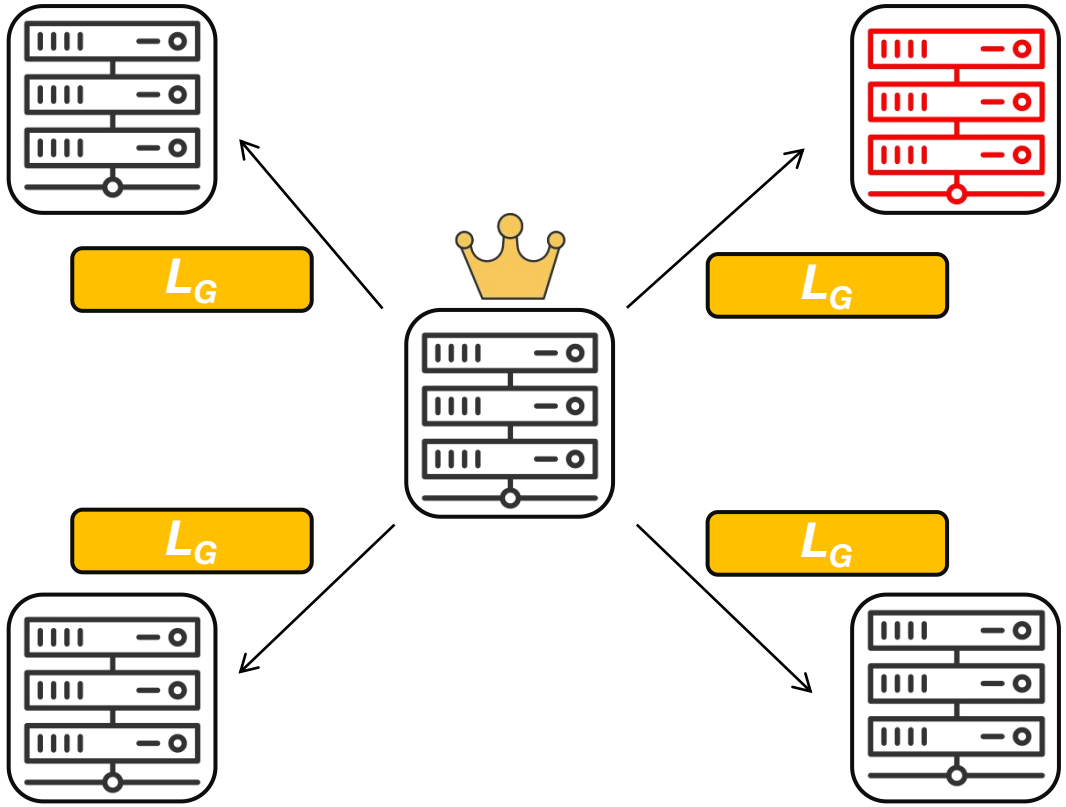
- Leader computes a fair global ordering as a new fair proposal through local orderings
- Leader proposes fair proposal



Fair Ordering Algorithm

Motivation

Leader-based order-fairness consensus



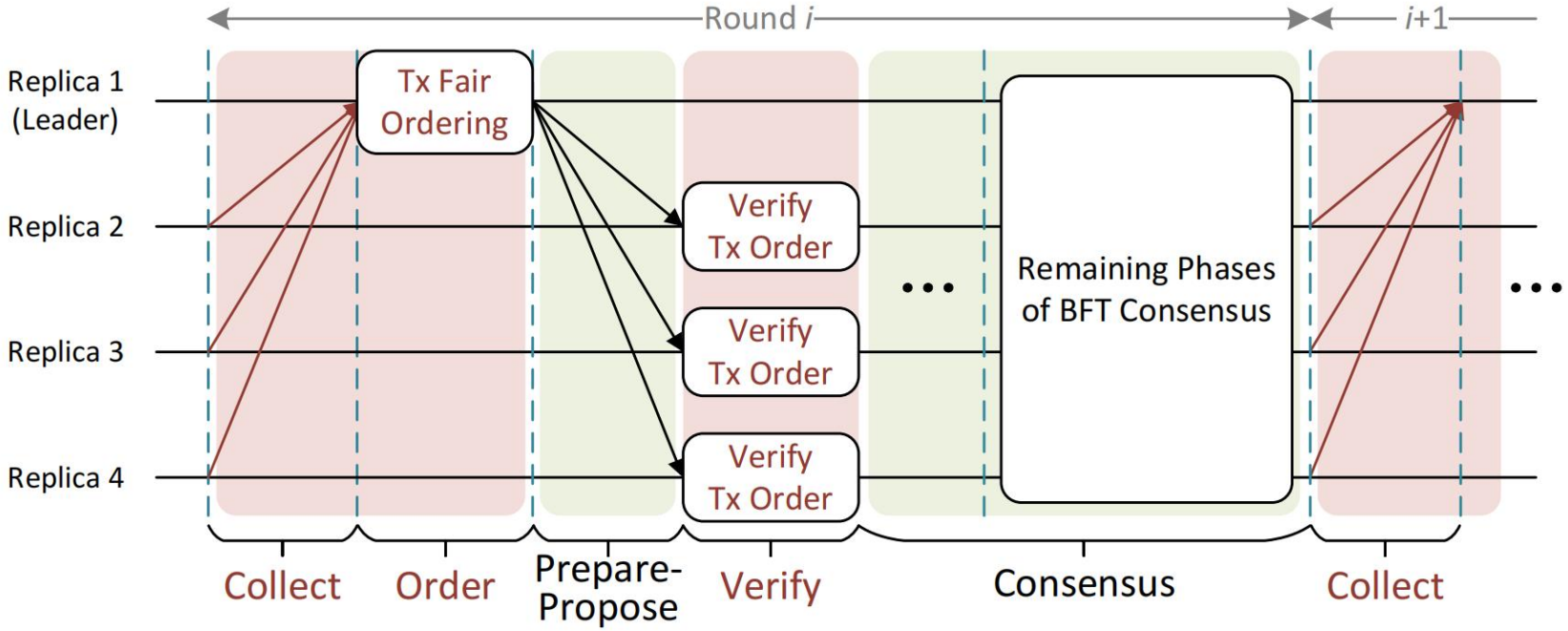
(iii) Verify & Consensus

- Replicas verify the fairness of the proposal by recalculating fair ordering
- If verified, replicas perform the remaining phases of the consensus protocol

Motivation

Current leader-based order-fairness protocols are not ideal

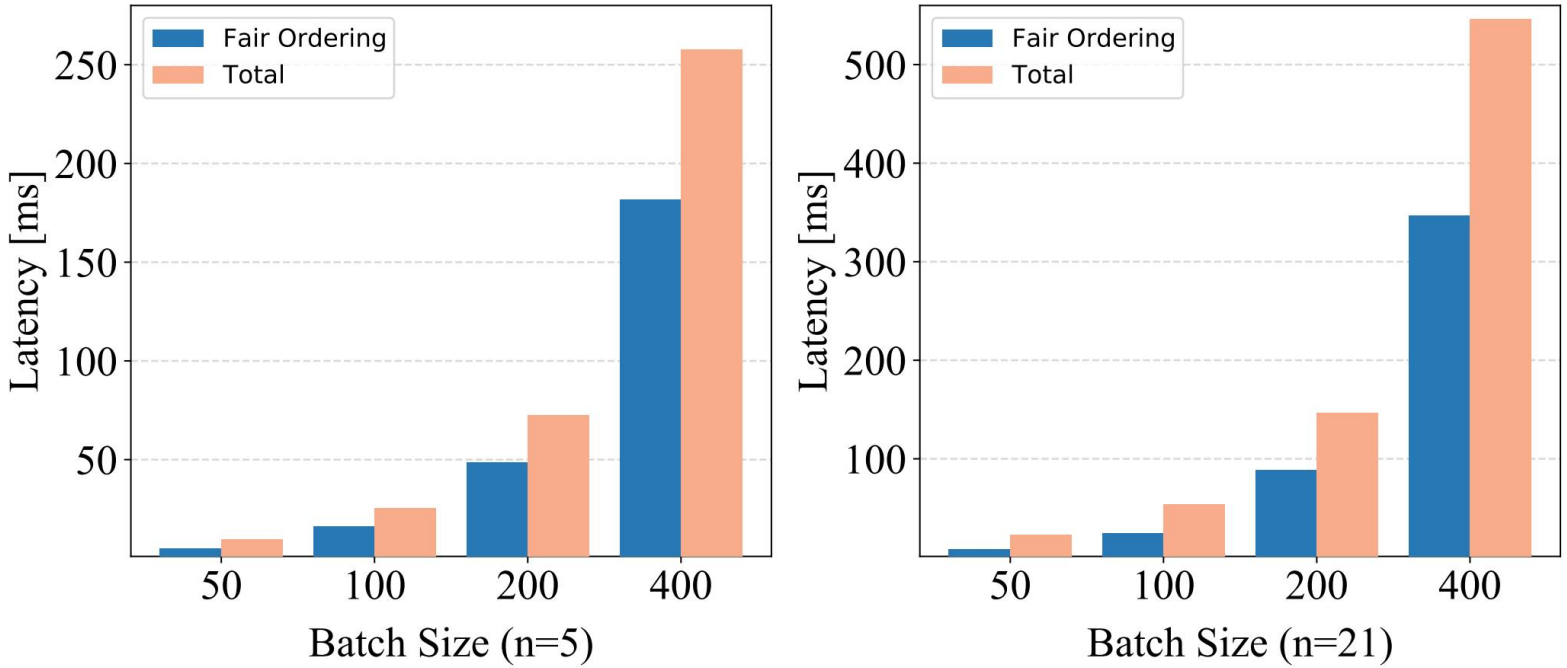
- **(i) Strongly Coupled Consensus and Fair Ordering**
  - Mutual waiting between consensus and ordering



Execution flow of a round in a leader-based order-fairness consensus protocol

Motivation

- **(ii) Expensive Fair Ordering**
  - Fair ordering and verifying accounts for 35%-70% latency (increase with batch size)



Compare the latency of fair ordering and total consensus in Themis [Kelkar et al, CCS, 2023]

Our Solution: SpeedyFair

Key Observations

- (i) Fair ordering does not rely on the transaction execution results of the previous proposal after the consensus, but only relies on the previous transaction order
- (ii) Ordering is computationally intensive. Consensus is communication intensive.

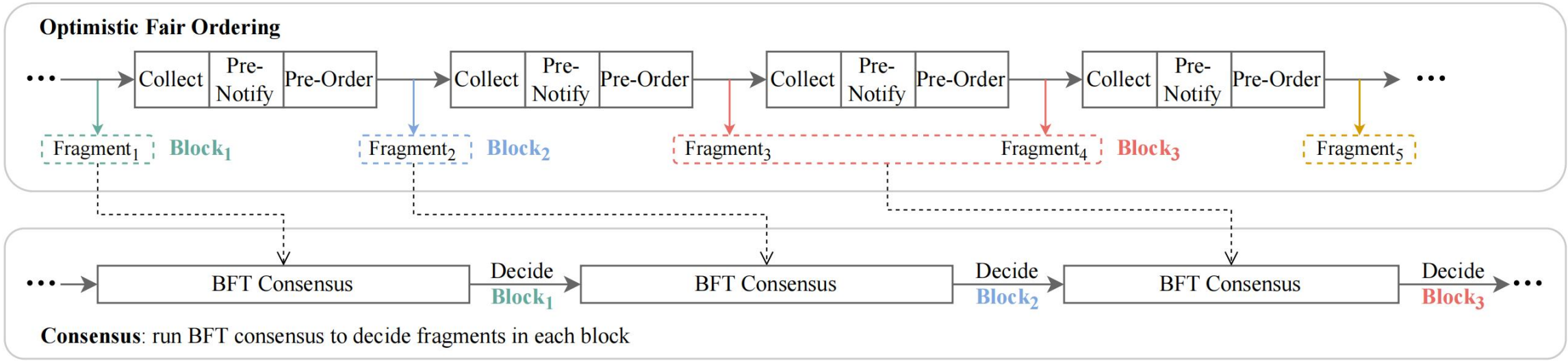
Main Idea

- Separate fair ordering and verification from the critical path of consensus
- Optimistic Fair Ordering (OFO): execute fair ordering independently and continuously
- Minimal modification in BFT consensus: agree on valid ordering results from OFO

Our Solution: SpeedyFair

SpeedyFair Architecture

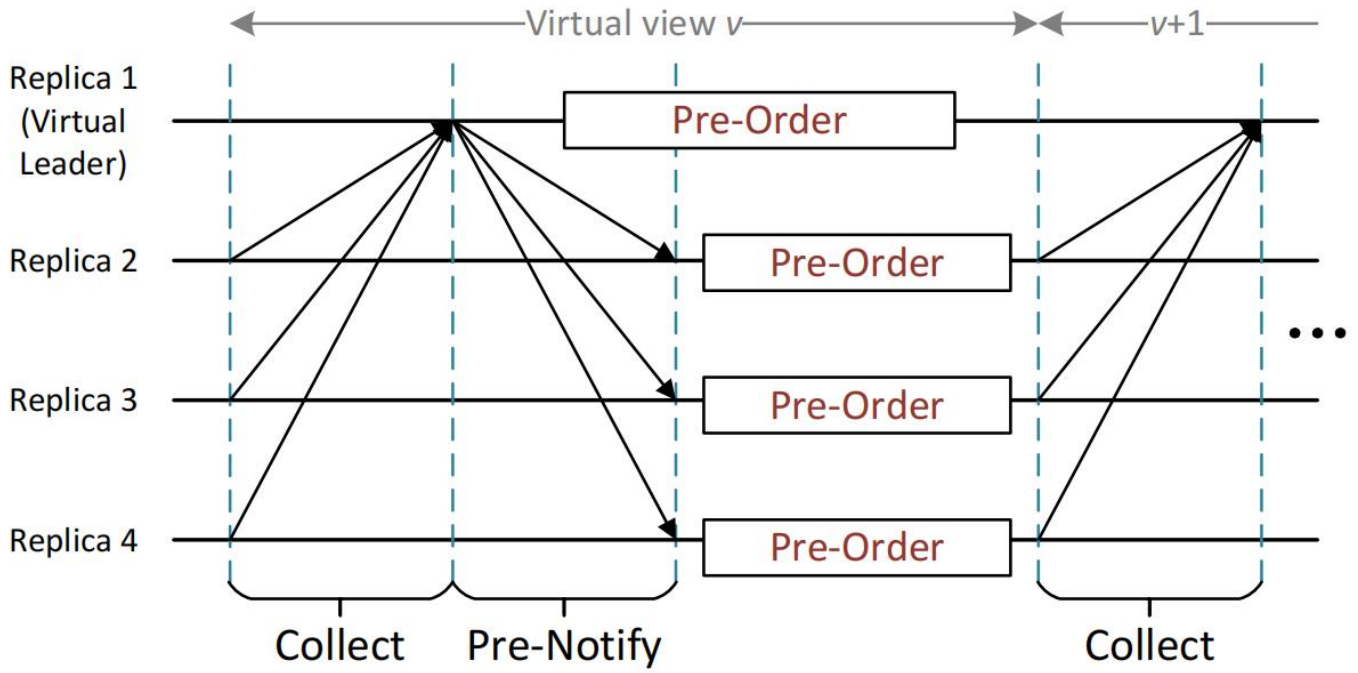
- Optimistic Fair Ordering (OFO)
  - Fair ordering performs individually and consecutively without waiting for consensus
  - Saving valid fair ordering outputs in fragments
- Modified BFT Consensus



Our Solution: SpeedyFair

Optimistic Fair ordering (OFO)

- Parallelize fair ordering and verification in Pre-Order
- Design a quorum certificate using threshold signatures to guarantee that the valid fair ordering outputs (fragment) of OFO can be eventually selected by consensus



Our Solution: SpeedyFair

Modified BFT Consensus

- **Minimal Modification** on prepare phase in consensus

- Prepare Proposal
  - Leader appends valid fragments (with quorum certificate) generated by OFO into the newest fair proposal

- Verify Proposal
  - Replicas verify the proposal with a simple verification function
  - The simple verification function only checks if the fragment in the proposal is the same as the fragment calculated by OFO (no need to recalculate the fair ordering)

Our Solution: SpeedyFair

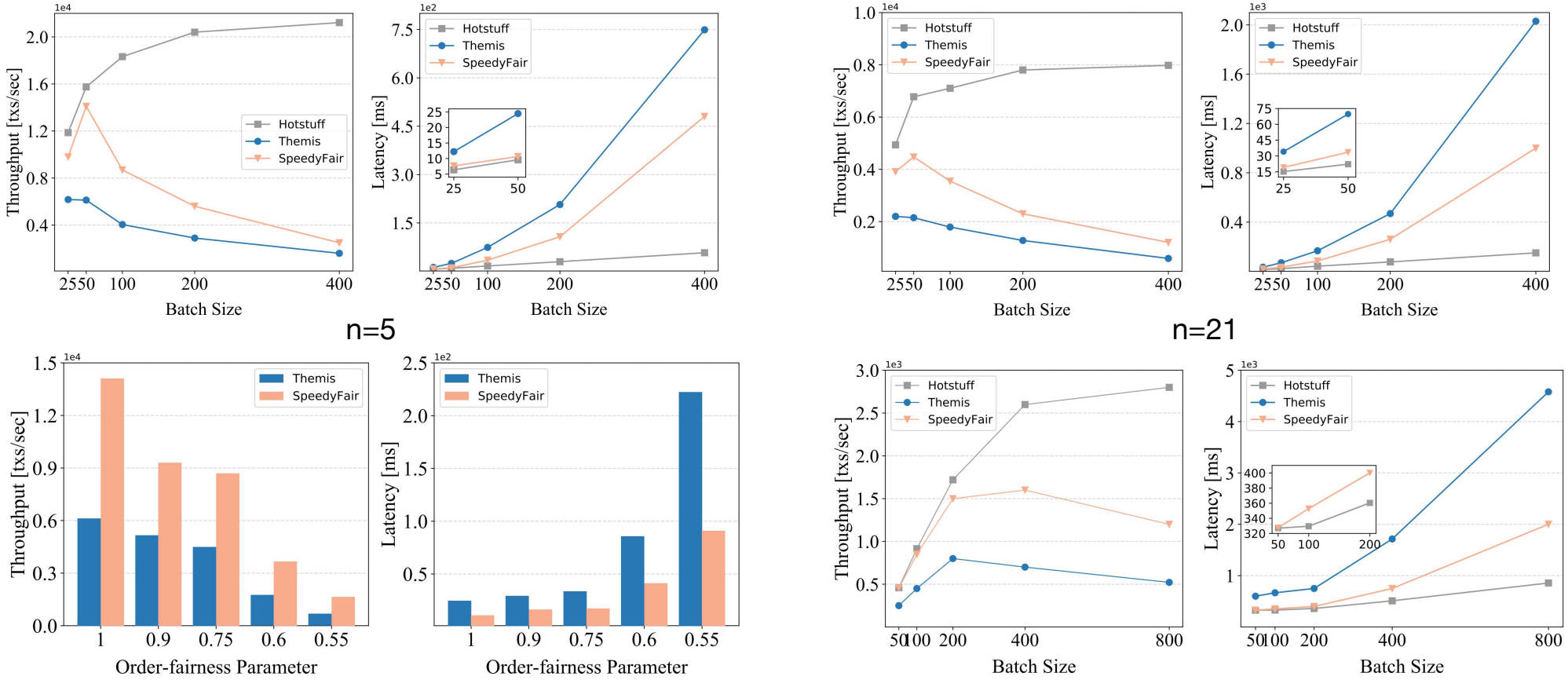
Getting rid of the liveness and data availability problem

- Liveness problem (in OFO):
  - OFO process can be blocked by malicious leaders or long network latency (timeout)
- Pacemaker Mechanism
  - Advance the view in OFO when a timeout occurs

- Data availability problem:
  - Slow replicas may not completely compute expensive fair ordering in OFO locally, causing delays in both OFO and verification in consensus
- Data Synchronization Mechanism
  - Synchronous valid fragments through quorum certificate (QC)
  - QC proves at least *f+1* replicas have computed the same fair-ordered fragments

Performance Results

- SpeedyFair outperforms Themis
- Comparable performance with Hotstuff baseline when batch size is relatively small



Conclusion

- Design **SpeedyFair**, a high-performance order-fairness BFT protocol.
- Decouples fair ordering from consensus to reduce delays waiting for each other.
- Supports parallel ordering and verifying.
- Prevents liveness, data availability issues in decoupled paradigm.

Thank you!