41st IEEE International Conference on Data Engineering

HONG KONG SAR, CHINA | MAY 19 – 23, 2025

Orthrus: Accelerating Multi-BFT Consensus through Concurrent Partial Ordering of Transaction

Hanzheng Lyu[1], Shaokang Xie[2], **Jianyu Niu[3]**, Ivan Beschastnikh[1] , Yinqian Zhang[3], Mohammad Sadoghi[2], and Chen Feng[1]

[1]University of British Columbia, Canada

[2]University of California, Davis, USA

[3]Southern University of Science and Technology, China

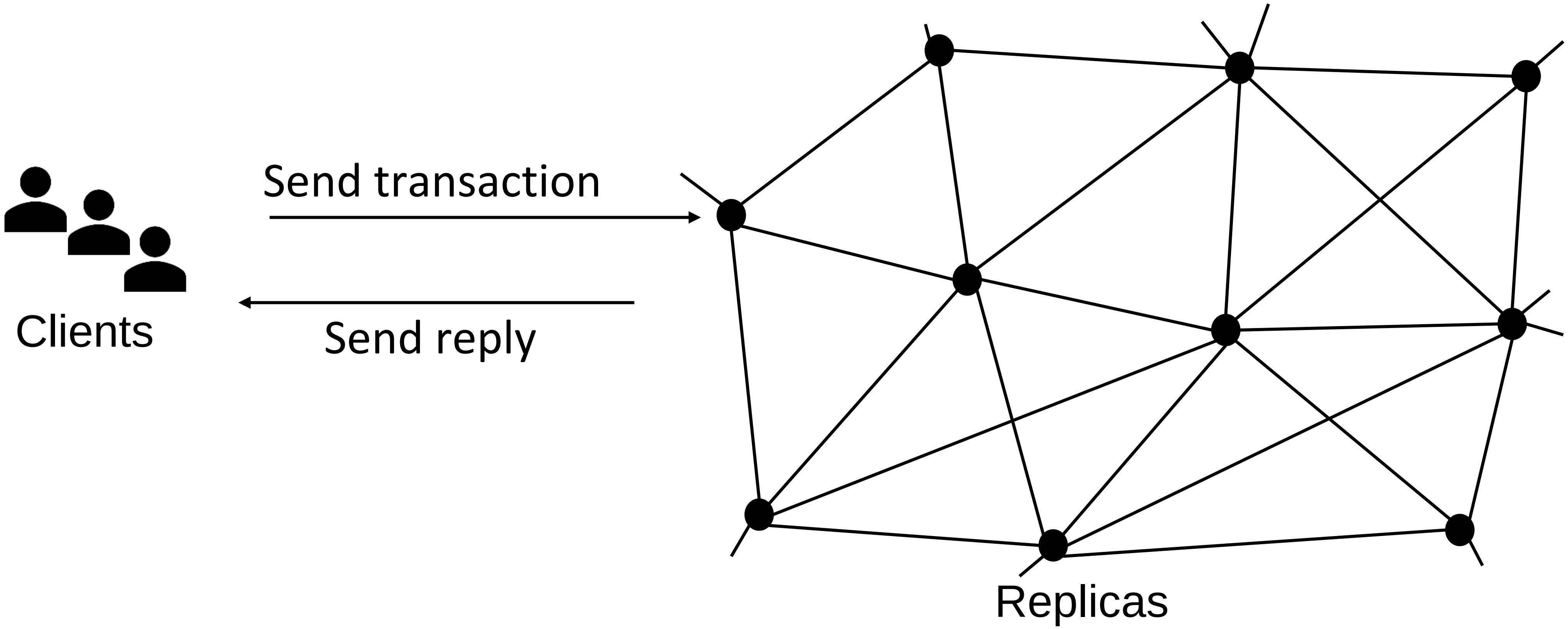
THE UNIVERSITY OF BRITISH COLUMBIA

UC DAVIS

SUSTech Southern University of Science and Technology

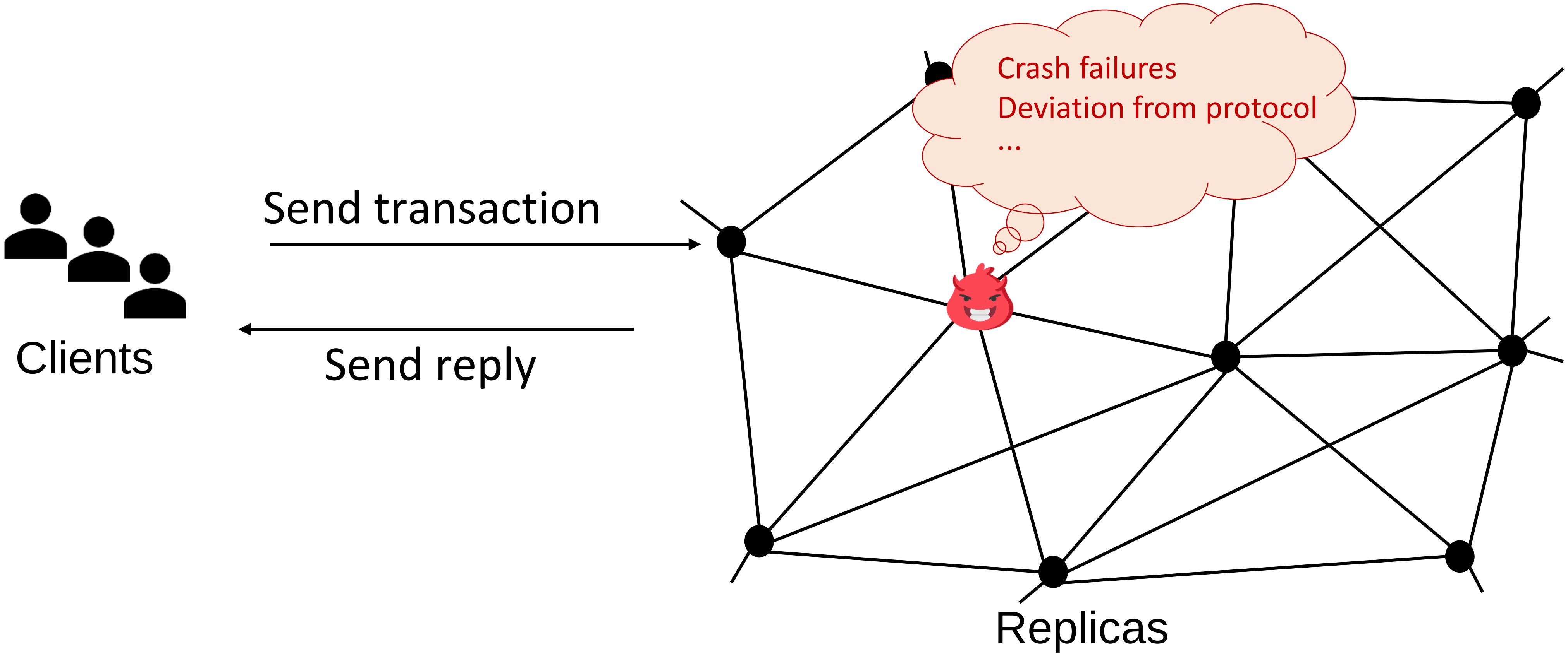
BFT Consensus

Byzantine Fault Tolerant (BFT) Consensus allows a decentralized system to reach agreement among replicas despite Byzantine failures.



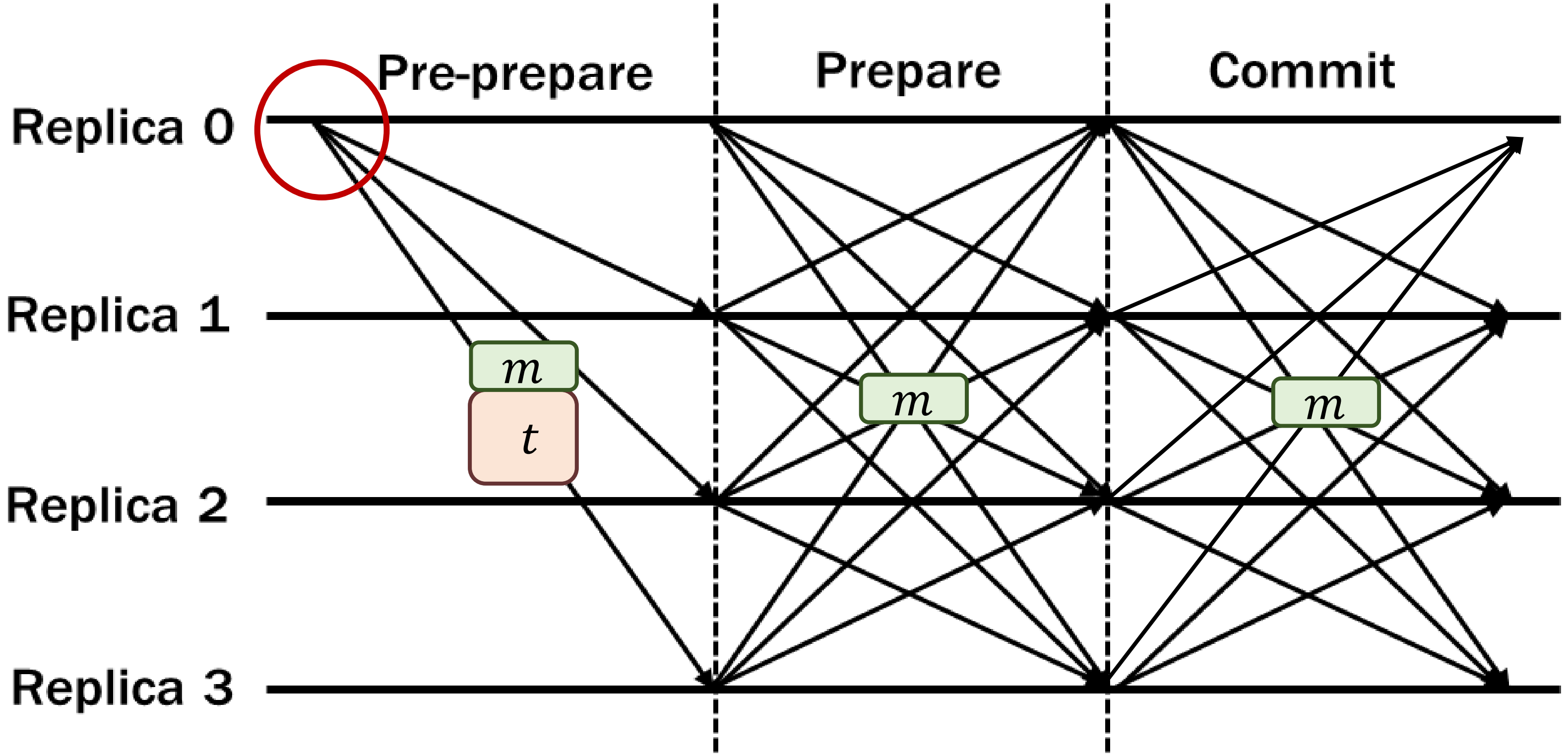
BFT Consensus

Byzantine Fault Tolerant (BFT) Consensus allows a decentralized system to reach agreement among replicas despite Byzantine failures.



Leader-based BFT[1]

The leader limits the overall performance of the BFT protocol.

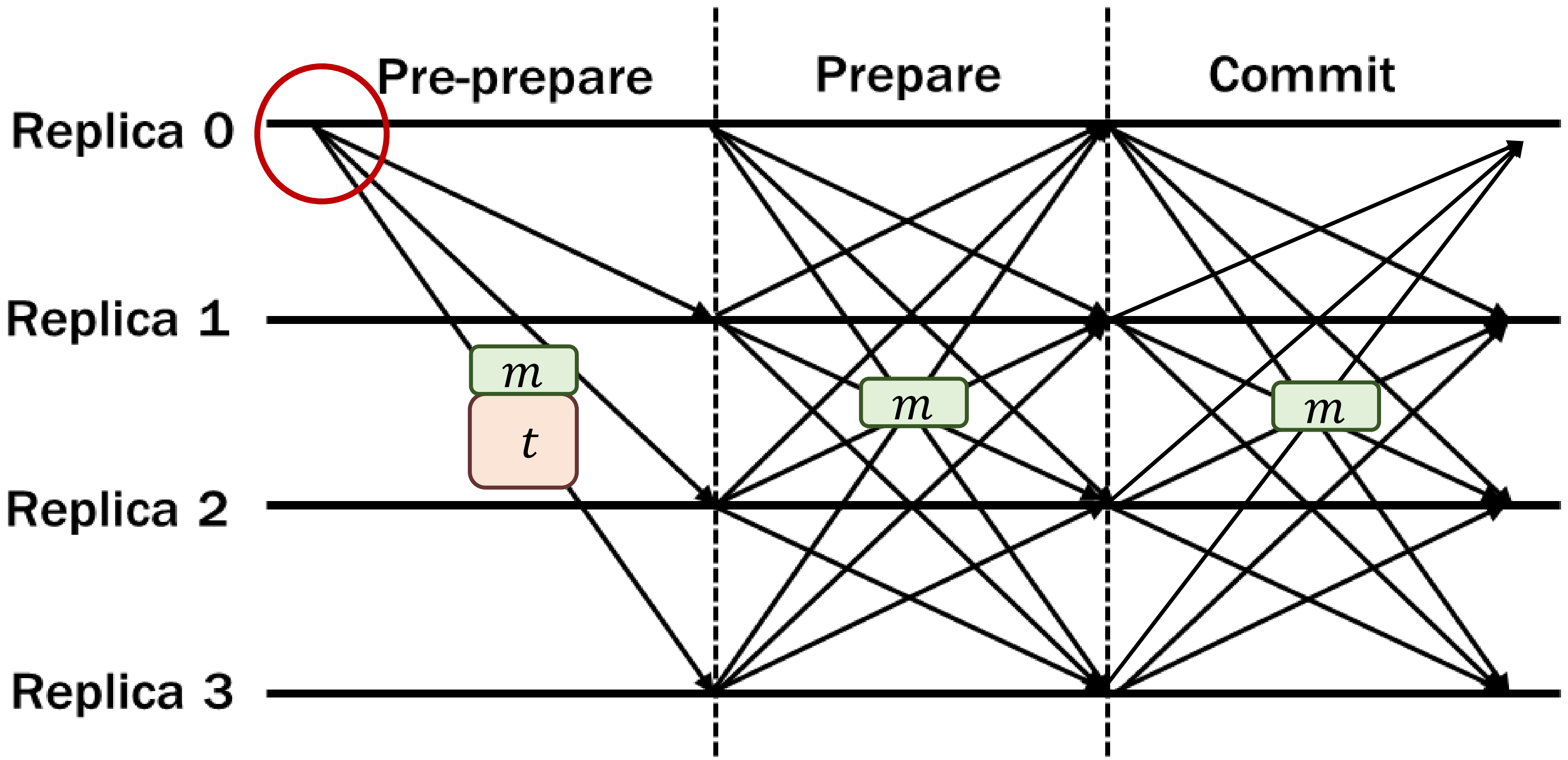


Practical Byzantine Fault Tolerance (PBFT) Protocol

[1] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In OSDI, 1999.

Leader-based BFT[1]

The leader limits the overall performance of the BFT protocol.



Practical Byzantine Fault Tolerance (PBFT) Protocol

$$\boldsymbol{size(t) >> size(m)}$$

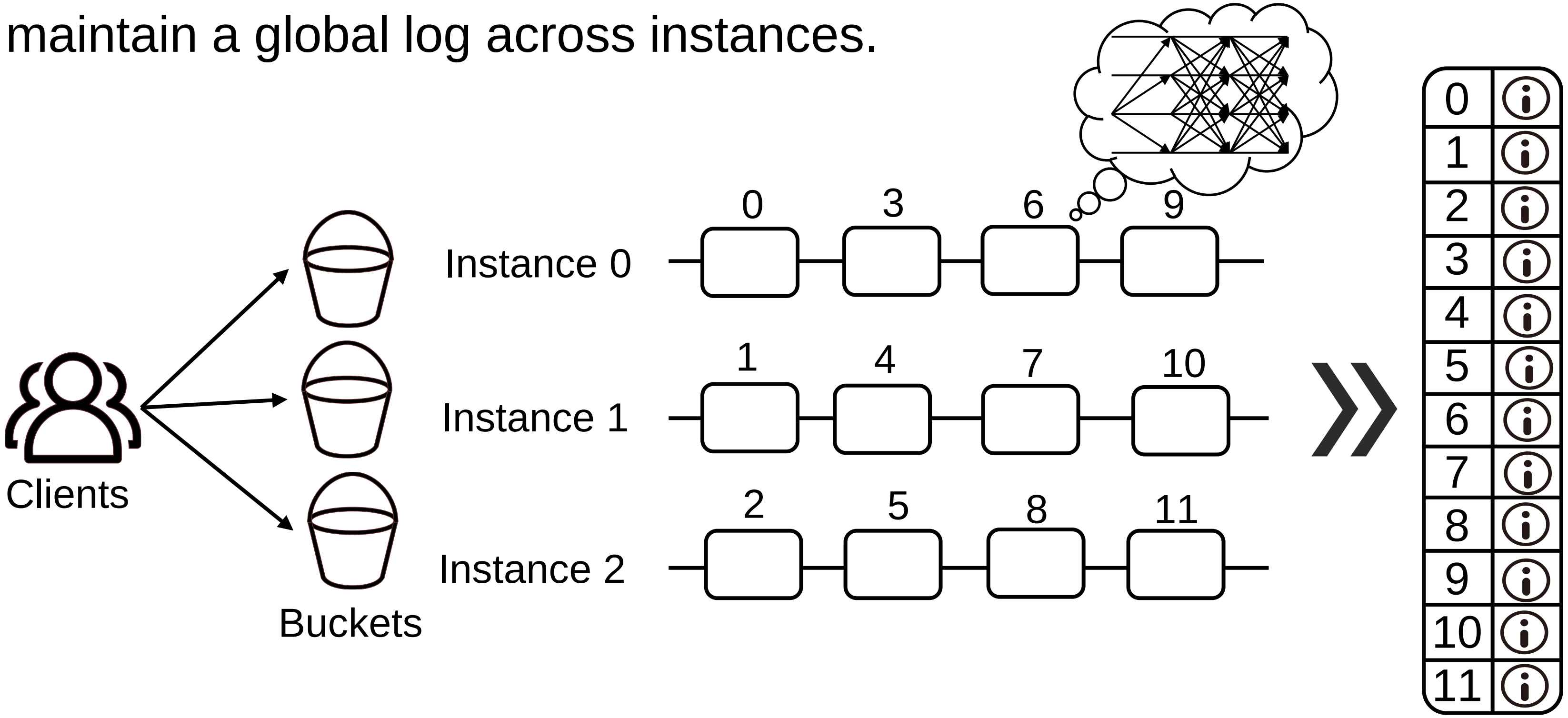
$$T_{\max} \approx \frac{B}{(n-1)size(t)}$$

- $T_{\max}$: Maximum throughput
- B: Leader's bandwidth
- n: number of replicas

[1] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In OSDI, 1999.

Multi-BFT Consensus[1-3]

- Each replica acts as the leader of one instance and backups of other instances.
- Each instance independently outputs a sequence of partially committed blocks.
- All replicas maintain a global log across instances.

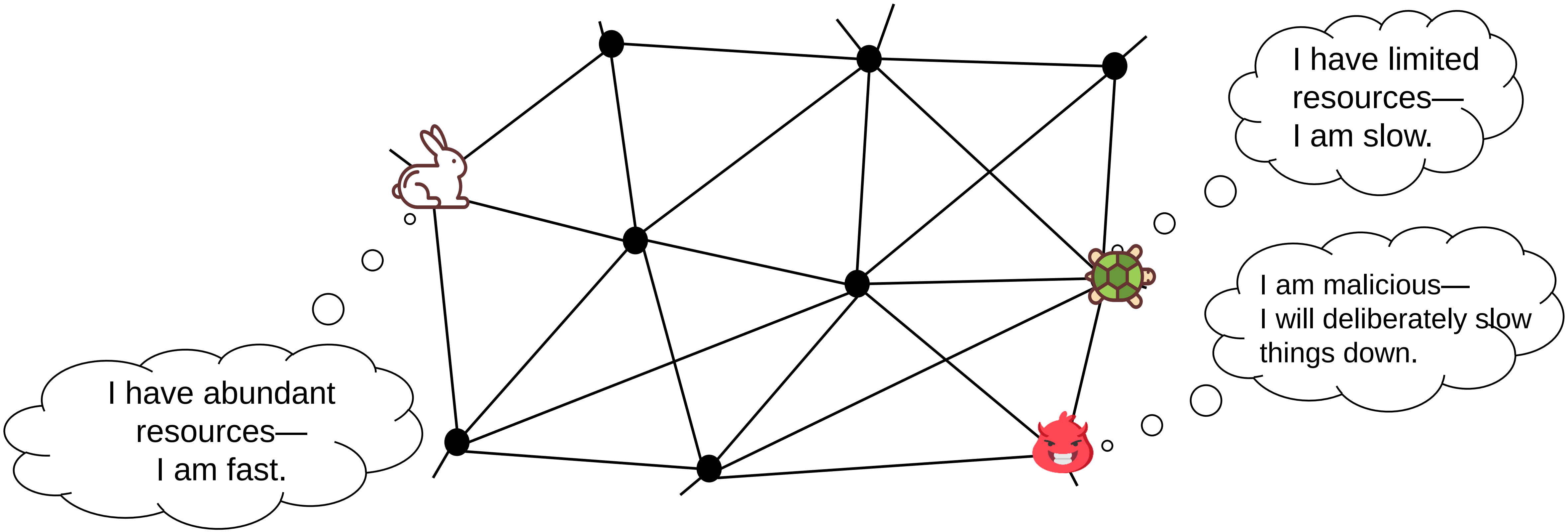


[1] Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolić. State Machine Replication Scalability Made Simple, in ACM EuroSys'22.
[2] Chrysoula Stathakopoulou, Tudor David, and Marko Vukolic. Mir BFT: Scalable and Robust BFT for Decentralized Networks, in Jsys'22.
[3] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. RCC: Resilient Concurrent Consensus for High-Throughput Secure Transaction Processing, in IEEE ICDE'21.

Stragglers in Network

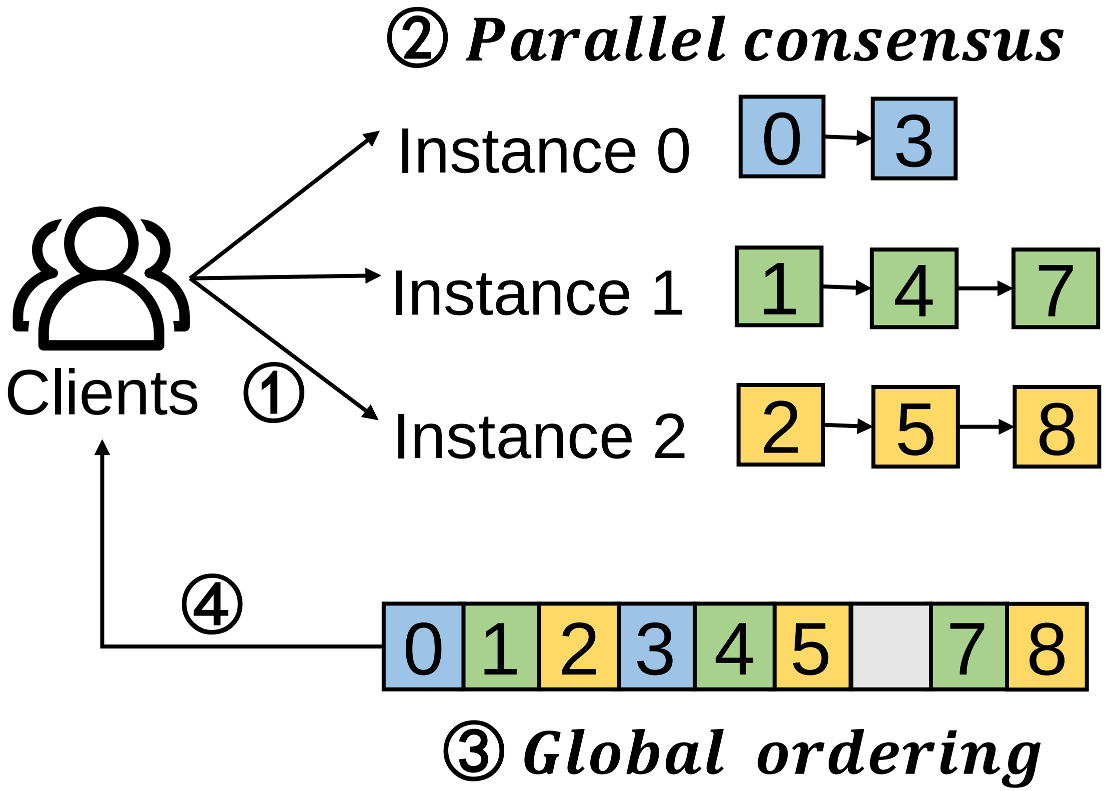
Stragglers are slow replicas which may be significantly slower than others due to:

- Limited computational resources and network bandwidth
- Malicious behavior, where Byzantine replicas intentionally delay their behaviors

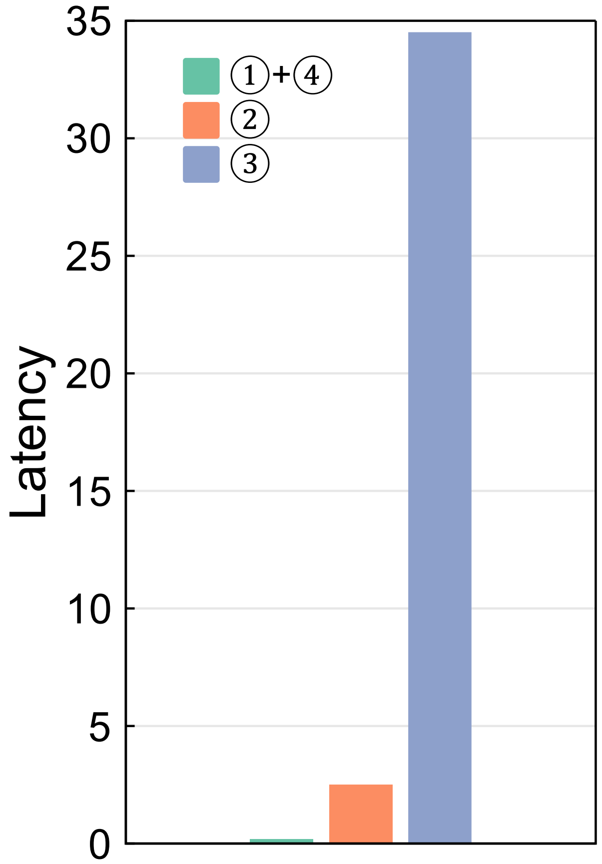


The Impact of Stragglers

Performance degradation of pre-determined global ordering with stragglers



(a) Multi-BFT paradigm



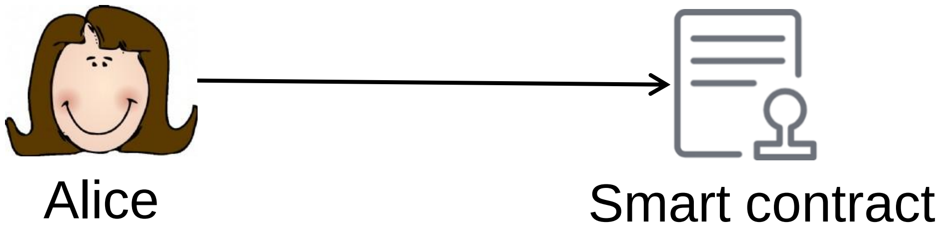
(b) Breakdown latency

Key Observations

There are two primary classes of transactions in blockchain-based financial applications: payments and smart contracts.

There are two types of operations in payment transactions:

- incremental (e.g., adding funds)
- decremental (e.g., withdrawing funds)

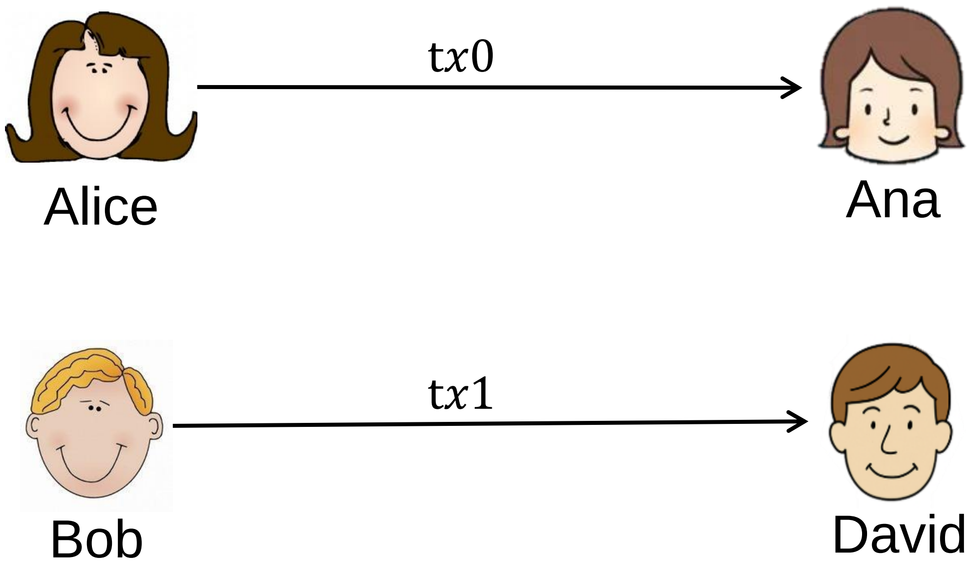


Smart contracts often involve complex, non-commutative operations (e.g., assignments)

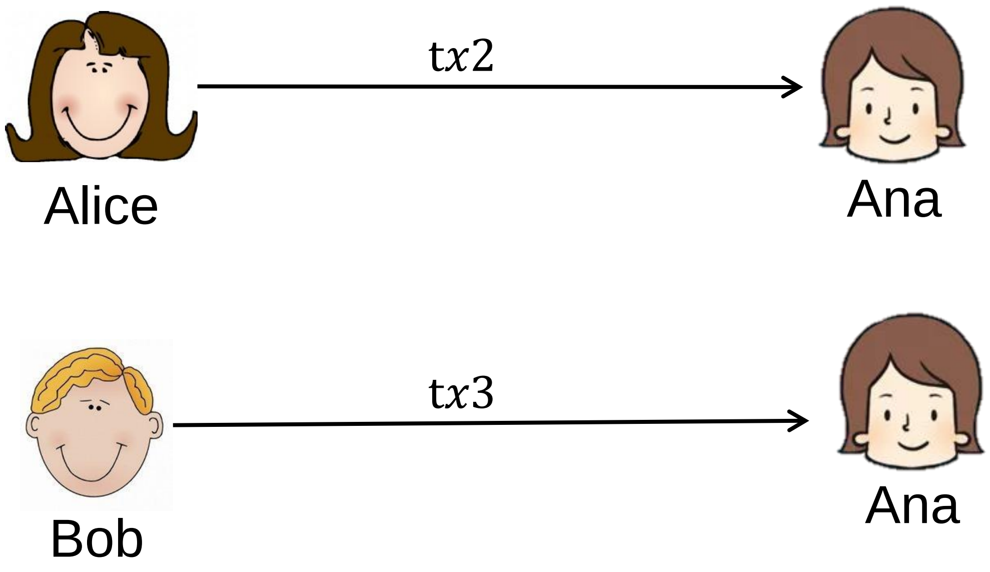
Key Observations

THE UNIVERSITY OF BRITISH COLUMBIA

Observation 1: If two payment transactions have different payers and neither transaction affects the balance of the other's payer, they can be executed concurrently.



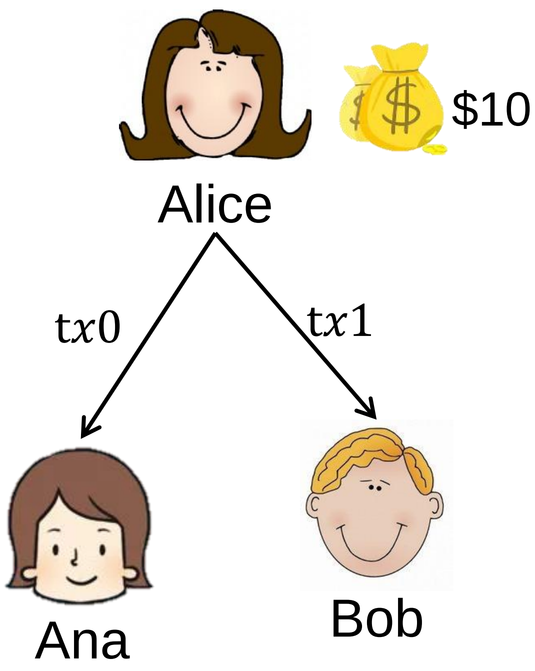
$tx0, tx1$ *can be executed concurrently*



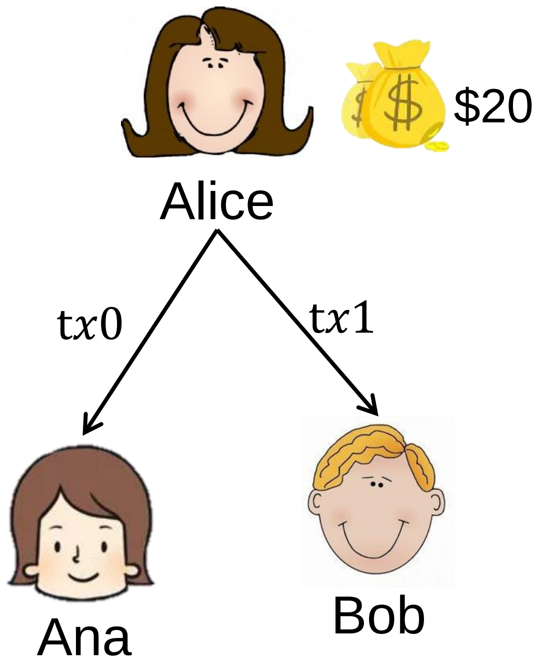
$tx2, tx3$ *can be executed concurrently*

Key Observations

Observation 2: If one payment transaction may affect the balance of the other payment transaction's payer, they may need to be executed sequentially.



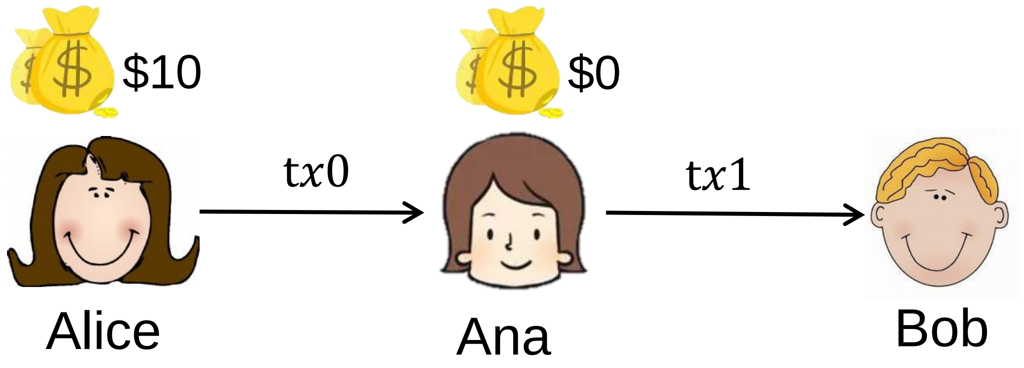
Alice has $10
$tx0$: Alice→Ana $10
$tx1$: Alice→Bob $10
$tx0 - tx1: tx1\ failed$
$tx1 - tx0: tx0\ failed$



Alice has $20
$tx0$: Alice→Ana $10
$tx1$: Alice→Bob $10
$tx0 - tx1: all\ txs\ success$
$tx1 - tx0: all\ txs\ success$

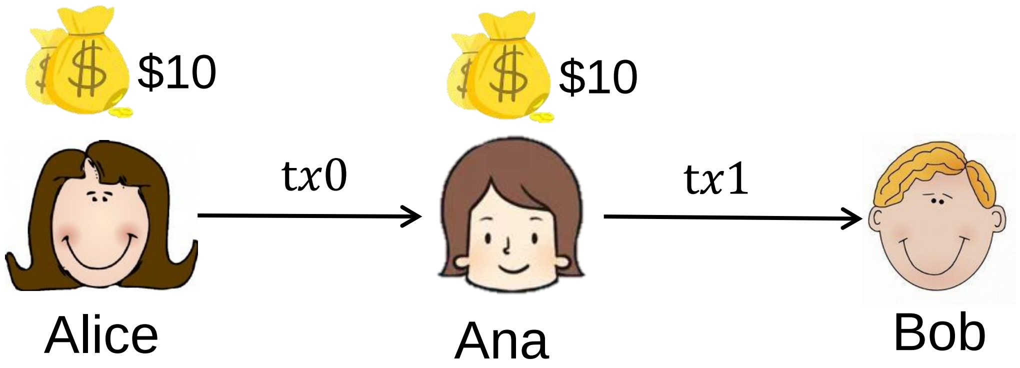
Key Observations

Observation 2: If one payment transaction may affect the balance of the other payment transaction's payer, they may need to be executed sequentially.



Alice has $10, Ana has $0
$tx0$: Alice→Ana $10
$tx1$: Ana→Bob $10

$tx0 - tx1: all\ txs\ success$
$tx1 - tx0: tx1\ failed$

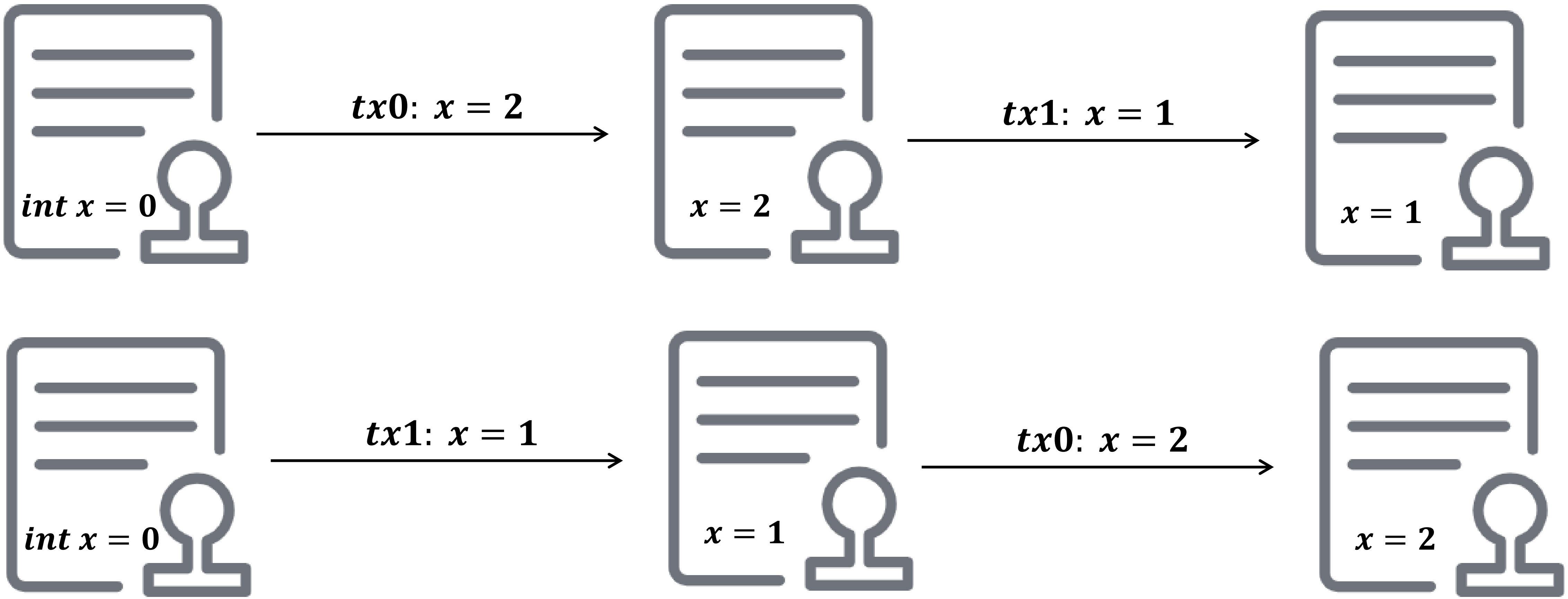


Alice has $10, Ana has $10
$tx0$: Alice→Ana $10
$tx1$: Ana→Bob $10

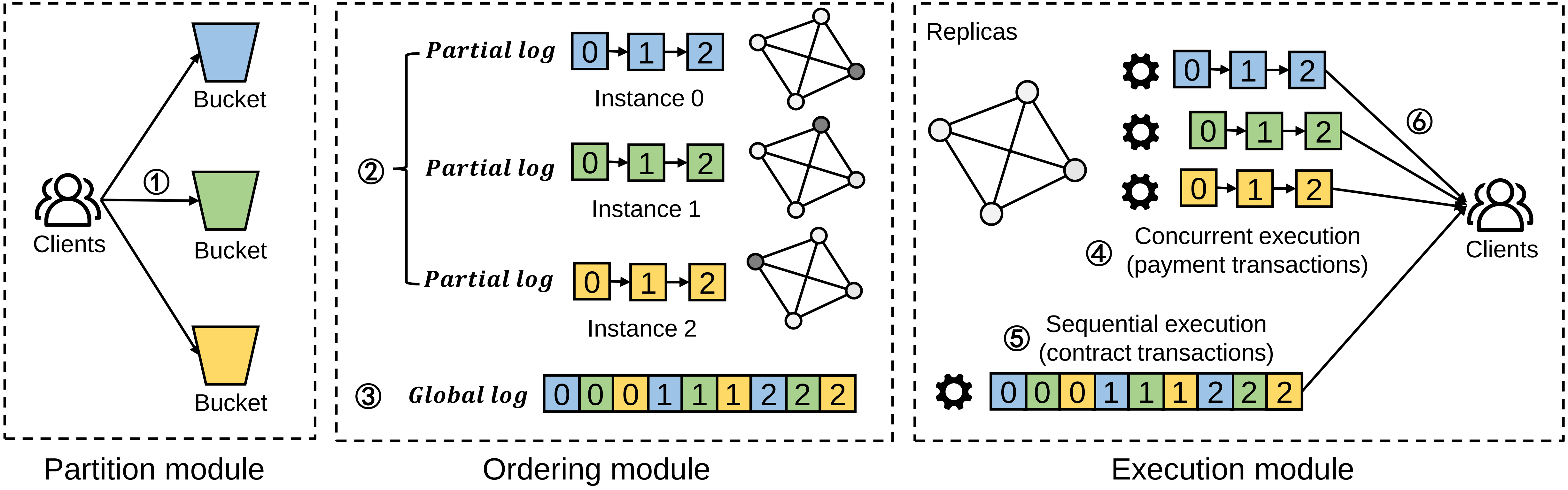
$tx0 - tx1: all\ txs\ success$
$tx1 - tx0: all\ txs\ success$

Key Observations

Observation 3: Smart contracts modifying the same record must be executed sequentially to preserve consistency.

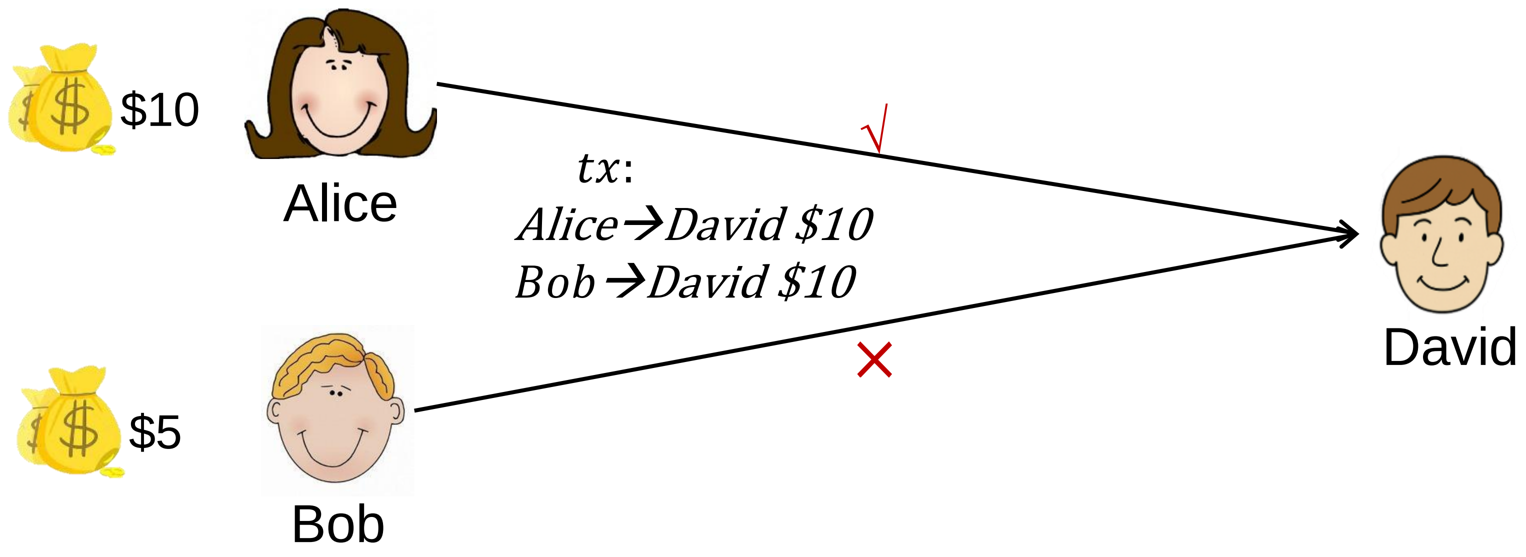


Orthrus Overview



Challenges and Solutions

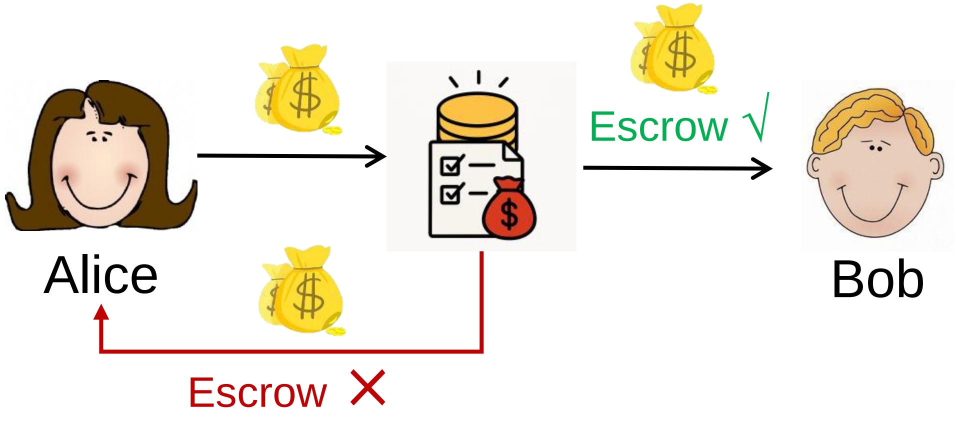
Challenge 1: Ensuring atomicity of transactions with multiple payers.

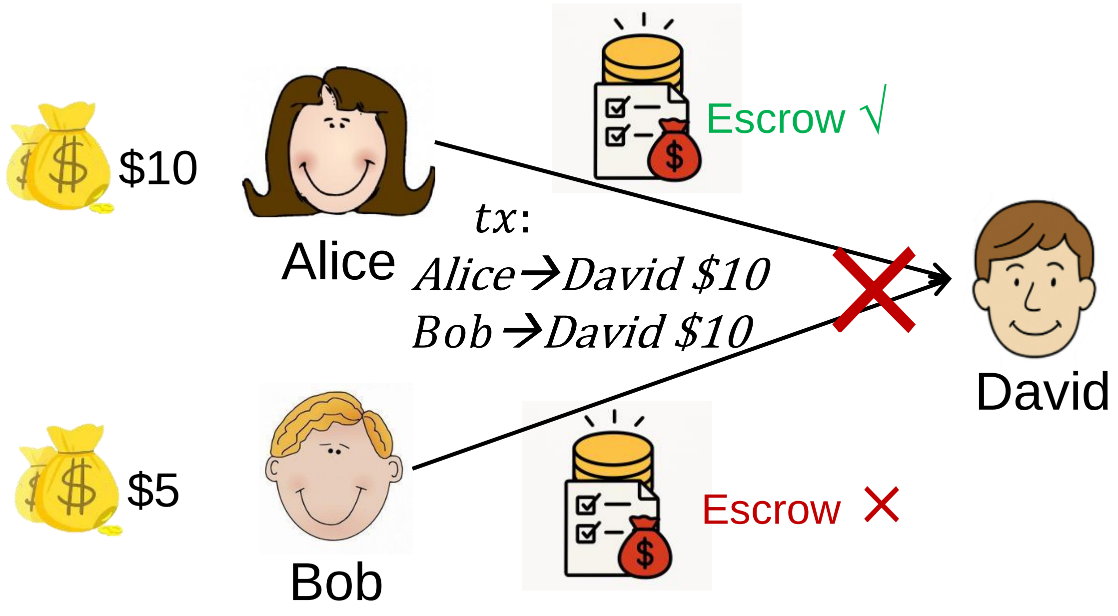


Challenges and Solutions

Solution: Escrow mechanism for atomicity.

Escrow can be understood as a "fund freezing" mechanism, which holds the funds temporarily and only releases them after the transaction succeeds, protecting both parties.

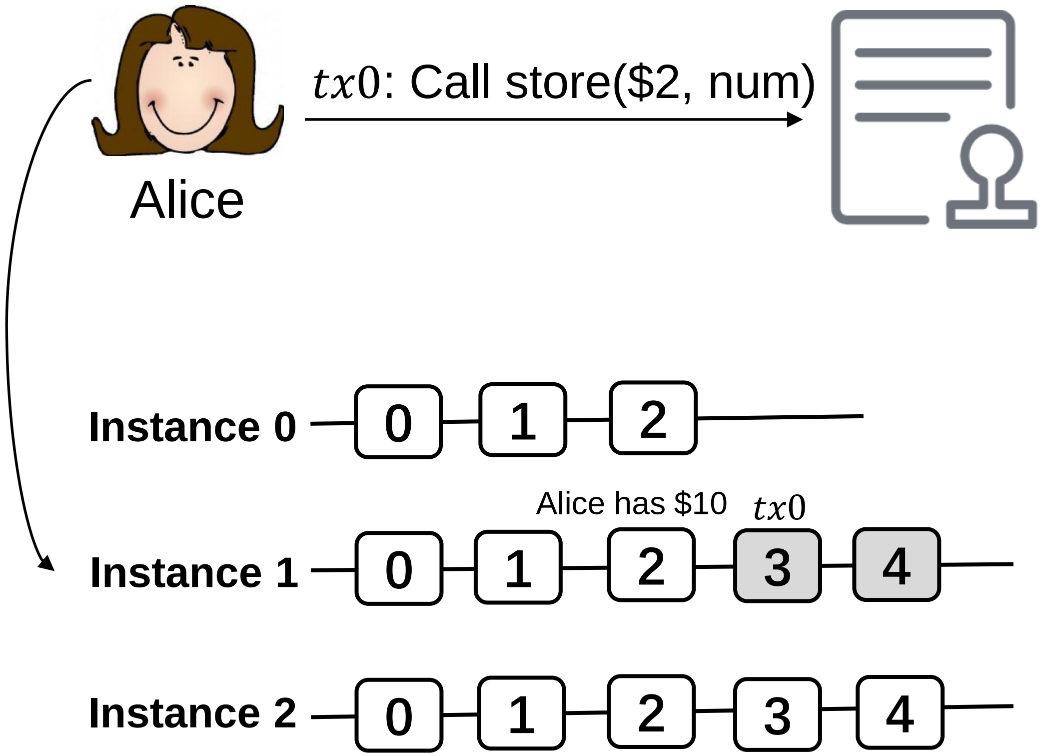


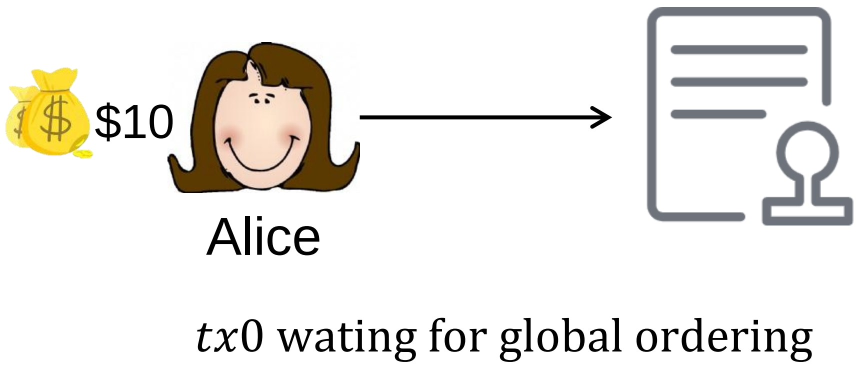


Challenges and Solutions

Challenge 2: Avoiding payment transactions blocking.

- Contract transaction may contain payment behavior (gas fee)

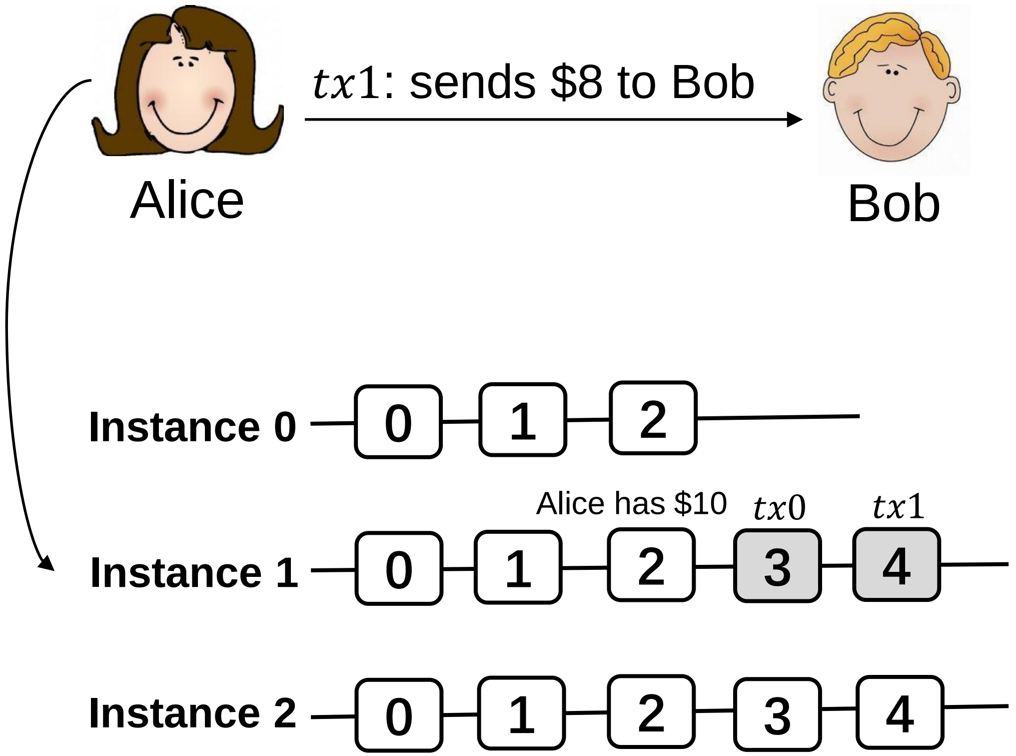


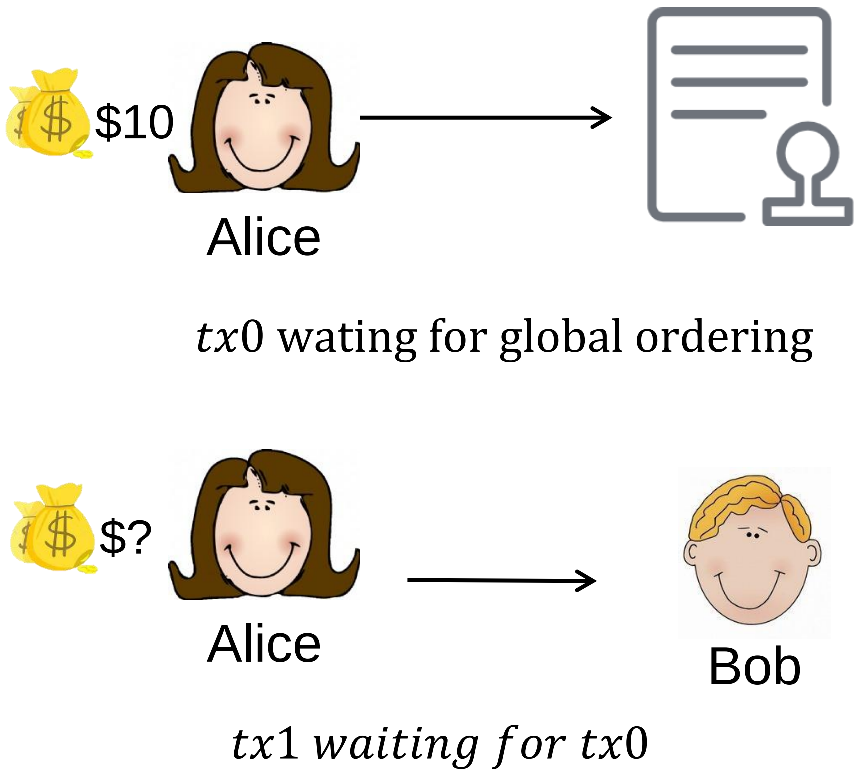


Challenges and Solutions

Challenge 2: Avoiding payment transactions blocking.

- Contract transaction may contain payment behavior (gas fee)

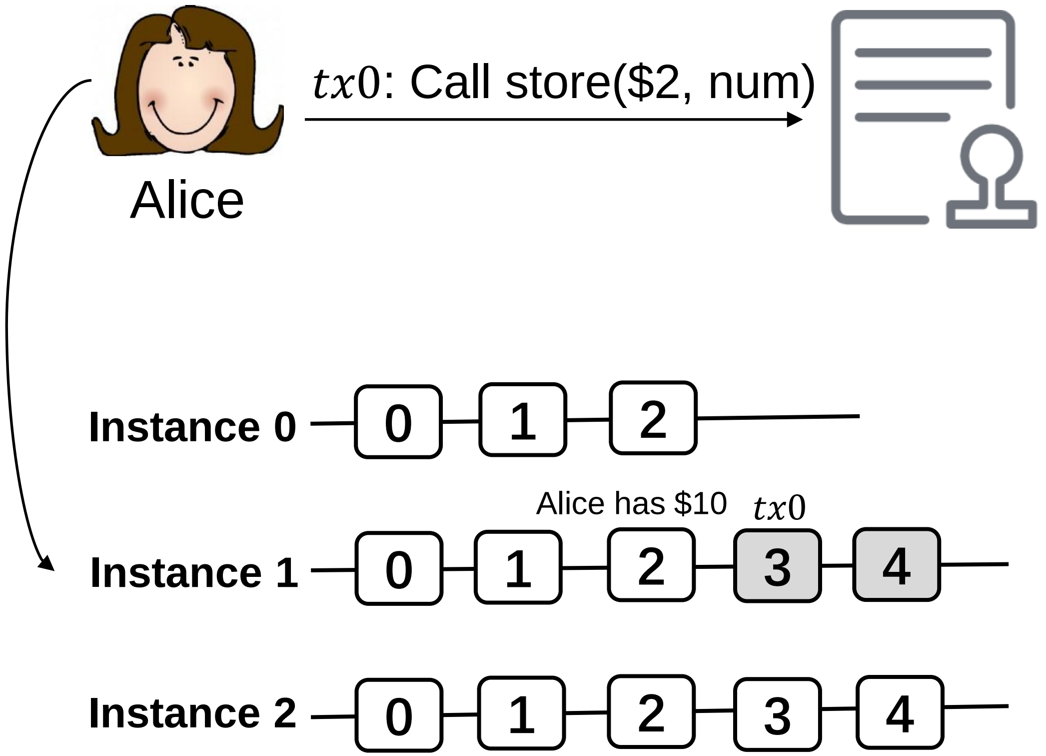


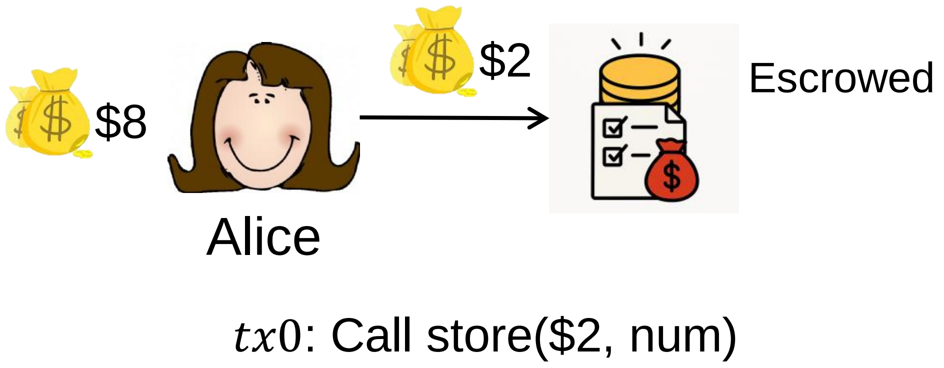


Challenges and Solutions

Solution: Escrow mechanism to avoid blocking.

- Contract transaction may contain payment behavior (gas fee)

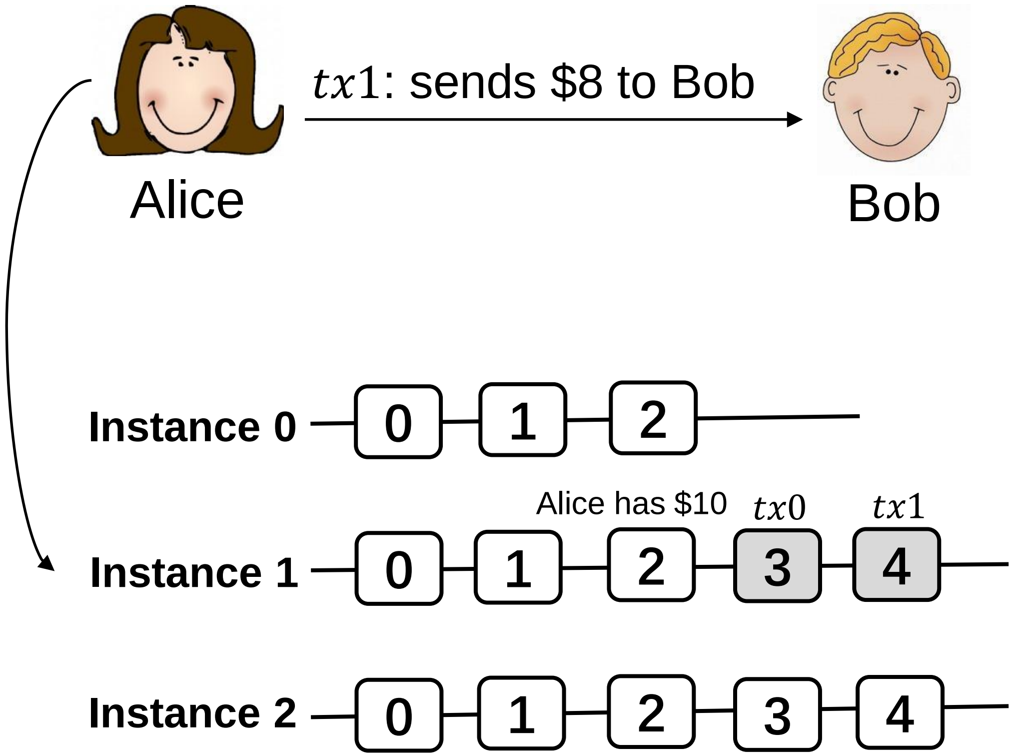


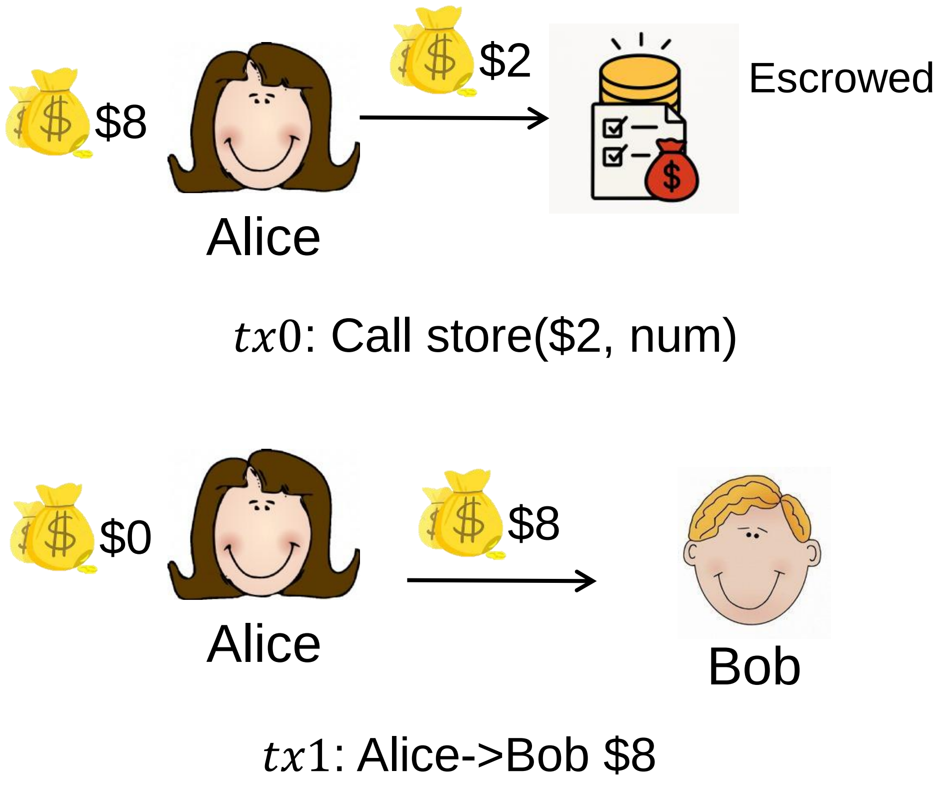


Challenges and Solutions

Solution: Escrow mechanism to avoid blocking.

- Contract transaction may contain payment behavior (gas fee)





Evaluation Settings

Testbeds

- AWS EC2 c5a.2xlarge machines
- 8vCPUs and 16GB RAM Ubuntu Linux 22.04
- Deployed both in LAN and WAN
- 1Gbps bandwidths

[1] Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolić. State Machine Replication Scalability Made Simple, in ACM EuroSys'22.
[2] Chrysoula Stathakopoulou, Tudor David, and Marko Vukolic. Mir BFT: Scalable and Robust BFT for Decentralized Networks, in Jsys'22.
[3] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. RCC: Resilient Concurrent Consensus for High-Throughput Secure Transaction Processing, in IEEE ICDE'21.
[4] Balaji Arun and Binoy Ravindran. Scalable Byzantine Fault Tolerance via Partial Decentralization, in PVLDB'22.
[5] Hanzheng Lyu, Shaokang Xie, Jianyu Niu, Chen Feng, Yinqian Zhang, Ivan Beschastnikh. Ladon: High-Performance Multi-BFT Consensus via Dynamic Global Ordering, in ACM EuroSys'25.

Evaluation Settings

THE UNIVERSITY OF BRITISH COLUMBIA

Testbeds

- AWS EC2 c5a.2xlarge machines
- 8vCPUs and 16GB RAM Ubuntu Linux 22.04
- Deployed both in LAN and WAN
- 1Gbps bandwidths

Baseline protocols :

- ISS[1]: PBFT/HotStuff
- Mir-BFT[2]: PBFT
- RCC[3]: PBFT
- DQBFT[4]: PBFT
- Ladon [5]: PBFT

[1] Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolić. State Machine Replication Scalability Made Simple, in ACM EuroSys'22.
[2] Chrysoula Stathakopoulou, Tudor David, and Marko Vukolic. Mir BFT: Scalable and Robust BFT for Decentralized Networks, in Jsys'22.
[3] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. RCC: Resilient Concurrent Consensus for High-Throughput Secure Transaction Processing, in IEEE ICDE'21.
[4] Balaji Arun and Binoy Ravindran. Scalable Byzantine Fault Tolerance via Partial Decentralization, in PVLDB'22.
[5] Hanzheng Lyu, Shaokang Xie, Jianyu Niu, Chen Feng, Yinqian Zhang, Ivan Beschastnikh. Ladon: High-Performance Multi-BFT Consensus via Dynamic Global Ordering, in ACM EuroSys'25.

Evaluation Settings

THE UNIVERSITY OF BRITISH COLUMBIA

Testbeds

- AWS EC2 c5a.2xlarge machines
- 8vCPUs and 16GB RAM Ubuntu Linux 22.04
- Deployed both in LAN and WAN
- 1Gbps bandwidths

Baseline protocols :

- ISS[1]: PBFT/HotStuff
- Mir-BFT[2]: PBFT
- RCC[3]: PBFT
- DQBFT[4]: PBFT
- Ladon [5]: PBFT

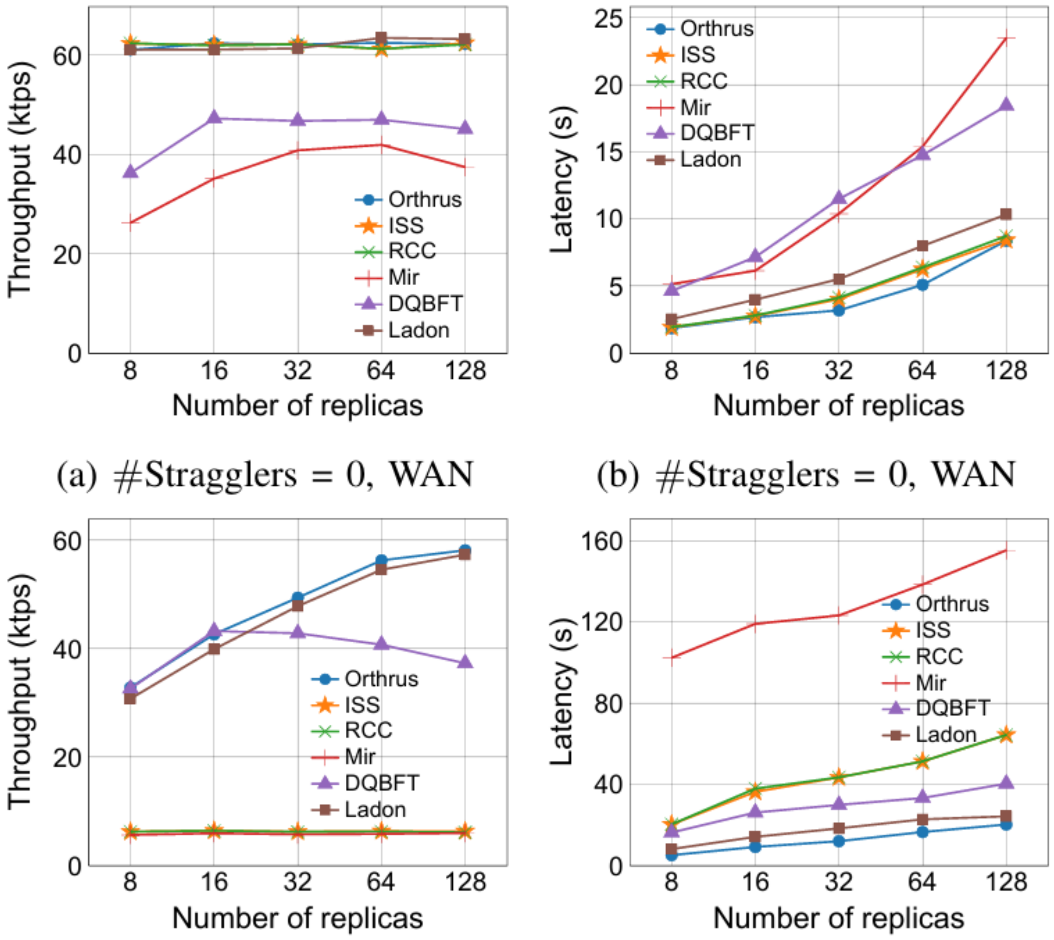
Research questions:

- How does Orthrus perform with varying replicas as compared to the baseline protocols?
- How does Orthrus perform under faults?

[1] Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolić. State Machine Replication Scalability Made Simple, in ACM EuroSys'22.
[2] Chrysoula Stathakopoulou, Tudor David, and Marko Vukolic. Mir BFT: Scalable and Robust BFT for Decentralized Networks, in Jsys'22.
[3] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. RCC: Resilient Concurrent Consensus for High-Throughput Secure Transaction Processing, in IEEE ICDE'21.
[4] Balaji Arun and Binoy Ravindran. Scalable Byzantine Fault Tolerance via Partial Decentralization, in PVLDB'22.
[5] Hanzheng Lyu, Shaokang Xie, Jianyu Niu, Chen Feng, Yinqian Zhang, Ivan Beschastnikh. Ladon: High-Performance Multi-BFT Consensus via Dynamic Global Ordering, in ACM EuroSys'25.

Scalability in WAN

System scalability with varying number of replicas in WAN



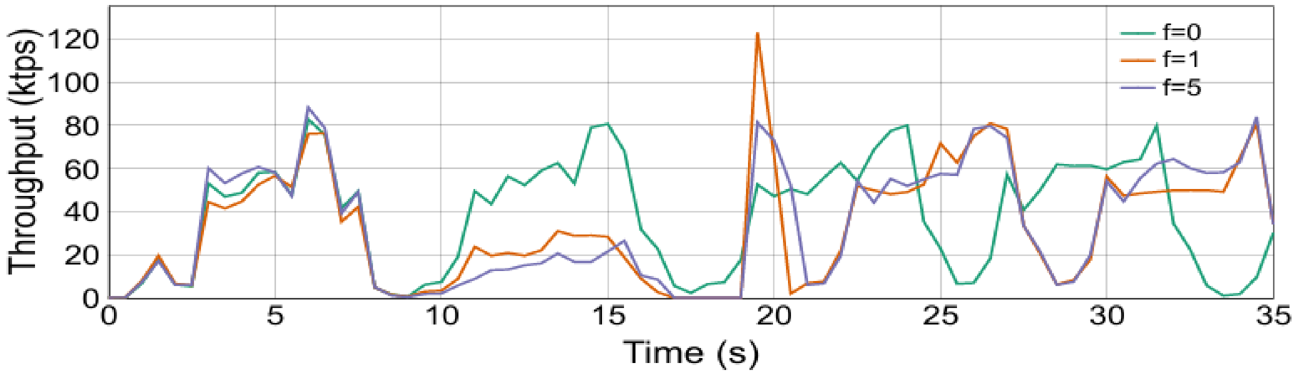
(a) #Stragglers = 0, WAN (b) #Stragglers = 0, WAN

(c) #Straggler = 1, WAN (d) #Straggler = 1, WAN

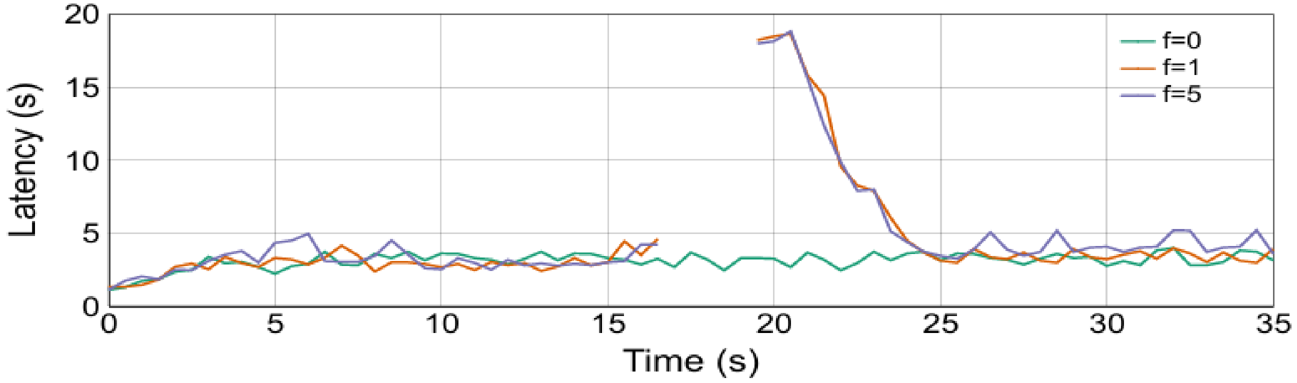
- 8/16/32/64/128 total replicas
- 0/1 stragglers
- ≈50% payment transactions
- Without stragglers
  - Comparable throughput/latency
- With one straggler
  - Superior throughput (9X)
  - Lower latency

Performance under Faults

Impact of detectable faults



(a) Throughput average (over 0.5s intervals) over time.

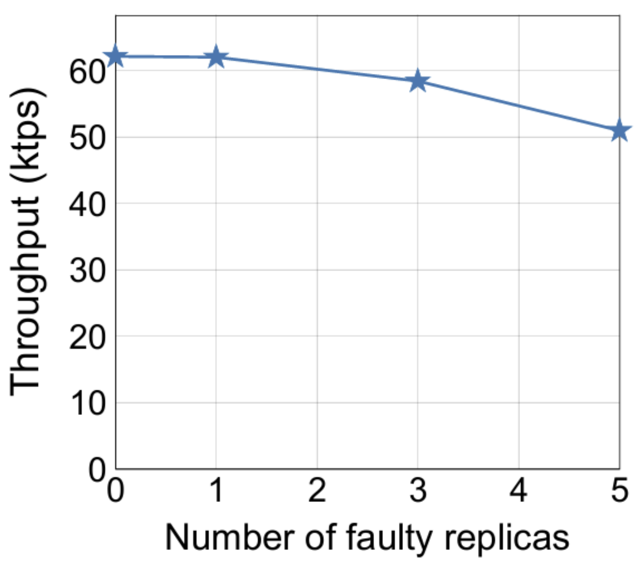


(b) Latency average (over 0.5s intervals) over time.

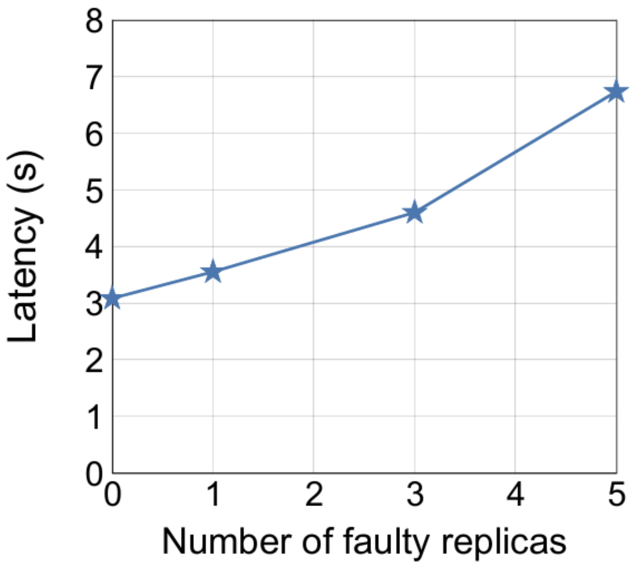
- 16 total replicas with 1 faulty leader
- View change timeout sizes: 10s
- The faults occur at 9 seconds
- Recovers after view change

Performance under Faults

Impact of undetectable faults



(a) Throughput



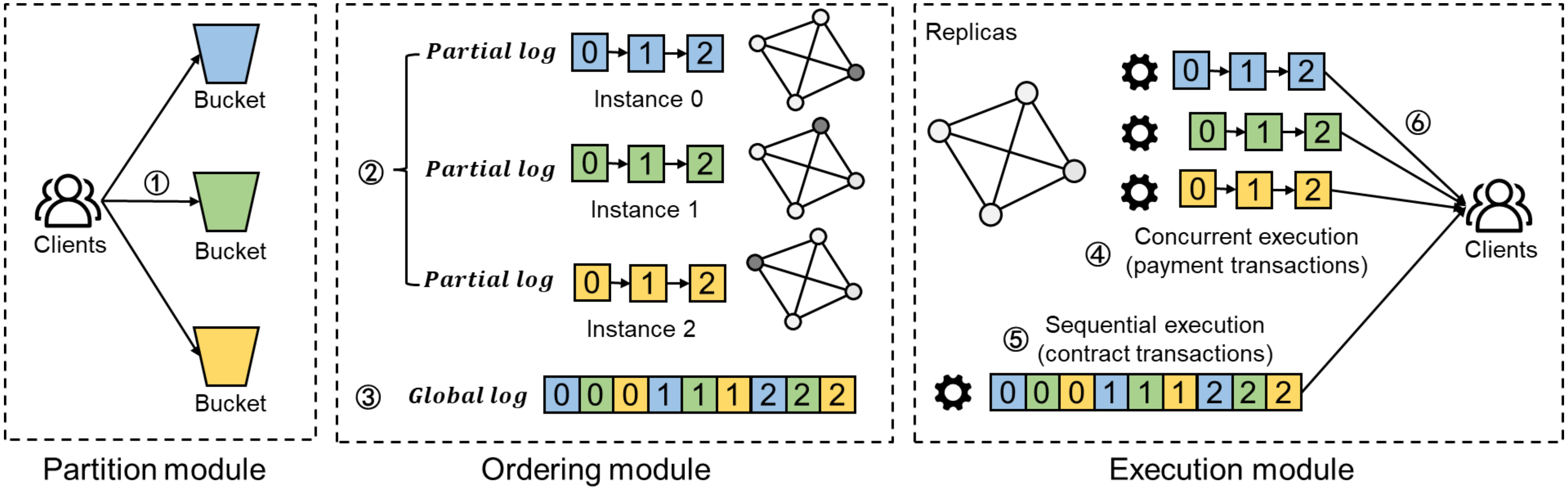
(b) Latency

- 16 total replicas with up to 5 faulty leaders
- The faulty leader avoids participating in other instances while continuing to propose blocks in the instance it leads
- throughput declines moderately
- Latency is impacted more significantly

Contributions

Orthrus: Accelerating Multi-BFT Consensus through Concurrent Partial Ordering of Transaction

- Code: **https://github.com/Hanzheng2021/Orthrus/**
- Leveraging a hybrid design, Orthrus integrates partial ordering for conflict-free transactions and global ordering for general non-commutative transactions.
- Introduce a customized escrow mechanism to ensure transaction atomicity and seam less interaction between partial and global logs.



Thanks for Listening!

Jianyu Niu

https://jianyu-niu.github.io/

niujy@sustech.edu.cn