BIDL: A High-throughput, Low-latency Permissioned Blockchain Framework for Datacenter Networks

Presenter: Chen-Yu Yu, Tsung-Chieh Chen, Jessica Chen, Elliot Lin

Ji Qi, Xusheng Chen, Yunpeng Jiang, Jianyu Jiang, Tianxiang Shen, Shixiong Zhao, Sen Wang, Gong Zhang, Li Chen, Man Ho Au, Heming Cui
The University of Hong Kong
Huawei Technologies Co., Ltd.

UCDAVIS

Outline

Introduction

System model

BIDL's workflow

BIDL's solution to malicious participants

Evaluation

Introduction

Current Permissioned Blockchain

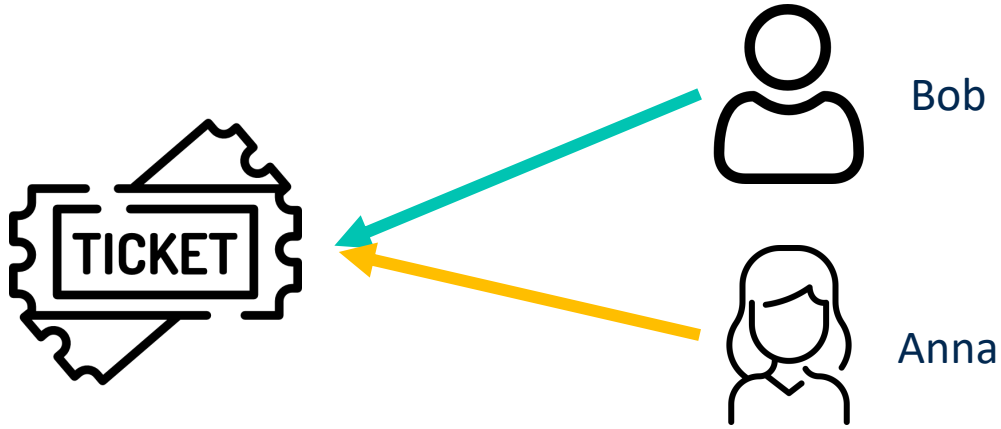
- **Growing Demand**: Increasingly pervasive with high-performance expectations
- Traditional Trading Systems (e.g., Hong Kong Stock Exchange)
  - Processes up to 50K transactions/second
  - Commit latency of only tens of milliseconds
- Lower performance on blockchains (e.g., Hyperledger Fabric (HLF))
  - Processes 9.3K transactions/second
  - Commit latency of 100 milliseconds
- Primary Reason: **Sequential workflow** in both consensus and transaction execution

Non-Deterministic Transactions

- Have outcomes that are not consistently predictable
- When executed multiple times under the same initial conditions, they might **yield different results**
- May happen with malicious node generating faulty results or sequence number
- Impact on blockchains
    - Different nodes observe different outcomes for the same transaction, consensus can break down

Contending Transactions

- Transactions are competing for the same resources, (e.g., data race)
- When multiple transactions try to **access or modify the same piece of data or resource concurrently**
- Impact on blockchains
  - Data integrity and consistency
- Example



Overbooking

Traditional blockchain workflow

1. Execute → Consensus (e.g., HLF):
    I. Nodes first concurrently execute received transaction
    II. Agree on the order of execution results through a BFT consensus protocol

    Advantage

    - Support non-deterministic transaction

    Disadvantage

    - **High abort rate on contending transactions without sequence numbers**

Traditional blockchain workflow

2. Consensus → Execute (e.g., Quorum):
    I. Nodes first use a BFT consensus protocol to order transaction contents
    II. Individually execute transactions according to this order

    Advantage

    - Support contending transactions
    - Zero abort rate on transactions

    Disadvantage

    - **Sequential workflow** with a deterministic single-threaded language (Tedious and inefficient)

BIDL Features

- Parallel workflow
- Able to use different BFT consensus protocols
- Zero abort rate
  - Able to handle non-deterministic transactions
  - Support contending transactions

BIDL System Overview

Participants

- **Clients**: Generate & sign transactions
- **Nodes**:
  - **Consensus Node (CN)**: Conduct BFT consensus to commit transactions
  - **Normal Node (NN)**: Handle transaction execution
  - Grouped into organizations
  - Can be in multiple organizations

Organizations (e.g., banks, merchants)

- Contains at least one consensus node (CN) and one normal node (NN)
- Nodes trust all members within the same organization, but don't trust nodes from other organizations

BIDL System Model Assumptions

Network Assumption: Triangle Inequality

- Time taken for message delivery satisfies: $l_{i \to j} + l_{j \to k} \geq l_{i \to k}$
- Setup
  - Nodes are located within a datacenter
  - Dedicated network cables ensure stable connections
- Communications between correct nodes occur within a predefined maximum delay

BIDL Workflow Overview

Leader responsibilities

- Assigns sequence numbers to transactions
- Multicasts transactions to all nodes

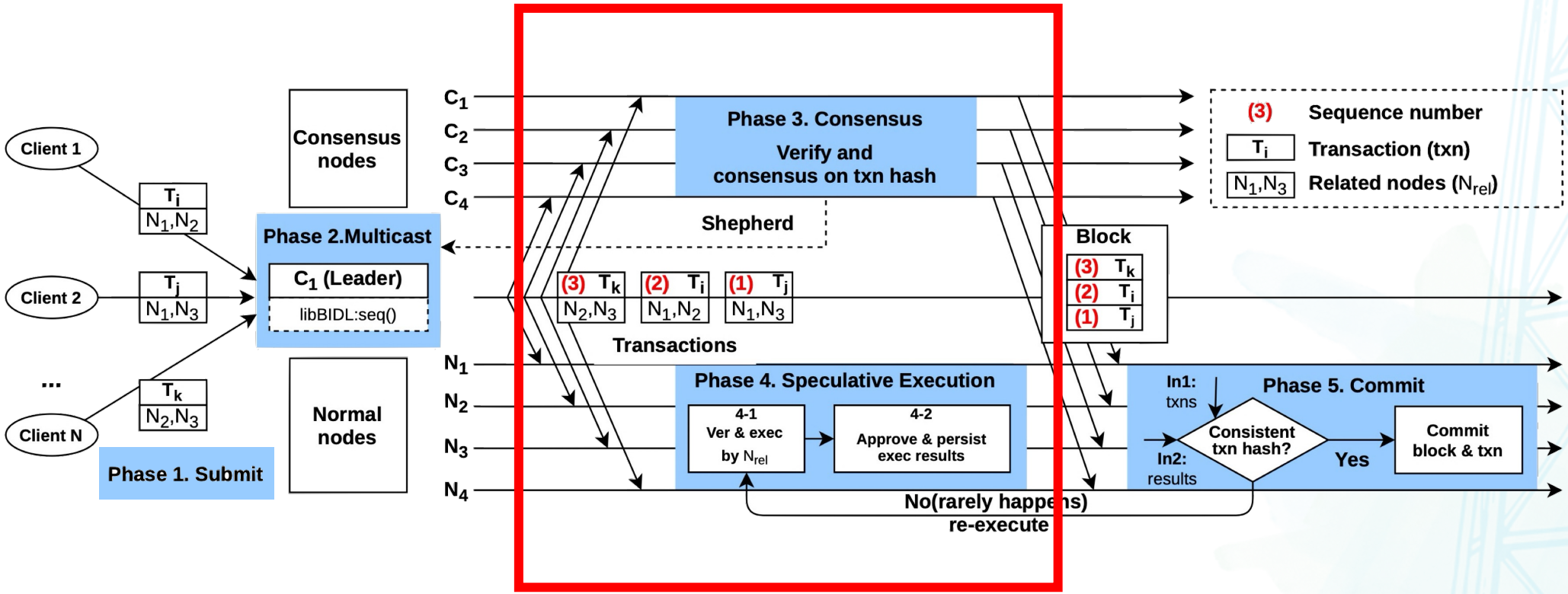
Parallelized workflow

- $\boldsymbol{CN}$ **:** Conduct BFT consensus for committing transactions
- $\boldsymbol{NN}$**:** Performs speculative execution based on sequence numbers

] Execute in parallel

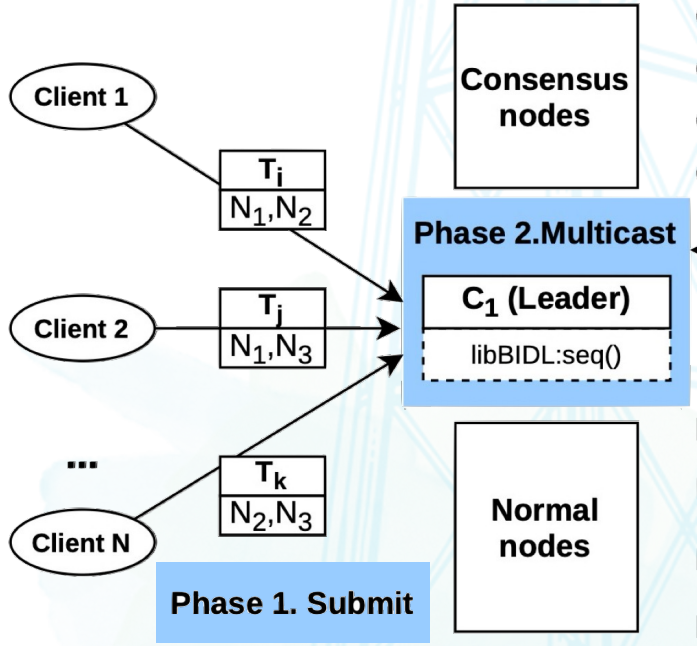
- Transactions are committed only if the sequence number are consistent
- In case of inconsistency, transactions are re-executed sequentially

BIDL Workflow Overview



Phase 1: Submit

- **Clients submit transactions** to the leader of CN
  - Message sign with client's private key
- The leader drops connection if a client send a malformed transaction (e.g., with invalid signatures)
- Helps prevent clients from mounting DoS attacks on the leader



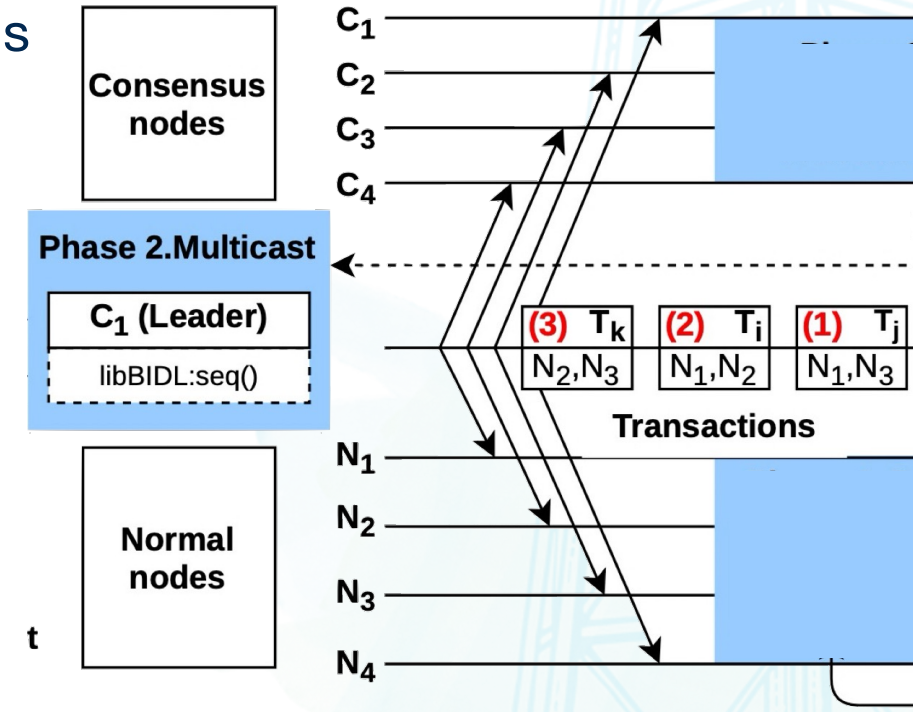
Phase 2: Multicast

Leader (Selected from CNs)

- Assigns consecutive sequence numbers to transactions
- **Multicasts to all CNs and NNs**

Performance optimization

- Avoid signing sequence numbers:
  - Reduce computational cost to prevent resource exhaustion attacks from adversaries
- Adopt a denylist design



Phase 2: Multicast

Transaction Verification Process after receiving transaction in CN and NN

A. Sequence Check

- If sequence number s already received or belongs to a previous block → Discard transaction
- If s belongs to a future block → Cache transaction for later
- If s is in the current block → Proceed to step B

B. Replay Check

- If transaction with the same hash has been received earlier → Discard transaction
- Otherwise → Proceed to step C

C. Signature Check

- Verify client's signature if the transaction pass prior checks (**computationally intensive)**

Phase 3: Consensus

Consensus Nodes (CNs)

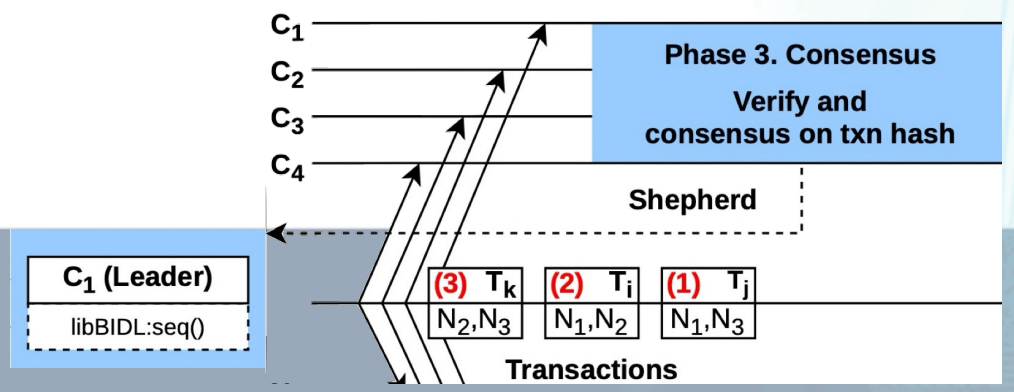
- **Run a BFT protocol to agree on a sequence of transaction** received from the leader
- BIDL treats the BFT protocol as a blackbox

Phase 3 Processes

1. Leader broadcasts a PROPOSE message to all CNs
2. If a CN misses any transaction, it requests the leader for a retransmission
3. All transactions for a block are received or timeout occurs, CN begins the BFT protocol
4. Upon reaching consensus, the CN assembles transactions into a block and send to NN

Performance optimization

- Compare transaction hashes only
- Each transaction hash is unique



Phase 4: Speculative Execution

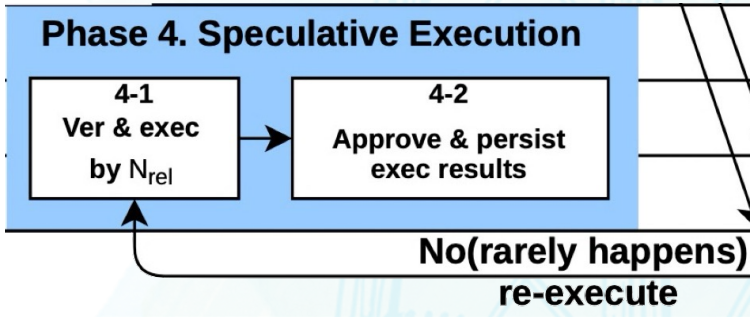
Run in parallel with Phase 3

Phase 4-1: Speculatively Execute Client Transactions

- *NNs* execute transactions according to sequence numbers
- Eliminates aborts caused by the concurrent execution of contending transactions

Phase 4-2: Approve & Persist Execution Results

- BIDL treats all transactions as potentially non-deterministic
- Ensure identical and retrievable execution results



Phase 4: Speculative Execution

Phase 4-2 Processes

Delegate

1. Each organization pick a delegate from its NNs
2. Delegate collects the signed result of each executed transaction from every organization and creates a vector $\vec{r}$
3. Delegate send $\vec{r}$ to all CNs

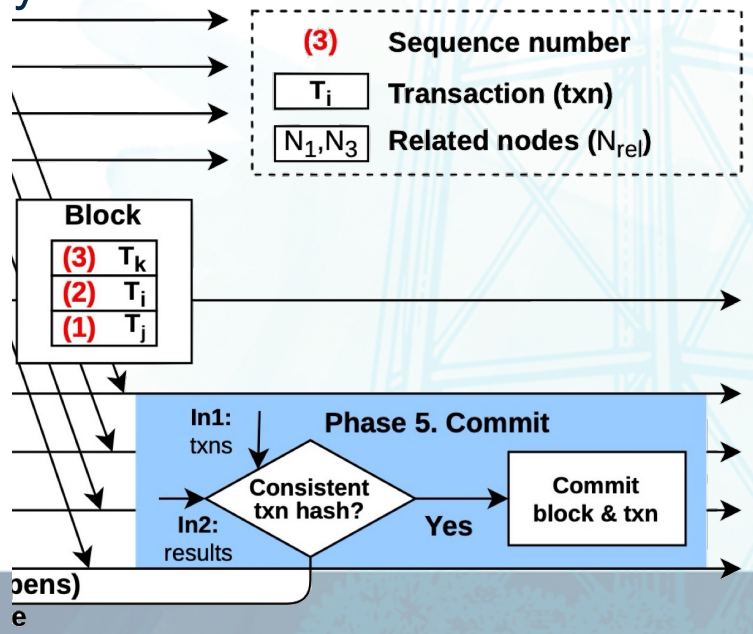
→

CN

1. Persist one $\vec{r}$ for the transaction
2. Sends PERSIST message to NNs

Phase 5: Commit

1. Block reception: *NNs* receive a signed block from **Phase 3**
2. Message reception: *NNs* receive PERSIST messages from *CNs*
3. Consistency verification
    I. Block must contain 2f+1 valid signature from different *CNs* for further verification
    II. NN must receive 2f+1 PERSIST message to commit locally

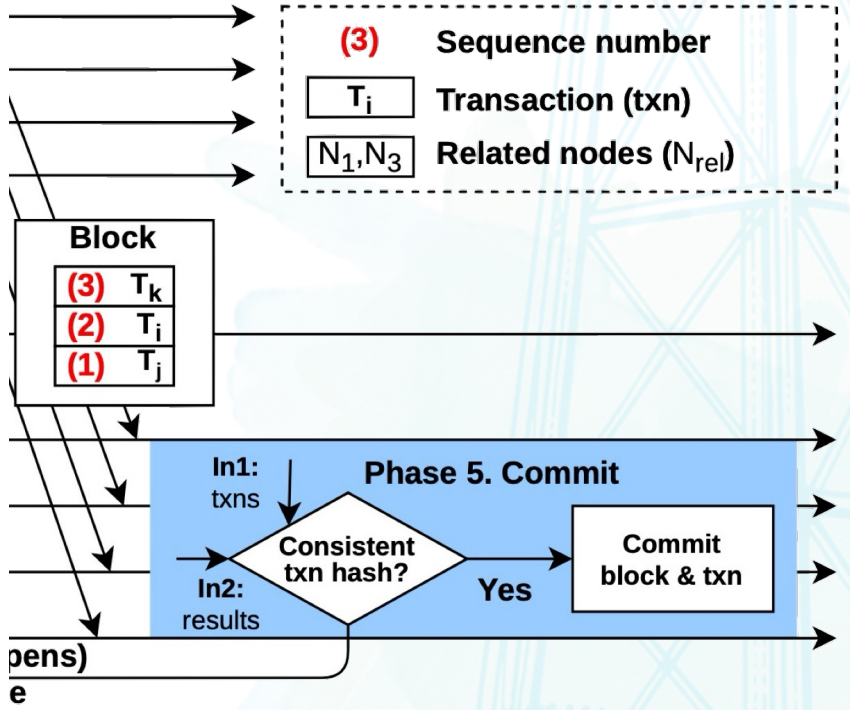


Phase 5: Commit

4. Transaction outcome:
   - Check the hashes of all transactions in the block with those locally stored
   - Consistent transactions
     - Commit transactions
     - Add block to the ledger
   - Inconsistent transaction
     - Re-execute all related transactions
     - Switch from parallel to sequential workflow

Performance optimization

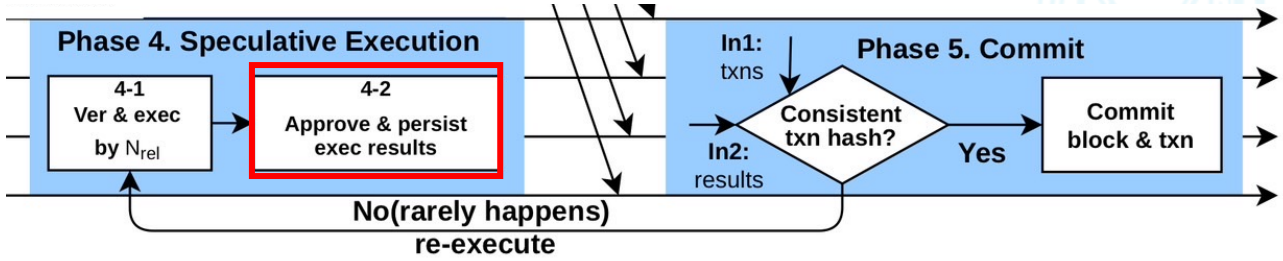
- Comparing only the hashes



How BIDL handle the challenges

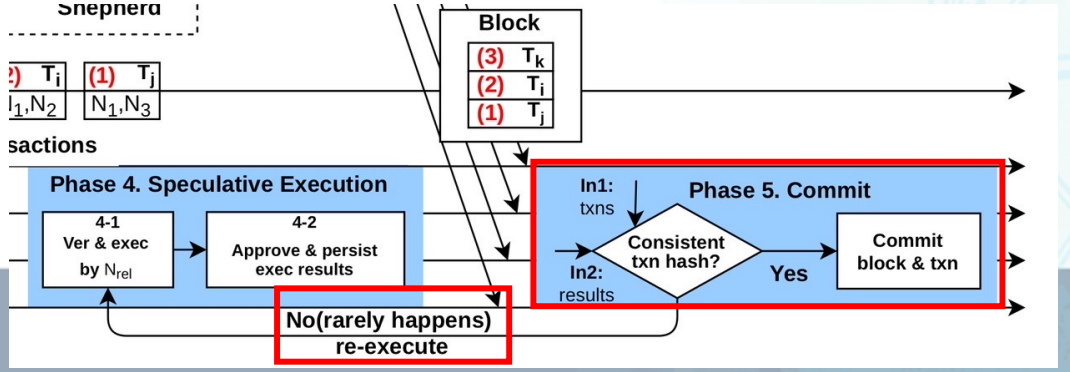
Handling **non-determinism** in transactions:

- Inconsistent results are regarded as aborted by NNs



Handling contending transactions:

- Transactions are speculatively executed based on their sequence number
- Eliminate aborts caused by contention



BIDL: A High-throughput, Low-latency Permissioned Blockchain Framework for Datacenter Networks

Presenter: Chen-Yu Yu, Tsung-Chieh Chen, Jessica Chen, Elliot Lin

Ji Qi, Xusheng Chen, Yunpeng Jiang, Jianyu Jiang, Tianxiang Shen, Shixiong Zhao, Sen Wang, Gong Zhang, Li Chen, Man Ho Au, Heming Cui
The University of Hong Kong
Huawei Technologies Co., Ltd.

UCDAVIS

Outline

Network Models

Safety and Liveness

Compare BIDL with PBFT

BIDL's solution to malicious participants

Evaluation

Different Network Models

Asynchronous Network

- No guaranteed maximum time delay
- Messages may be indefinitely delayed

Partially Synchronous Network

- Delay is unknown but finite
- Messages will be delivered eventually
- PBFT
- BIDL

How BIDL ensure safety

Safety in BIDL

- Unique Transaction Hashes
  - Ensures each transaction hash is distinct
  - Ensures integrity
- Data Redundancy
  - Results are stored in all non-faulty nodes
  - Maintain data integrity and availability

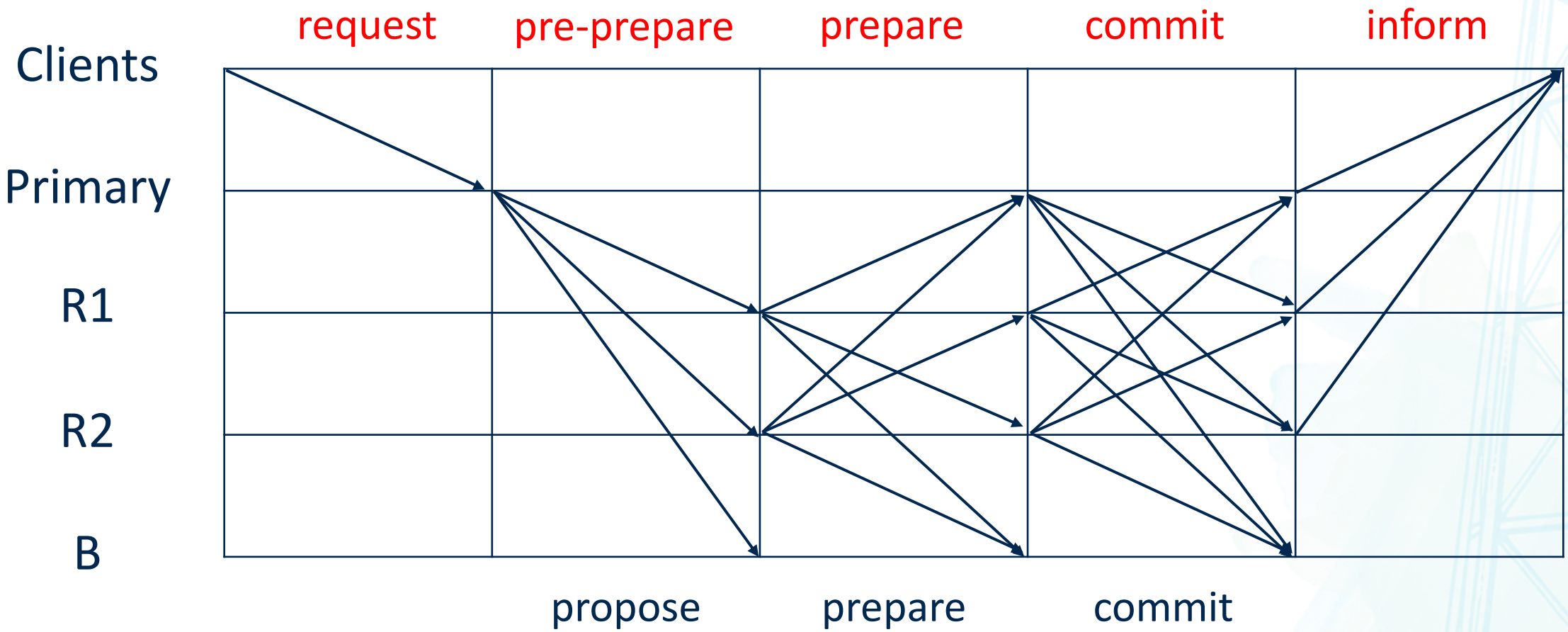
How BIDL ensure liveness

Liveness

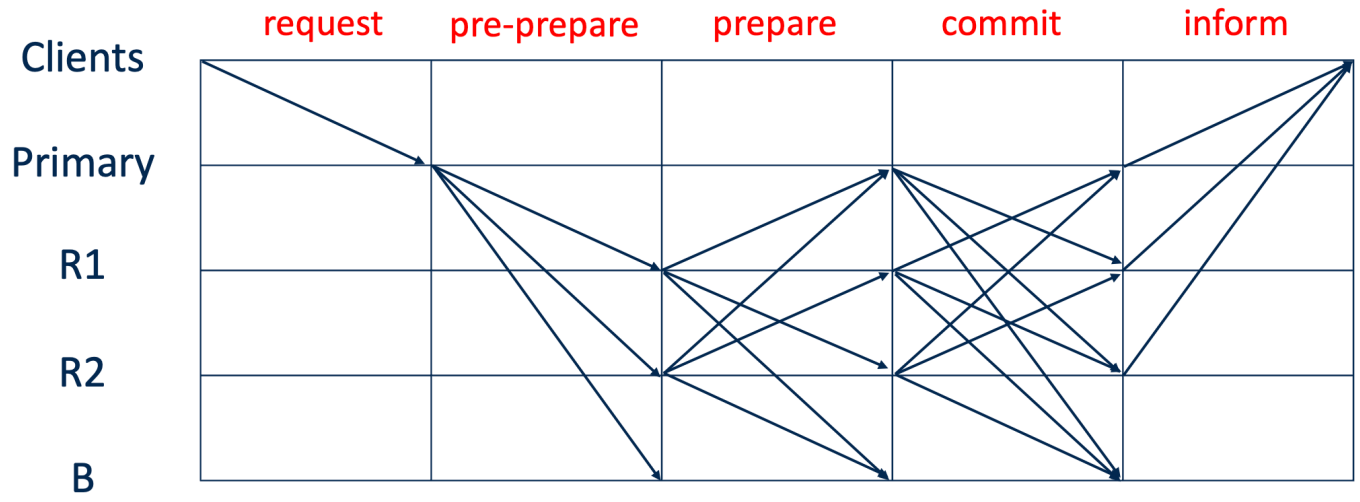
- Valid transactions will eventually be committed
    - Transaction stored in related organizations, which has no incentive to block that transaction
    - What if transactions are neglected
        1. Client sends the transaction to the leader
        2. After a timeout, the client resends the transaction to all CNs
            - Normal node will re-execute if CNs reach consensus
        3. If transactions are still not committed, the consensus node initiate a view change

PBFT

Purpose of commit: To ensure that the majority of replicas have been prepared



Compare BIDL with PBFT



PBFT phase:

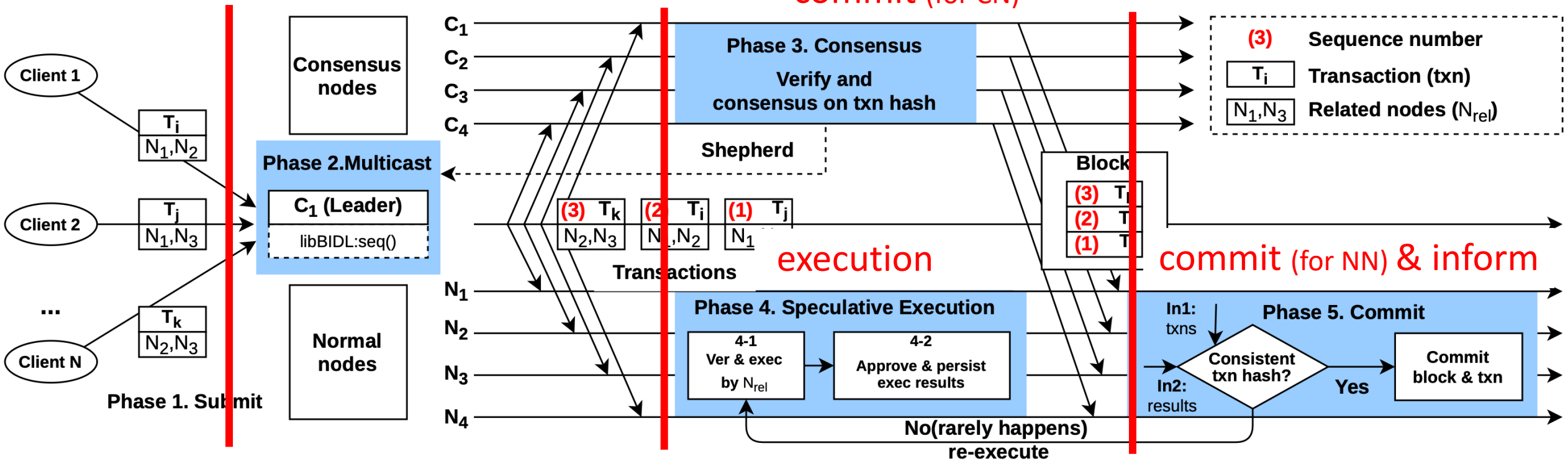
request

pre-prepare

pre-prepare & prepare & commit (for CN)

execution

commit (for NN) & inform



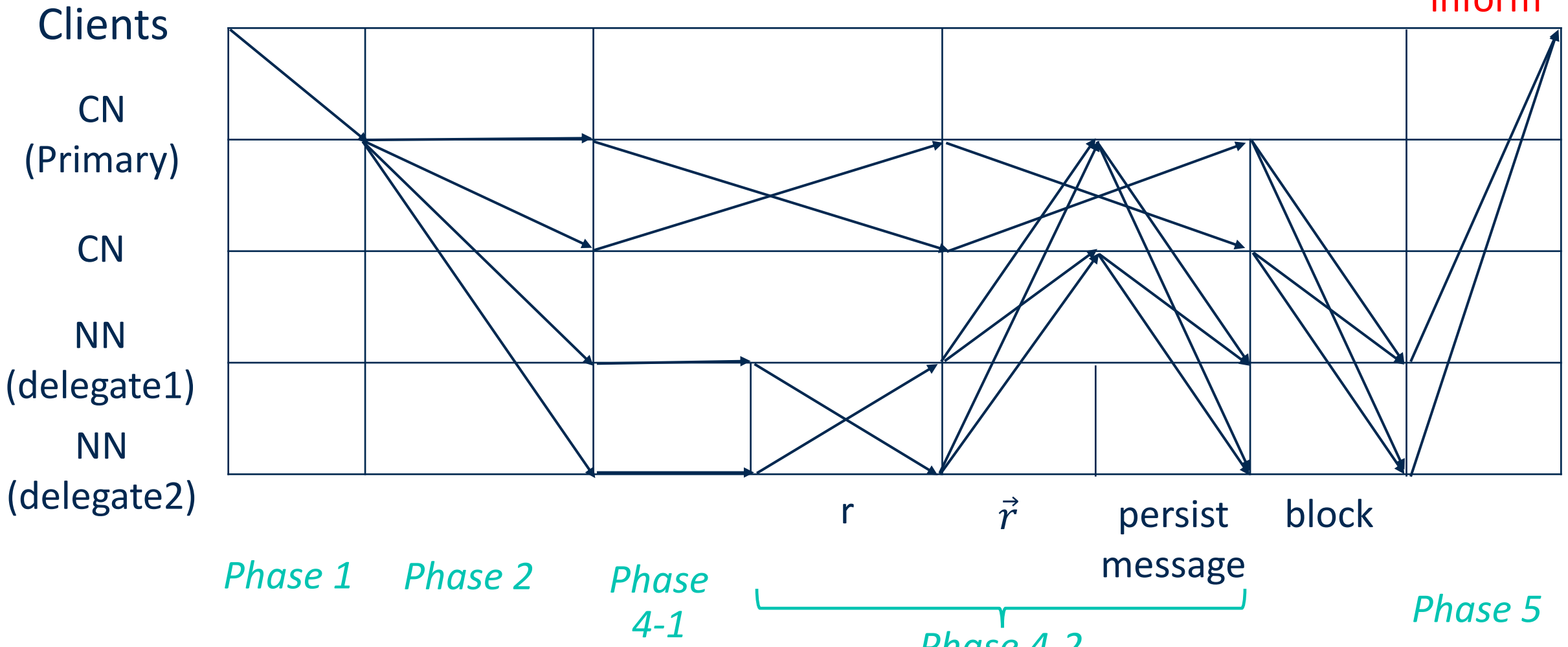
Compare BIDL with PBFT

Phase 3

Can be changed with corresponding phase of other BFT protocol

PBFT phase: request pre-prepare prepare commit (for CN)

commit (for NN) + inform



BIDL's Solution to Malicious Participants

Threat Model

How does BIDL handle participants node after eliminate signatures on sequence numbers?

Malicious participants Actions

- Send conflicting transactions with same sequence numbers to different nodes
- **Challenge:** Distinguishing if a conflict arises from an honest leader's error or an adversary's crafted transaction

Denylist-based view change protocol

- Nodes accept only the first received transaction, discarding subsequent ones
- Add malicious clients to the denylist to mitigate repeated adversarial actions

Denylist Protocol (for malicious client)

Local Suspect List

- A node detects conflicting transactions with the same sequence number from phases 2 and 3
- Add the suspected client to the local suspect list

→

Continue to monitor

- Considered suspicious by a node across f+1 views from different leaders
- Regard as malicious
- Include in view change message

→

Denylist

- If f+1 *CNs* mark the client as malicious
- Add the client to the denylist
- Client would be excluded from the denylist after a time window

Denylist Protocol (for malicious client)

- A client might be added to the denylist accidentally if the triangle inequality property is violated
  - e.g., abnormal network delay
- What could a client do after being added to the denylist:
  1. Client sends the transaction to the leader
  2. After a timeout, the client resends the transaction to all CNs
     - Normal node will re-execute if CNs reach consensus
  3. If transactions is still not committed, the consensus node initiates a view change
- Client would be excluded from the denylist after a time window
- BIDL **ensure liveness** (Valid transactions will eventually be committed)

View Change

It is similar with PBFT but with **two key differences**:

1. Unpredictable Leadership
   - PBFT Approach: Uses a round-robin selection
   - BIDL's Approach: Uses a Shepherding method
     - New leader is selected randomly from CNs
     - Selection based on the hash of the last committed block as the random seed
     - **Aim:** To prevent malicious client from framing the same leader
2. View change message contains BIDL’s denylist

View Change Trigger

1. Ensure high performance
    - A re-execute rate larger than 1% in **Phase 4**
    - Throughput degradation in **Phase 3**

      (smaller than 90% of the peak throughput in previous views)
    - Increasing delay for receiving persist requests in **Phase 4**
2. Situation discussed earlier when the client is in the denylist
3. Proactively initiated by a correct leader

Evaluation

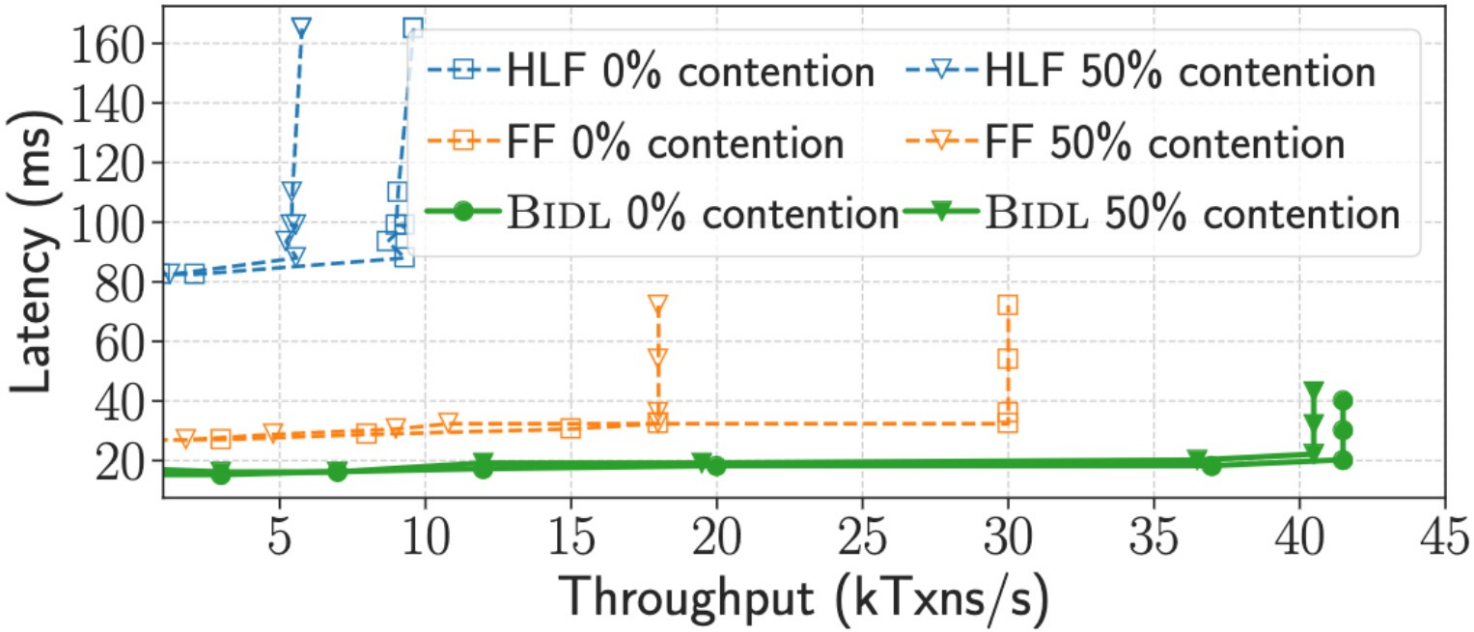
Evaluation

Compare BIDL with 3 blockchain frameworks:

1. HLF
   - Execute → Consensus workflow
   - Sequential
2. StreamChain
3. FastFabric (FF)

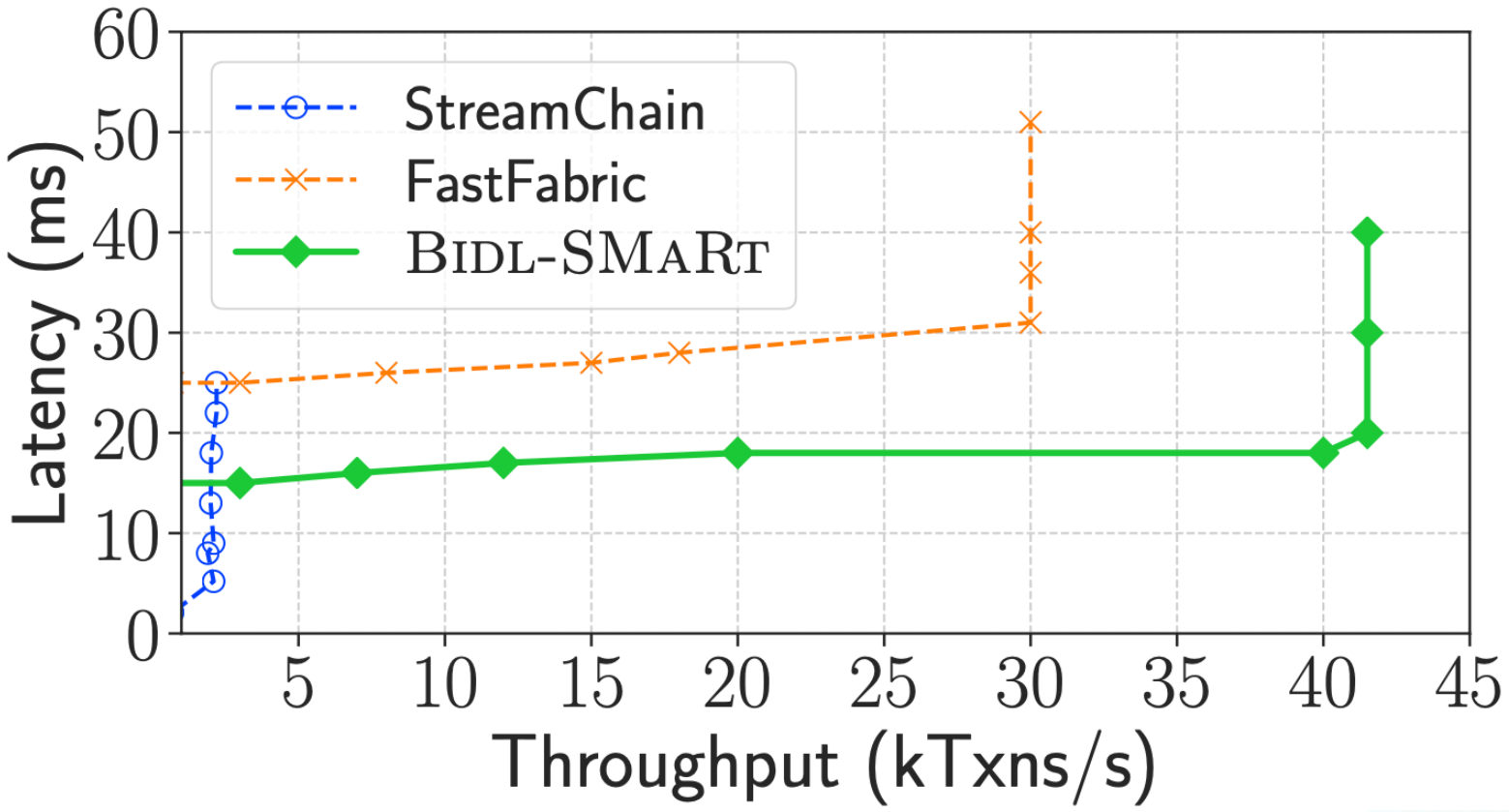
Evaluation

- Consensus Nodes: 4
- Normal Nodes: 50
- No malicious nodes



How Efficient is BIDL's parallel workflow

Throughput vs. latency on four consensus nodes without any malicious node



How robust is BIDL to malicious participants

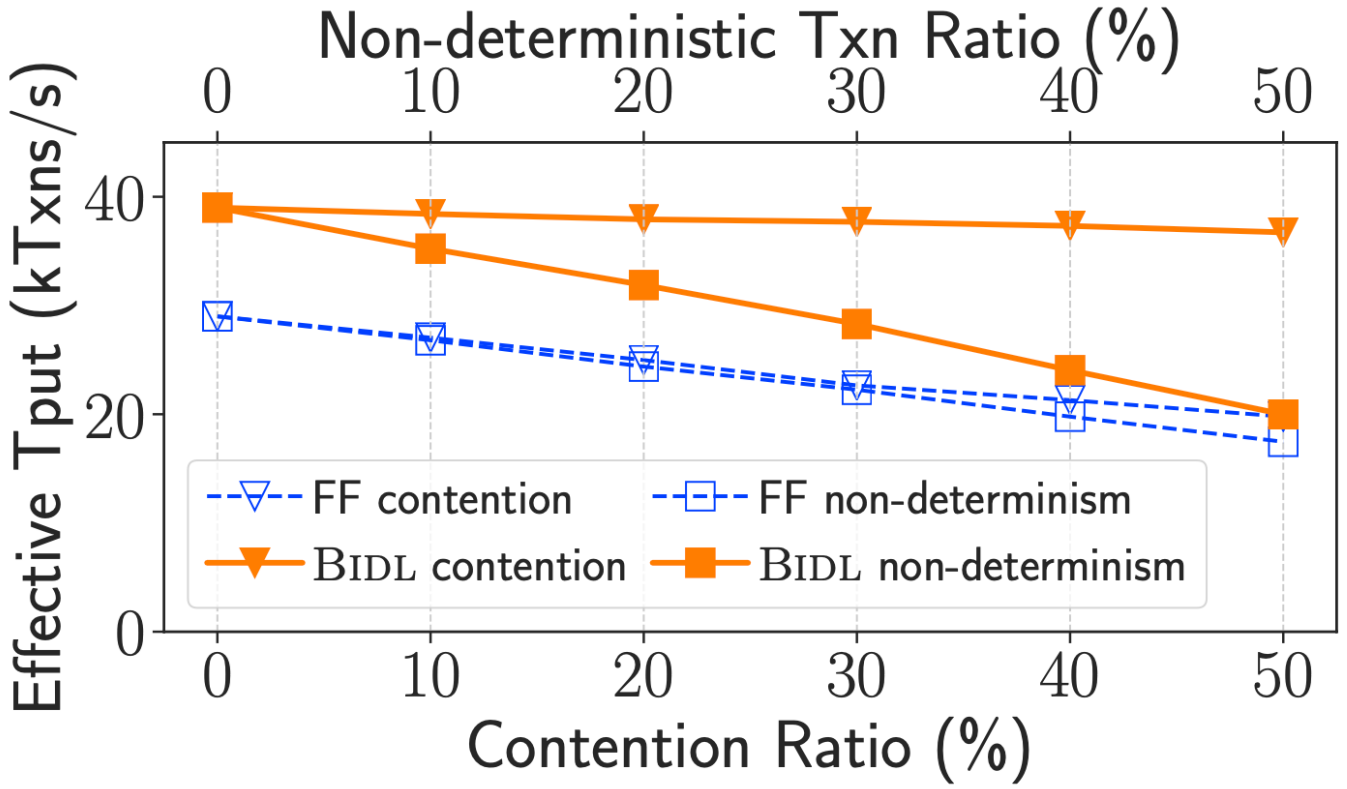
Effective throughput in three scenarios

- S1: fault-free case
- S2: the malicious leader proposes a series of invalid transactions
- S3: a malicious broadcaster broadcasts a carefully crafted series of transactions to all nodes

Blockchain Frameworks	Effective Throughput (kTxns/s)		
	S1 Fault Free	**S2** Malicious Leader	**S3** Malicious Broadcaster
StreamChain [65]	2.73	N/A	N/A
HLF [31]	9.25	9.25	9.25
FastFabric [51]	29.32	N/A	N/A
BIDL w/o denylist	41.67	41.67	10.75
BIDL	41.67	41.67	41.67

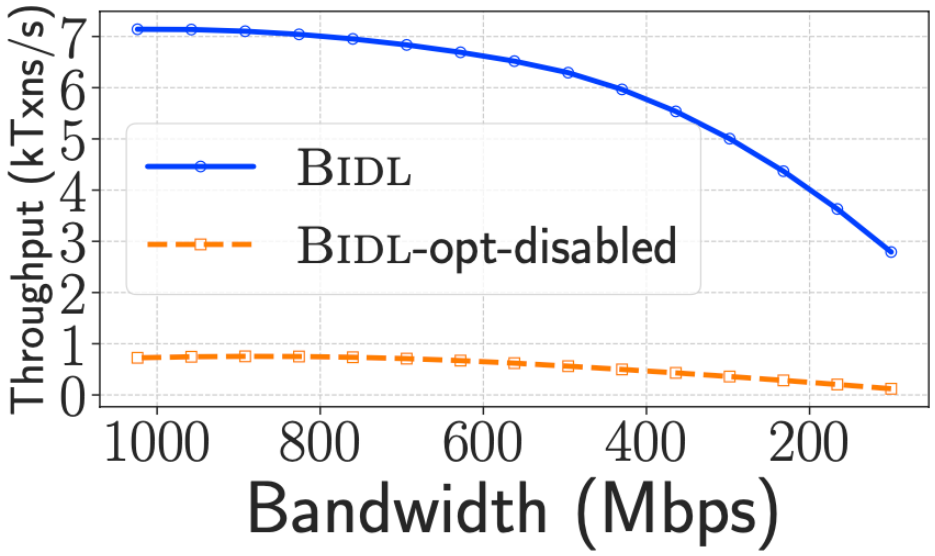
How robust is BIDL's performance to non-deterministic and high-contention workloads

Robustness of BIDL and FastFabric with different workloads

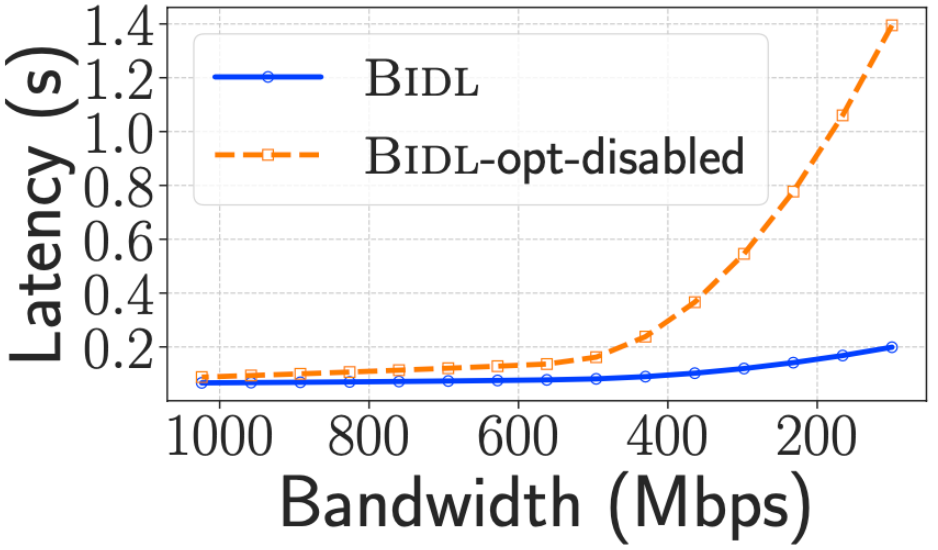


How is BIDL's performance over multiple datacenters

Performance in cross-datacenter deployment



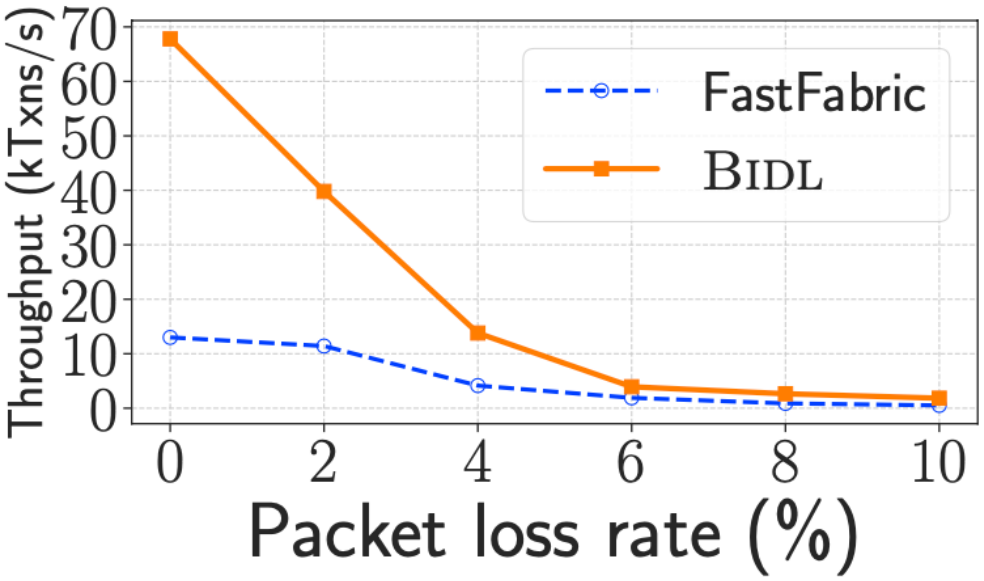
(a) End-to-end throughput.



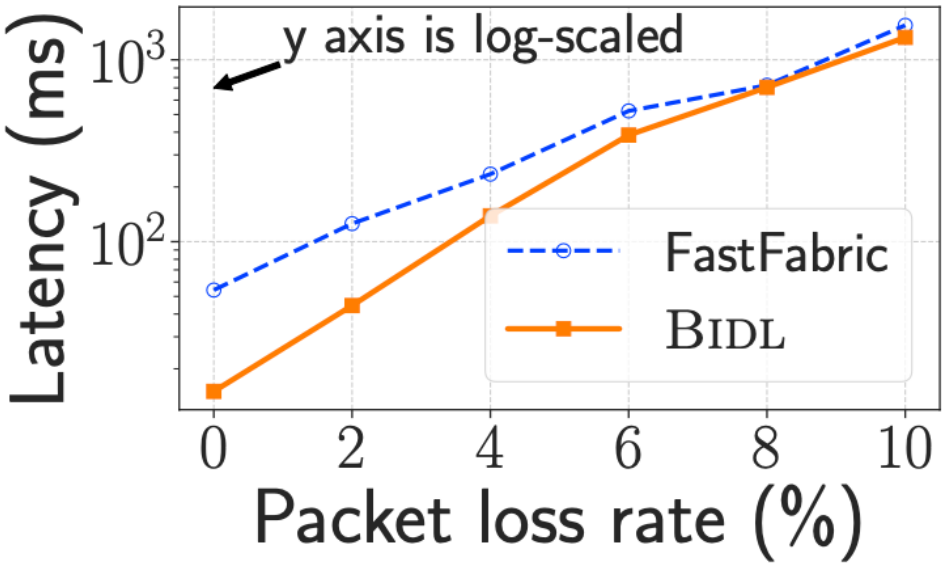
(b) End-to-end Latency.

How is BIDL's performance over multiple datacenters

Performance of BIDL and HLF with packet losses



(a) End-to-end throughput.



(b) End-to-end latency.

Thank You