A Sharding-based Hierarchical Blockchain for Large-Scale Transaction Processing

Yongrae Jo

Doctoral Thesis Defense

Pohang University of Science and Technology
Department of Computer Science and Engineering
System Software Laboratory

Dec 10. 2024

Thesis Outline

Background & Challenges

Proposed Systems:

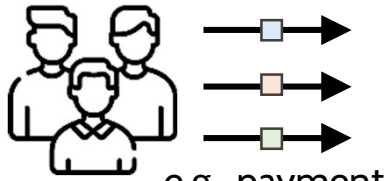
- DyloChain
- PyloChain

Discussion & Conclusion

Blockchain

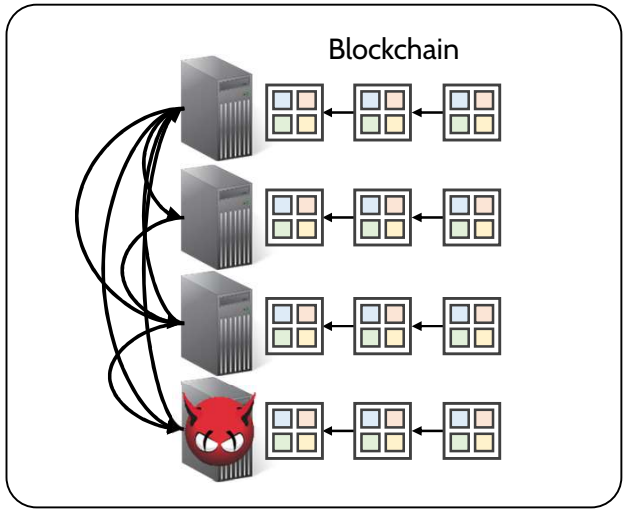
Users

TXs



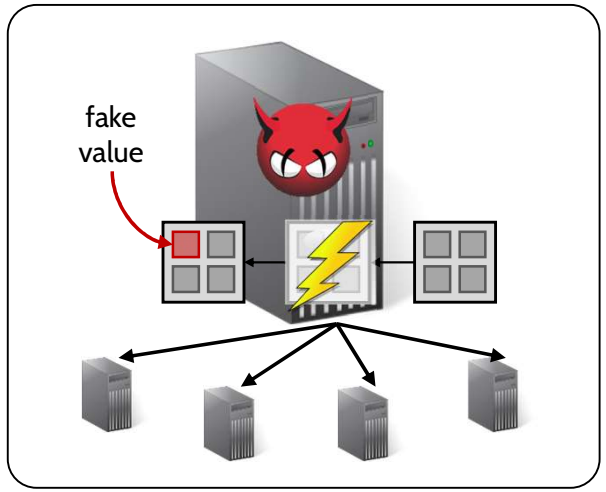
e.g., payment

Blockchain Network



- Decentralization (Consensus)
- Availability
- (Byzantine) Fault tolerance
- Tamper-resistance
- Transparency

Traditional Centralized Network

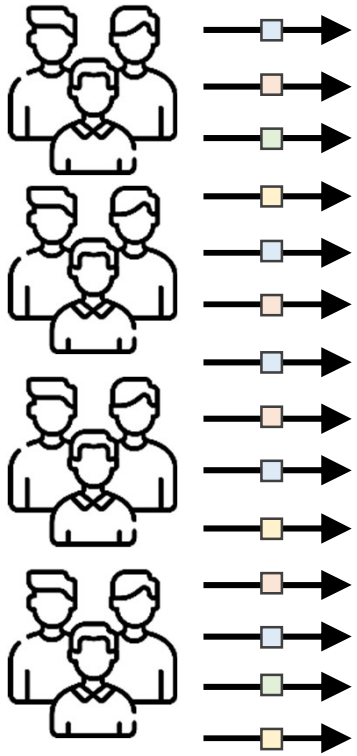


- Centralization
- Single point of failures

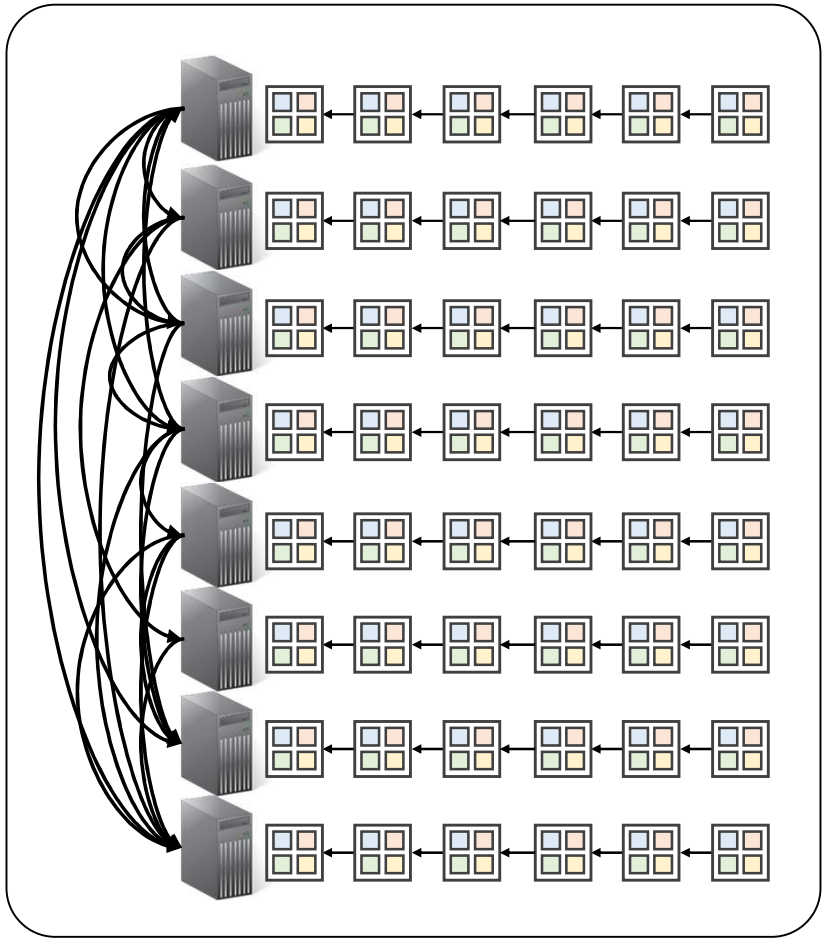
Scaling Blockchain

A **Larger** Userbase

TXs



Larger Blockchain Network



Higher overhead & Lower Performance

Limiting Blockchain Apps.

Examples:
- Visa: **avg. 8.5k, ~65k TPS** [1)]
- Hyperledger Fabric: **~3k** [2)]

Blockchain Scalability Matters !

1) https://thebankingscene.com/opinions/visas-innovations-enhancing-travel-and-payment-experiences-worldwide
2) https://www.lfdecentralizedtrust.org/blog/2023/02/16/benchmarking-hyperledger-fabric-2-5-performance

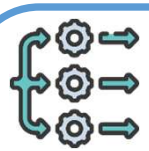
Blockchain Scalability Techniques

Application Layer
smart contract, Asset

Off-Chain Layer
PCN, roll-ups

Cryptography Layer
Signature, ZKP

Core Technologies



Execution Layer
Parallelism and Concurrency

Consensus Layer
PoW/PoS/BFT, **Sharding**

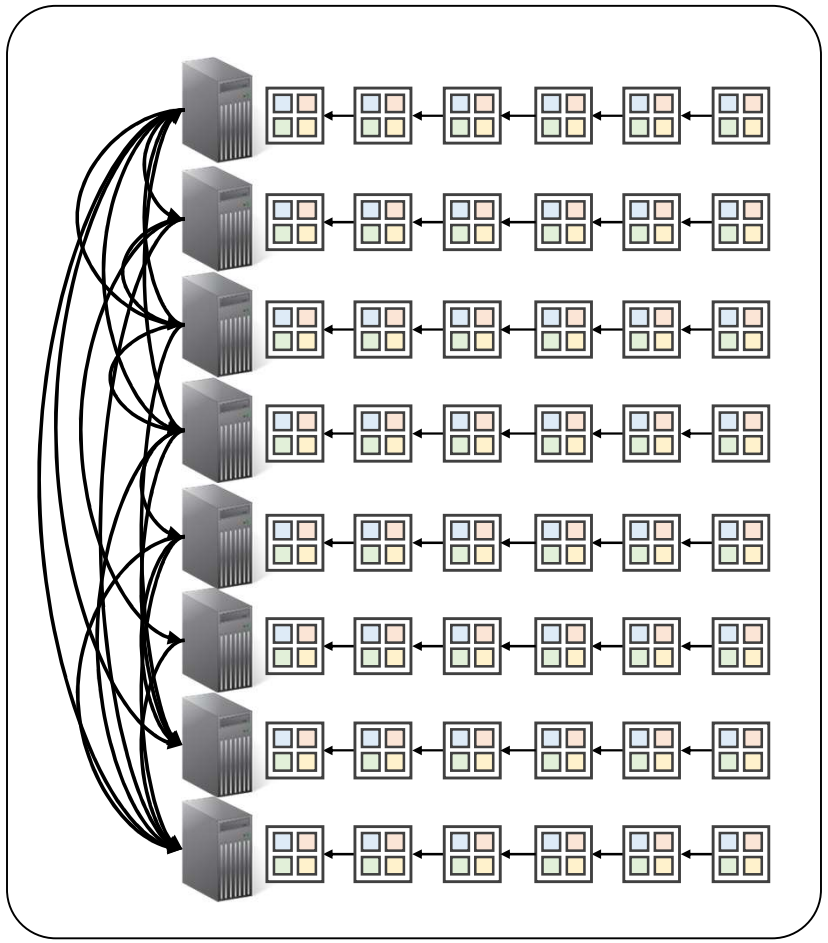
Thesis's focus

Network Layer
Topology, Gossip

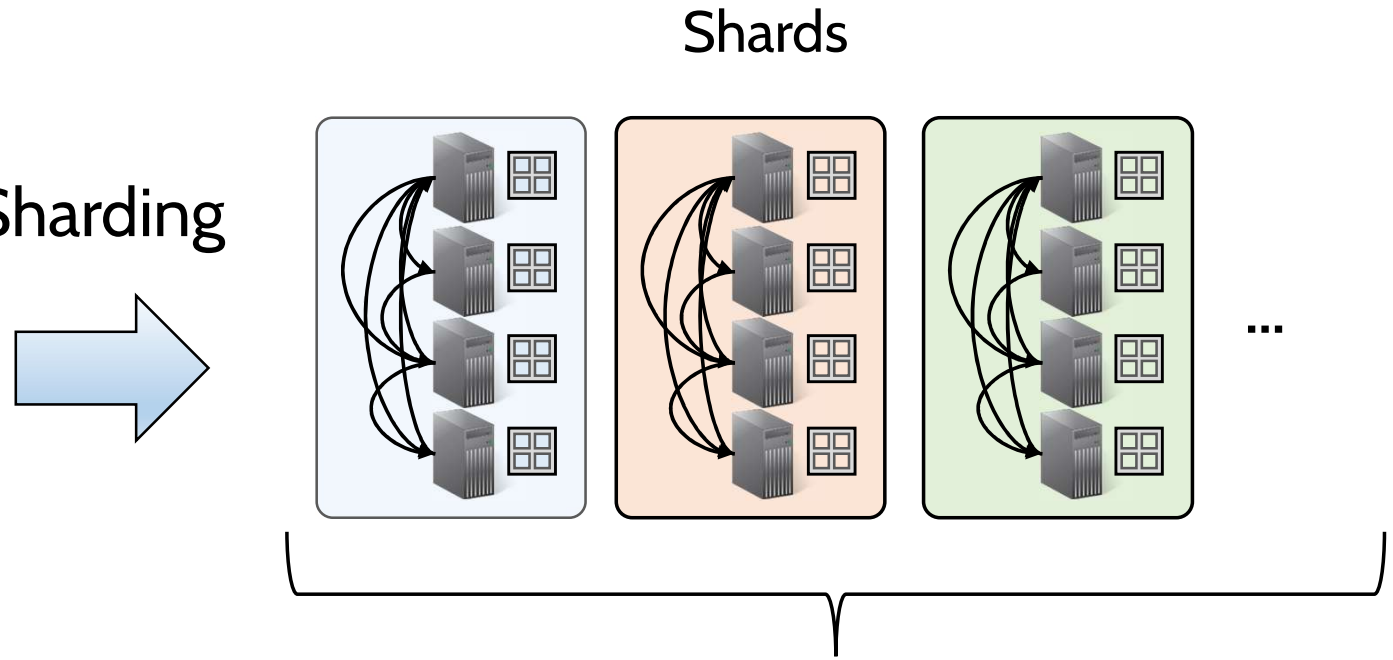
Storage Layer
DB, LSM Tree

Hardware Layer
CPU, Mem, TEE

Blockchain Sharding



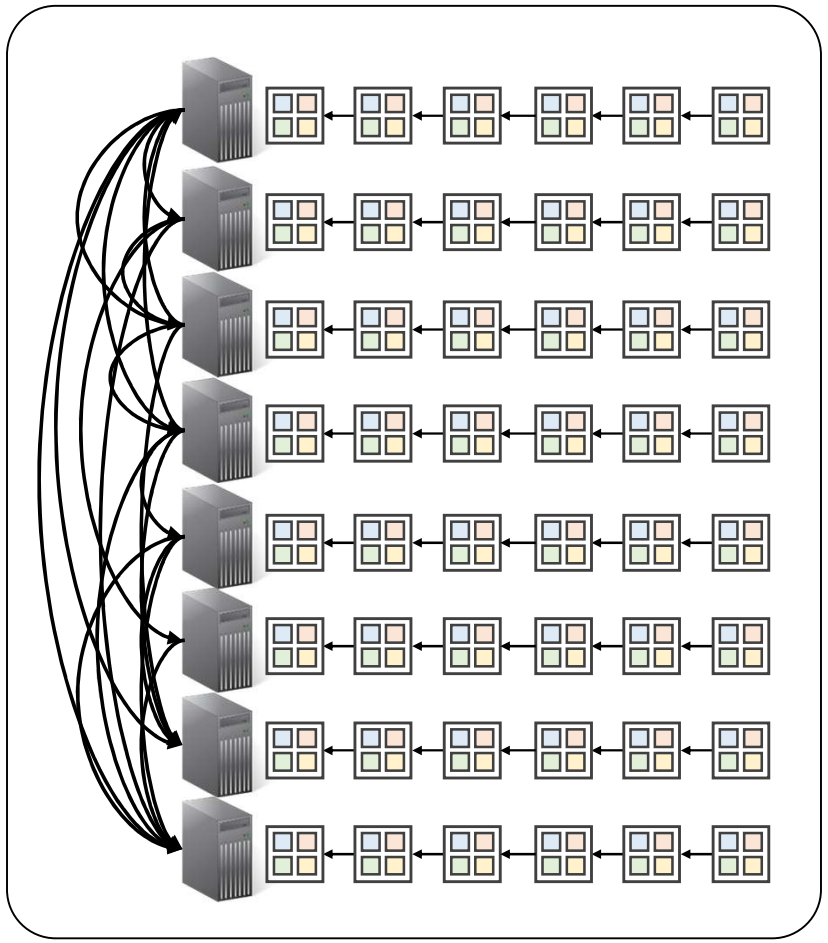
Sharding



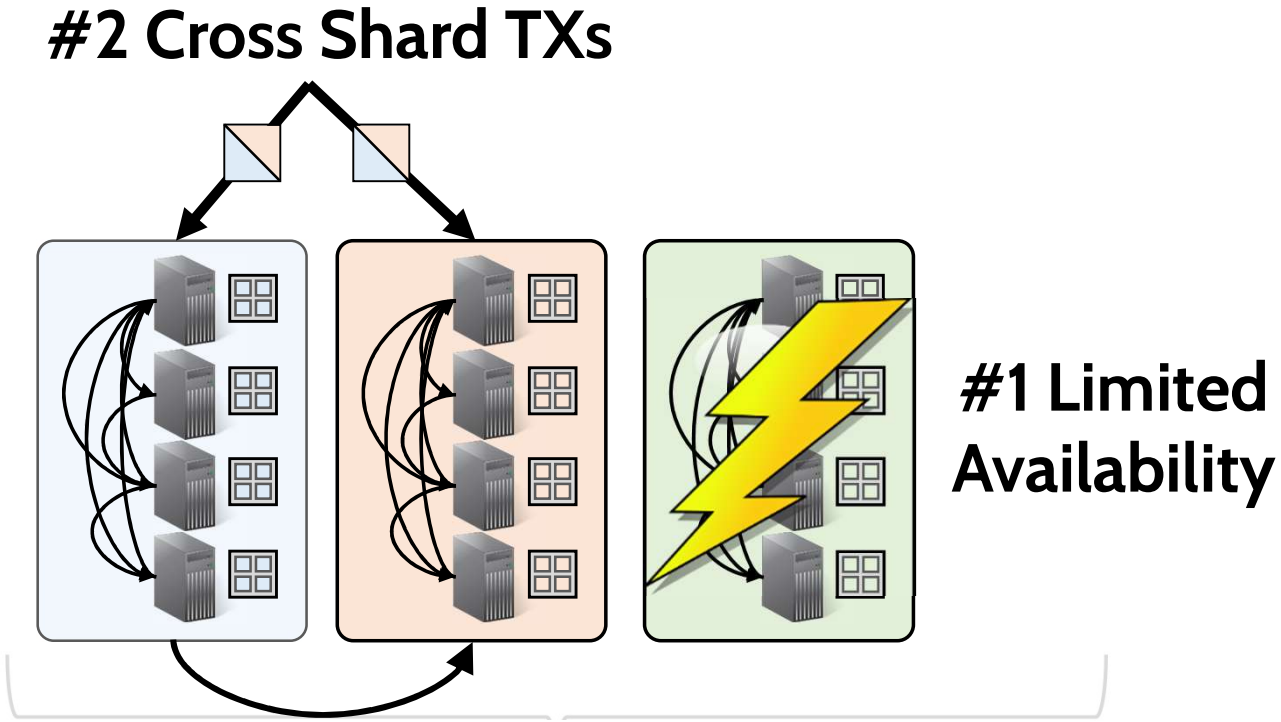
Horizontal Scalability

- Higher throughput
- Lower latency

Blockchain Sharding: Challenges



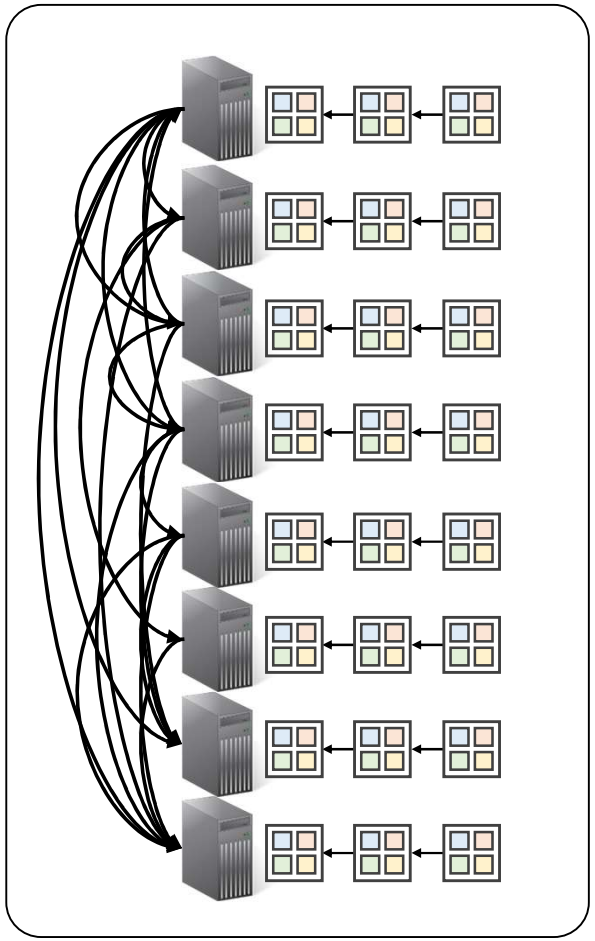
Sharding



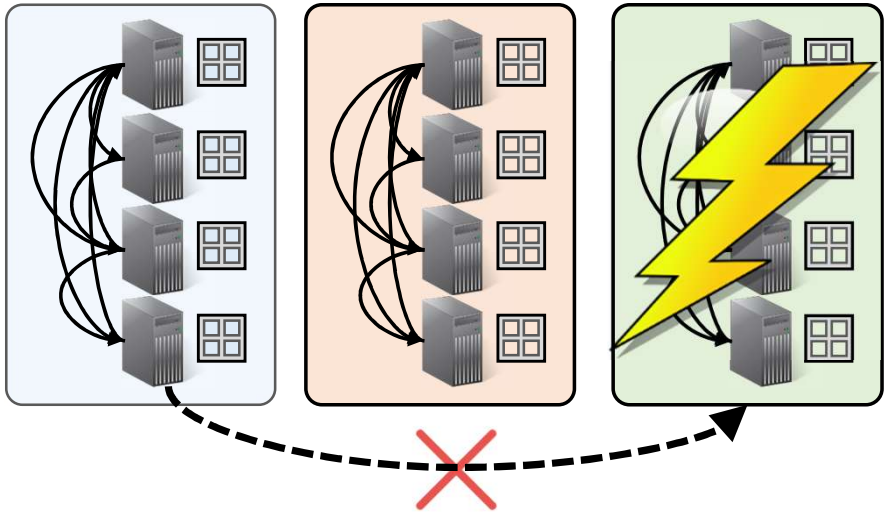
Horizontal Scalability
- Higher throughput
- Lower latency

Challenge #1: Overcoming Limited Availability

Full Availability



(Performance) Sharding

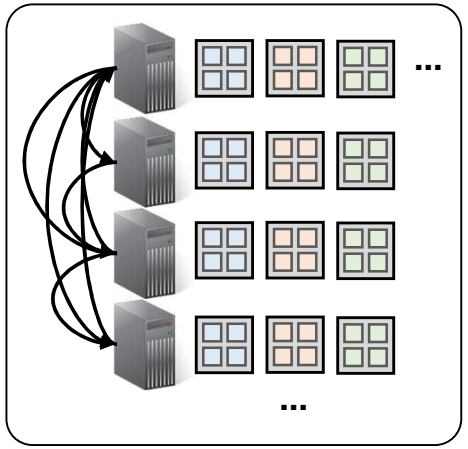


e.g., Bad software updates[1], Fire[2], Earthquake[3]

1) https://www.govtech.com/security/cybersecurity-update-causes-worldwide-microsoft-outages
2) https://www.seoul.co.kr/news/economy/IT/2022/10/15/20221015500039
3) https://www.datacenterdynamics.com/en/news/more-than-170-base-stations-offline-in-taiwan-following-earthquake/

⇨ Limited availability

Availability Sharding



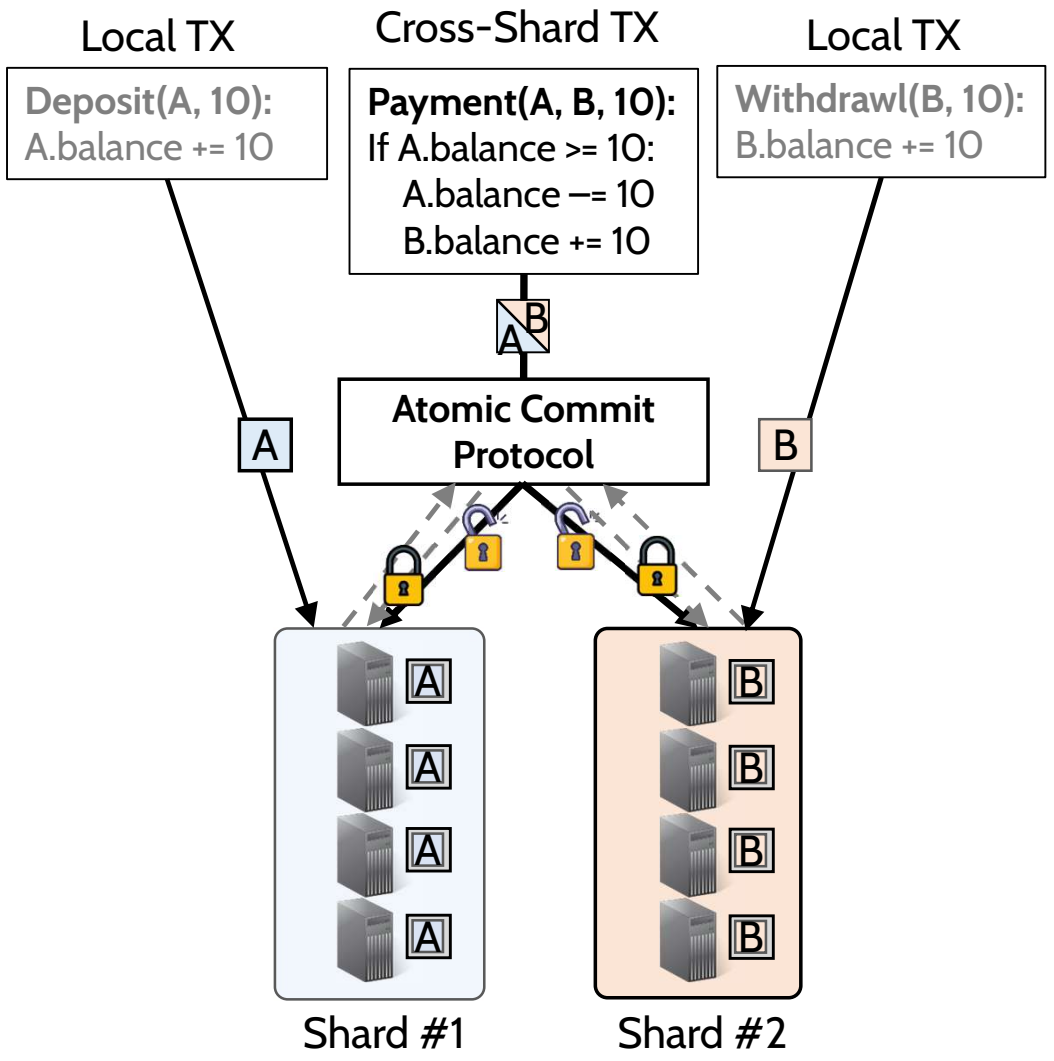
Parallel blockchain instances within each member

- Manages a full copy of shards, handling a shard failure
- Meepo (ICDE'22), OHIE (CCS'18)

⇨ Higher overhead on each member

⇨ Limited scalability

Challenge #2: Efficient Handling of Cross Shard TXs



Atomic Commit Protocol

Two phase commit (2PC)
- Coordinator-driven (e.g., client) complex locking mechanism
- e.g., OmniLedger (SP'18), AHL (SIGMOD'19)

⇨ Poor performance. Hard to guarantee correctness under failures

Flattened approach
- No coordinator; All involved members directly communicate
- e.g., Sharper (SIGMOD'21)

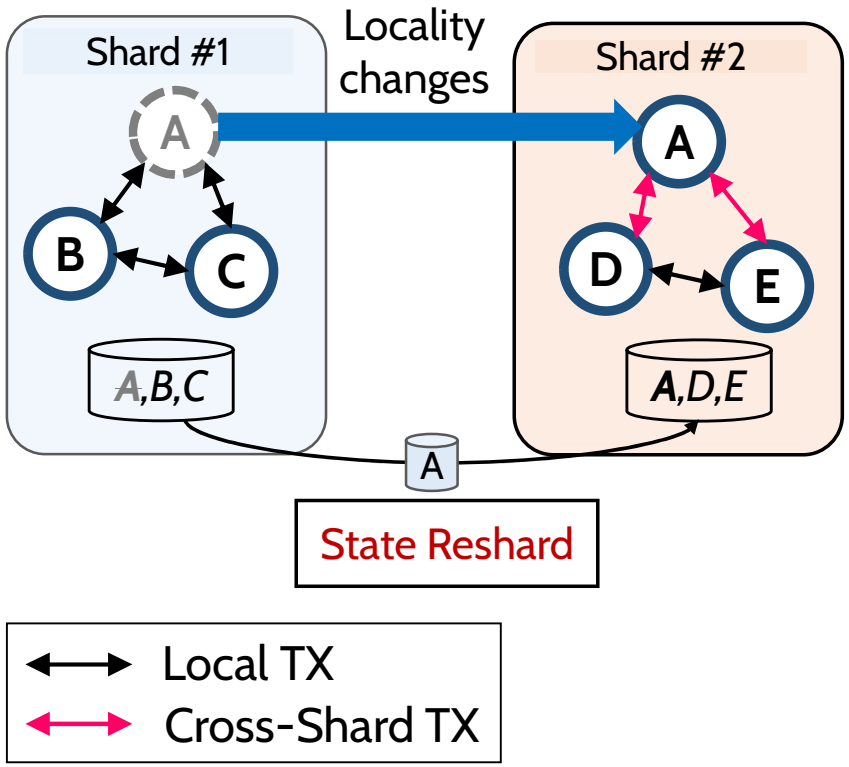
⇨ Very high communication cost

Sub transactions
- Split into the cross-shard TX into multiple sub TXs
- Monoxide (NSDI'19), RapidChain (CCS'18)

⇨ Higher overhead (as in Input TX, output TX, relay TX)

Challenge #3: Supporting Dynamic Locality

Locality may change over time
→ # of cross-shard TXs increases over time



Need to support State Reshard

Moving a state between blockchains is not simple
- One ownership at a time
- Latest & Correct
- User's consent

Moving states into a single shard when processing cross-shard TX or complex mathematical analysis of historical pattern
- without user's consent
- Ziziphus (ICDE'23), BrokerChain (INFOCOMM'22)

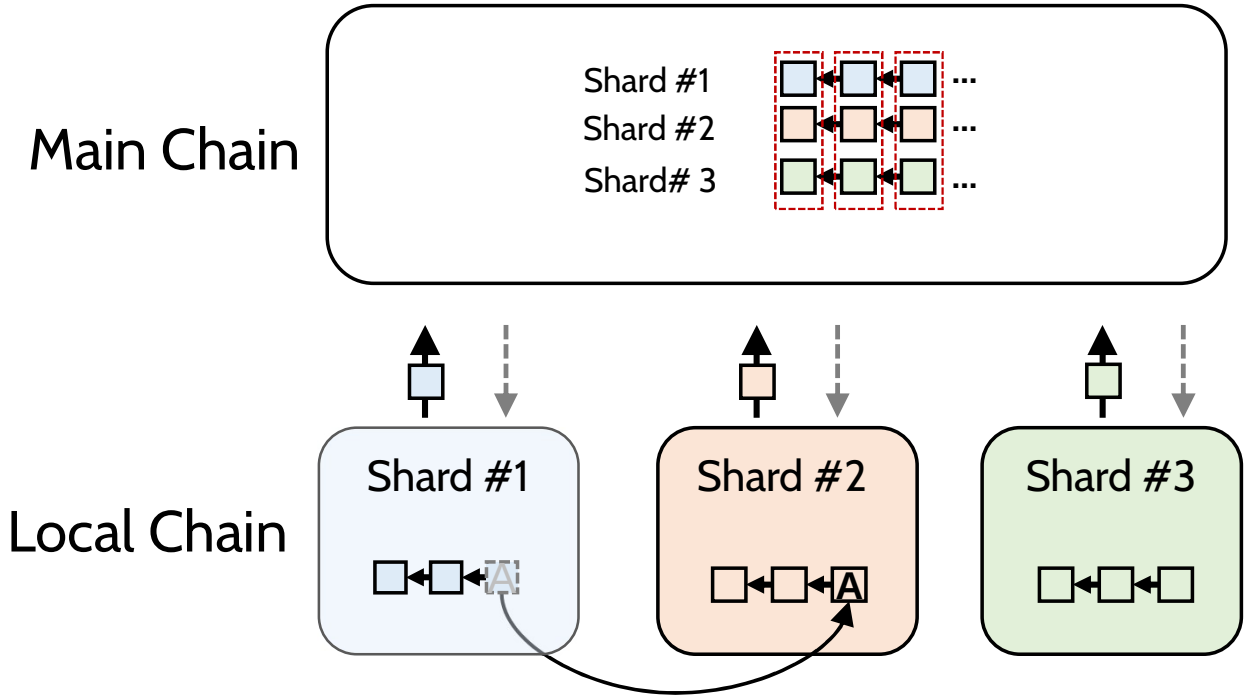
⇨ Arbitrarily moving user's state only for efficiency; unrealistic

State from limited availability local chain
- Data from a source local chain may not be availability
- Saguaro (ICDE'23)

⇨ Need a trustworthy data source

Our Approach & Contributions

Sharding-based Hierarchical Blockchain



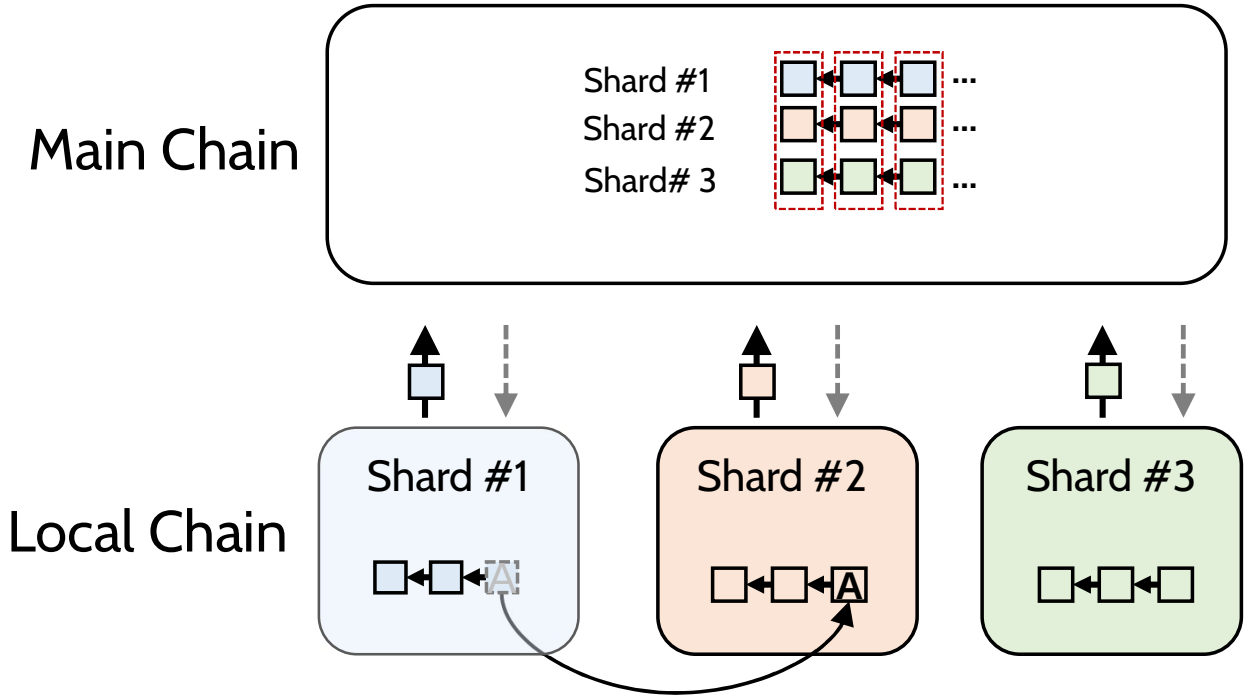
Full copy of local shards in Main Chain
(Challenge #1)

Efficient handling of cross-shard TXs within collocated shards in Main Chain
(Challenge #2)

Dynamic Locality via State Reshard based on Main Chain and Reshard TX
(Challenge #3)

Our Approach & Contributions

Sharding-based Hierarchical Blockchain



DyloChain

- Synchronous Main Chain
- Order-Execute-Order-Validate (O-X-O-V)
- State Reshard Protocol

PyloChain

- Partially Synchronous Main Chain
- Improving Main Chain Scalability
- Enhancing Cross Shard TX Efficiency

Our Approach & Contributions

Sharding-based Hierarchical Blockchain

DyloChain
- Synchronous Main Chain
- Order-Execute-Order-Validate (O-X-O-V)
- State Reshard Protocol

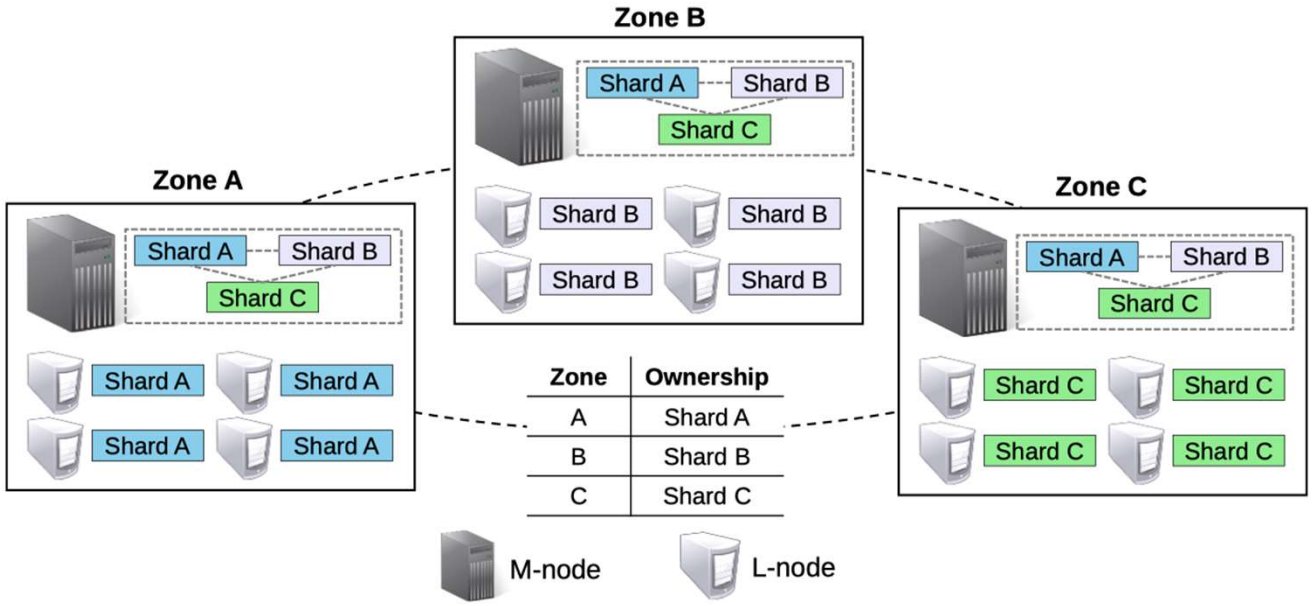
PyloChain
- Partially Synchronous Main Chain
- Improving Main Chain Scalability
- Enhancing Cross Shard TX Efficiency

DyloChain: Assumptions and Model

- **Permissioned**
  - Known identities with a public/private key pair
- **Network**
  - Main Chain: Synchronous communication
  - Local Chain: Partially synchronous communication
- **Failure**
  - Main Chain: f_F out of $2f_F + 1$
  - Local Chain: f_L, out of $3f_L + 1$
- **Transaction**
  - Read Set = A set of (key, value, version)
  - Write Set = A set of (key, value)

(BlockNumber, TXOffset)

DyloChain: An Overview



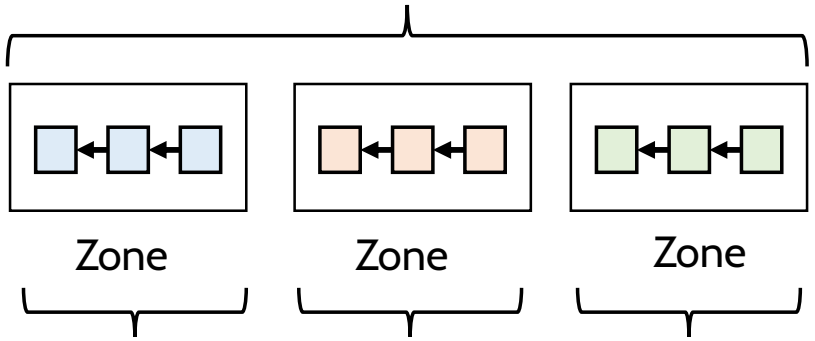
Nodes
- M-node for a full copy of shards
- L-node for a single copy of a shard

Ownership
- System-wide metadata for tracking user's locality
- Only one ownership at any given time

Transaction Types
- Local TX: All same ownerships
- Global TX: Different (or remote) ownerships
- Reshard TX: Changing ownership

O-X-O-V Transaction Processing: Why?

Mostly mutually independent across zones



High locality within a zone

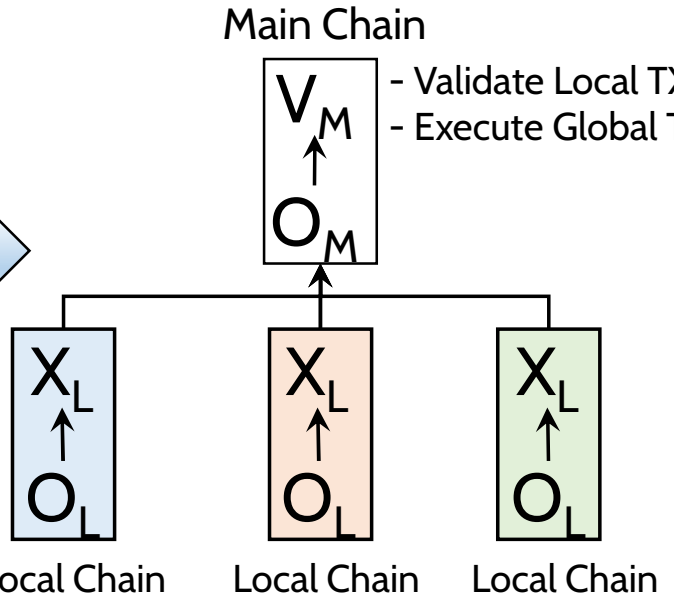
Execute-Order-Validate (X-O-V)

- Support highly concurrent executions
- Good performance under less conflicts

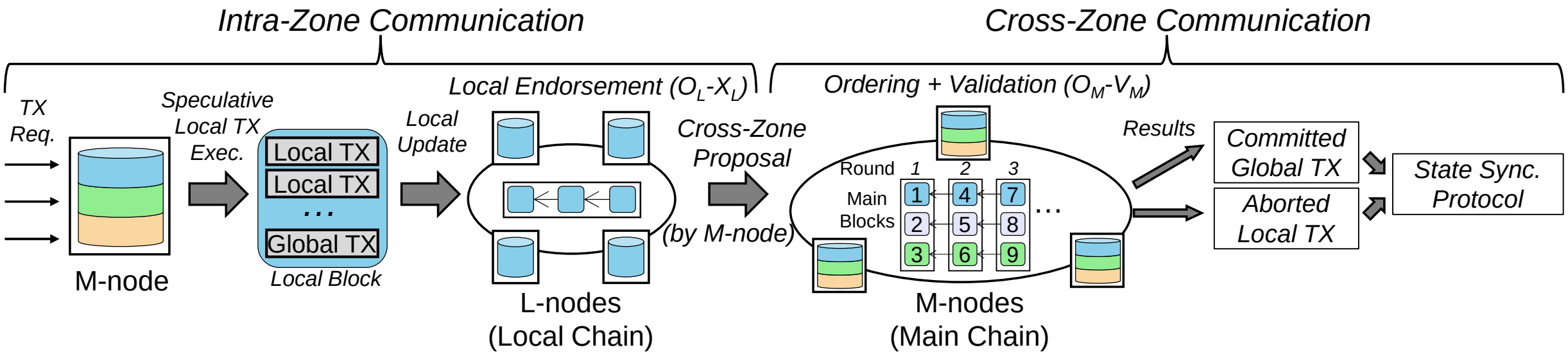
Order-Execute (O-X)

- Serialize First, no conflicts at all
- Better under high conflicts

Order-Execute-Order-Validate (O-X-O-V)

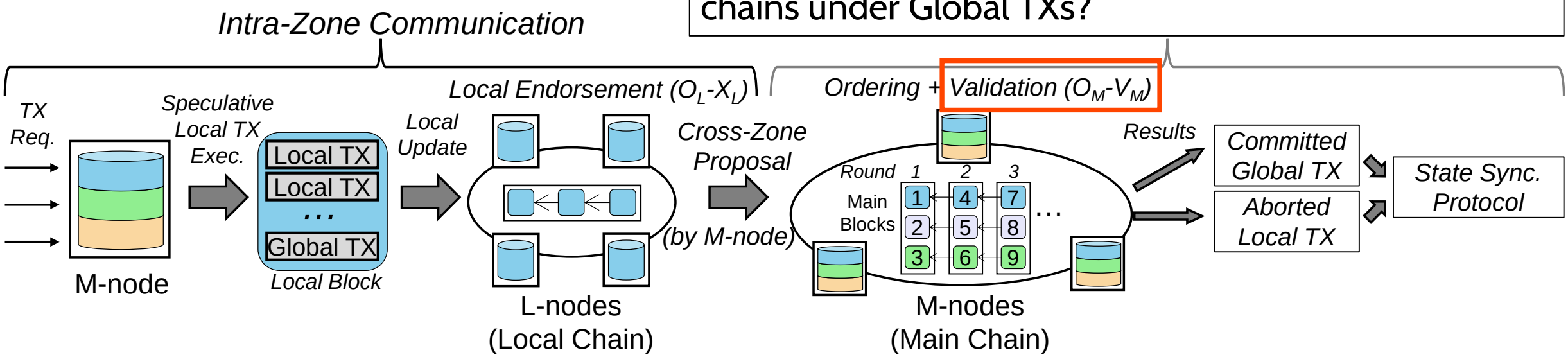


DyloChain: Transaction Processing Flow



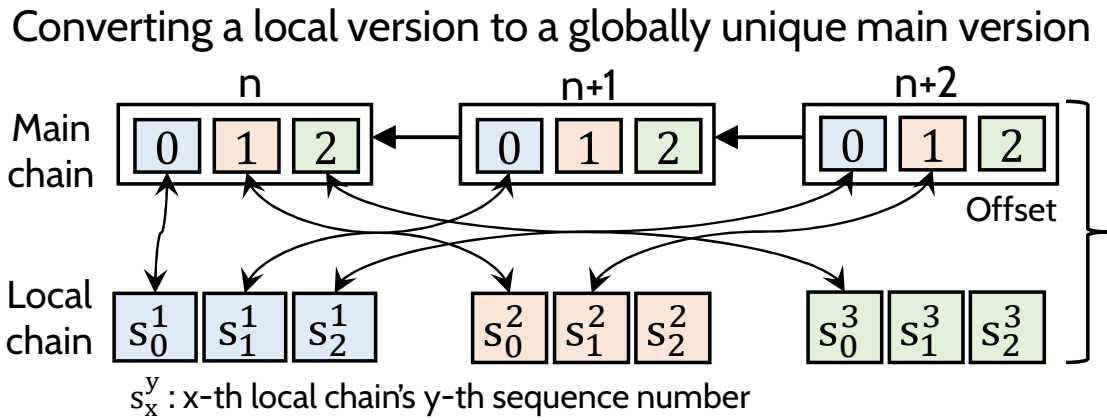
DyloChain: Transaction Processing Flow

Independent local ordering with its own sequences.
→ How to consistently validate Local TXs across local chains under Global TXs?



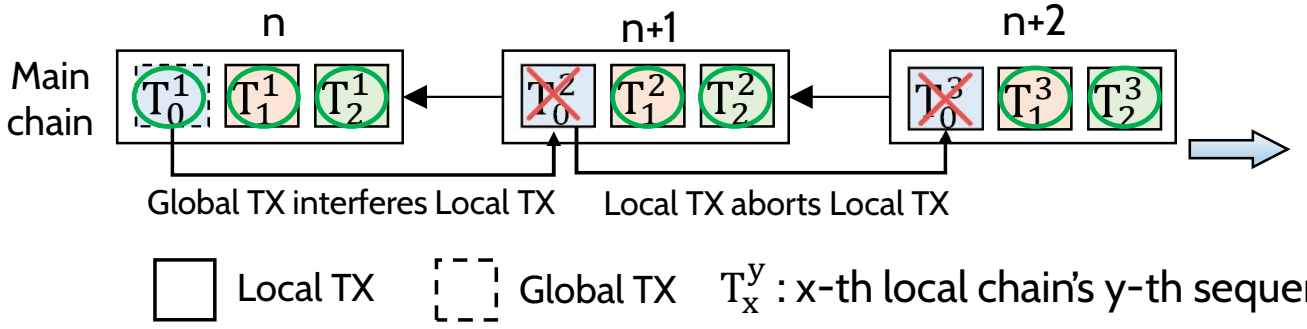
Transaction Validations

- Consistent transaction validation across local chains with Version Map



Local-Version	Main-Version
s_0^1	(n,0)
s_1^1	(n+1,0)
s_2^1	(n+2,0)
s_0^2	(n,1)
...	

- Then, multi-version concurrency control (MVCC)-based validation
  → generates aborted local TXs

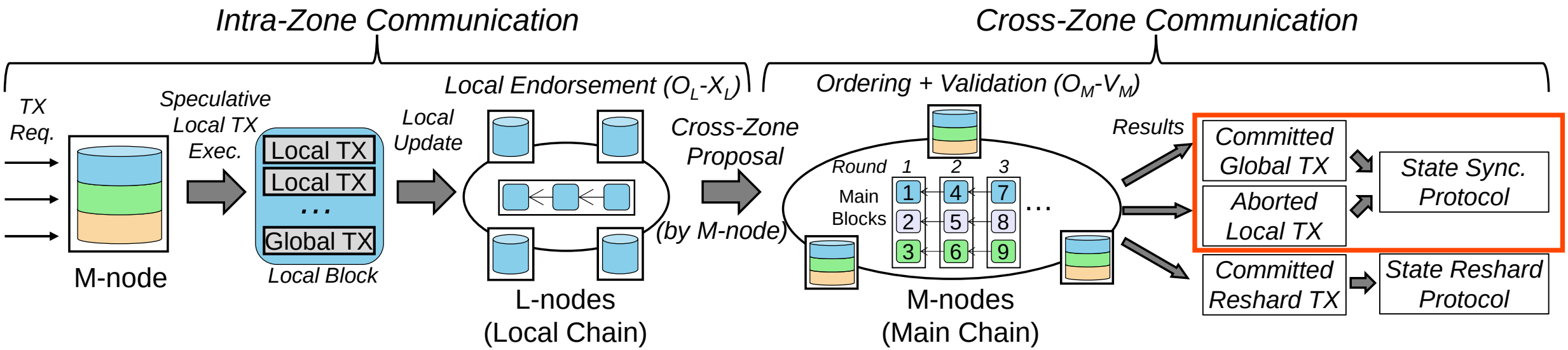


State Sync. for consistency between local/main chain

Algorithm 1: State Synchronization Protocol

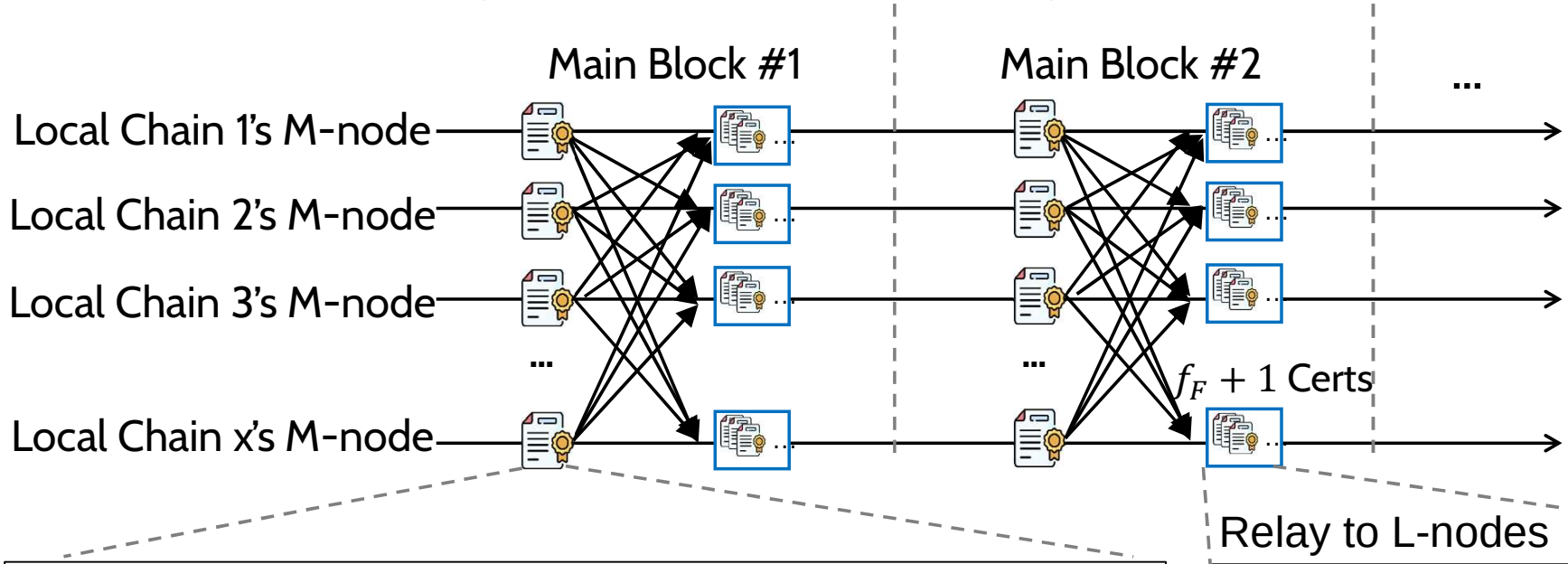
1 **on receiving** *i-th main block* B_i^M *at* $p \in \Pi_M^z$:
2 $s_I \leftarrow$ identify interfered states from committed global Txs and aborted local Txs from B_i^M
3 $s_M \leftarrow$ retrieve corresponding main states from s_I
4 $\pi_M(i) \leftarrow$ generate a main block proof from B_i^M and (s_I, s_M)
5 $\pi_{MQ}(i) \leftarrow$ exchange $\pi_M(i)$ among Π_M and derive a quorum certificate for the B_i^M
6 broadcast <SYNC, $\pi_{MQ}(i)$, s_I^z, s_M^z> to $p \in \Pi_L^z$
7 **on receiving** <SYNC, $\pi_{MQ}(i)$, s_I^z, s_M^z> at $p \in \Pi_L^z$:
8 verify $\pi_{MQ}(i)$; verify $D(s_I^z)$ and $D(s_M^z)$ by comparing the corresponding ones in $\pi_{MQ}(i)$
9 identify the involved states from s_I^z and commit the corresponding main states from s_M^z

DyloChain: Transaction Processing Flow



State Synchronization Protocol

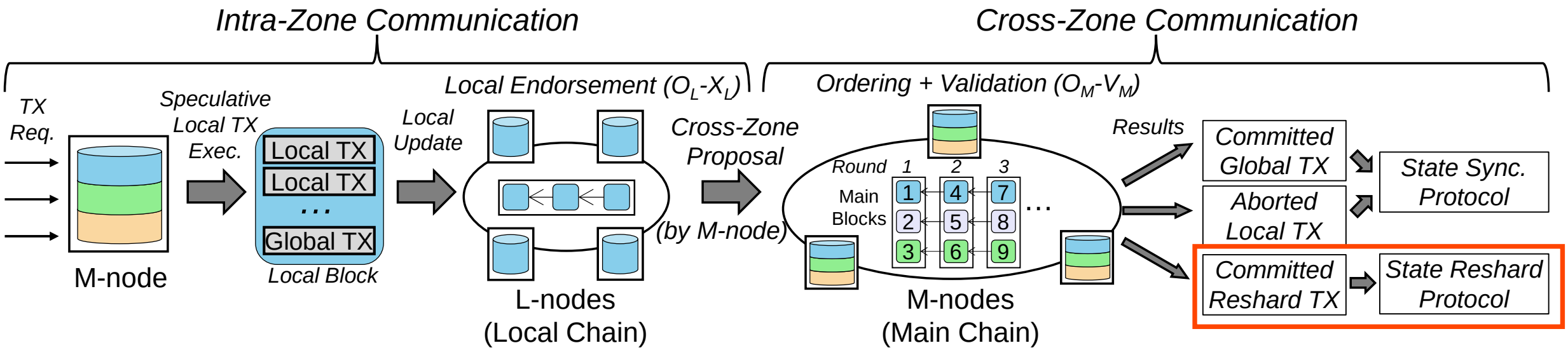
Invoked after processing a main block, reliably synching results into local chains



Main Block Processing Certificate (signed)
- Sync entries (from committed global TXs and aborted local TXs)
- Main block info (number, hash, previous hash)
- An ordered list of local block headers

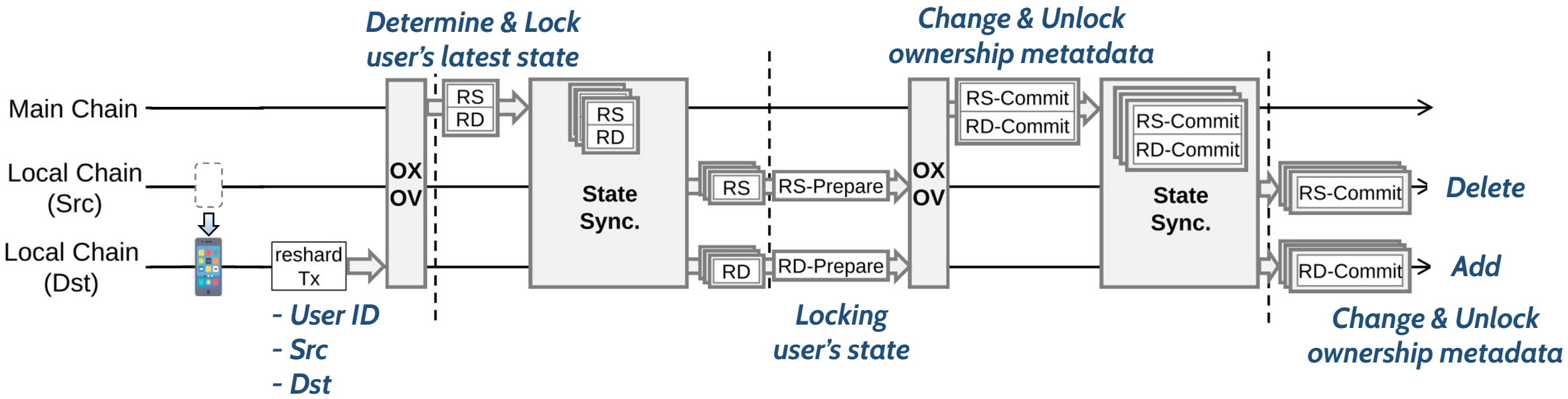
L-nodes's handling:
- Verify & Commit the sync entries
- Notify clients

State Reshard Protocol



State Reshard Protocol

- **Transfer user's state ownership**
  - From source to destination zone, reducing global (or cross-shard) TXs
- **Reshard TX:** A user-issued special TX for their locality changes



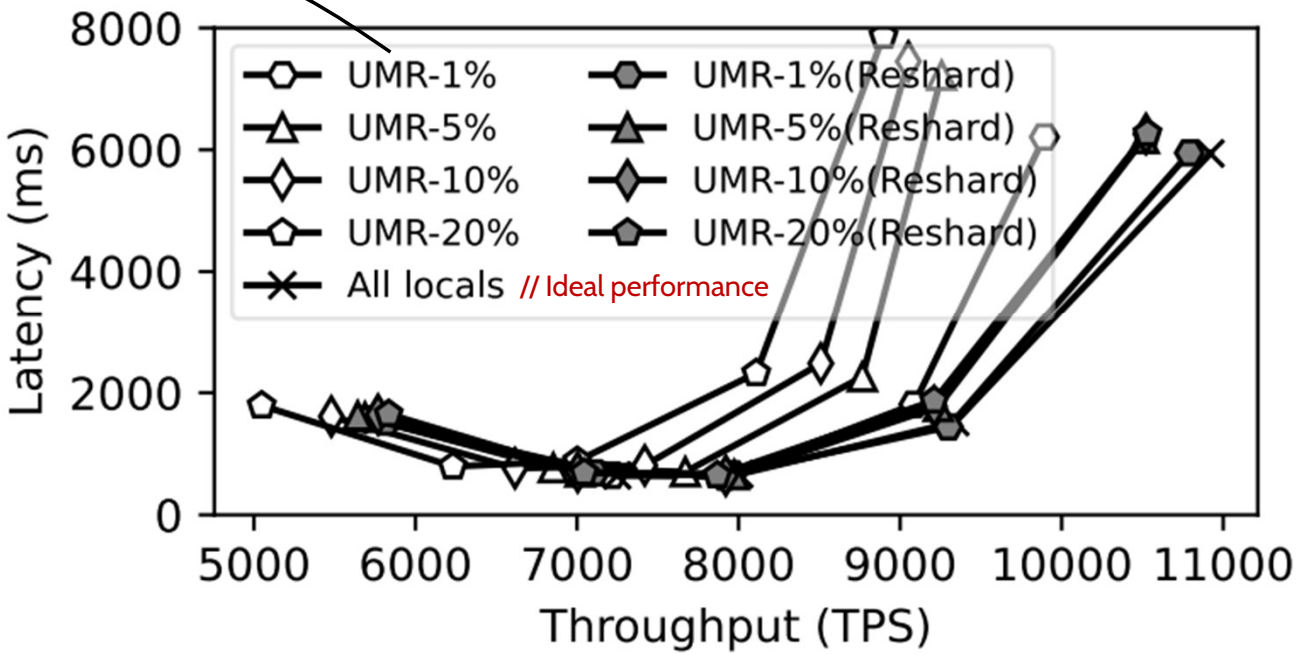
State Reshard Protocol to handle reshard TX

Experimental Setup

- Implementation
  - Golang, Hyperledger Fabric v2, PBFT-based consensus for local chains
- Networks
  - 1 zone = 1 M-node + 4 L-nodes
  - 3 ~ 15 zones
- 24 machines (in-lab cluster)
  - 64/32/16 CPUs, 256/128/64 GiB RAM, 10Gbps
- Workload
  - SmallBank with 30,000 accounts (equally distributed & uniformly accessed), 60 secs, 1 TX: 2.89KB
  - Global TX: by controlling user mobility ratio (denoted by UMR); locality changes after 20 secs.

Evaluation: Overall Performance

- 3 zones with 15 members with local block size: 100 TXs
- Global TXs easily saturate DyloChain, but state reshard effectively mitigates

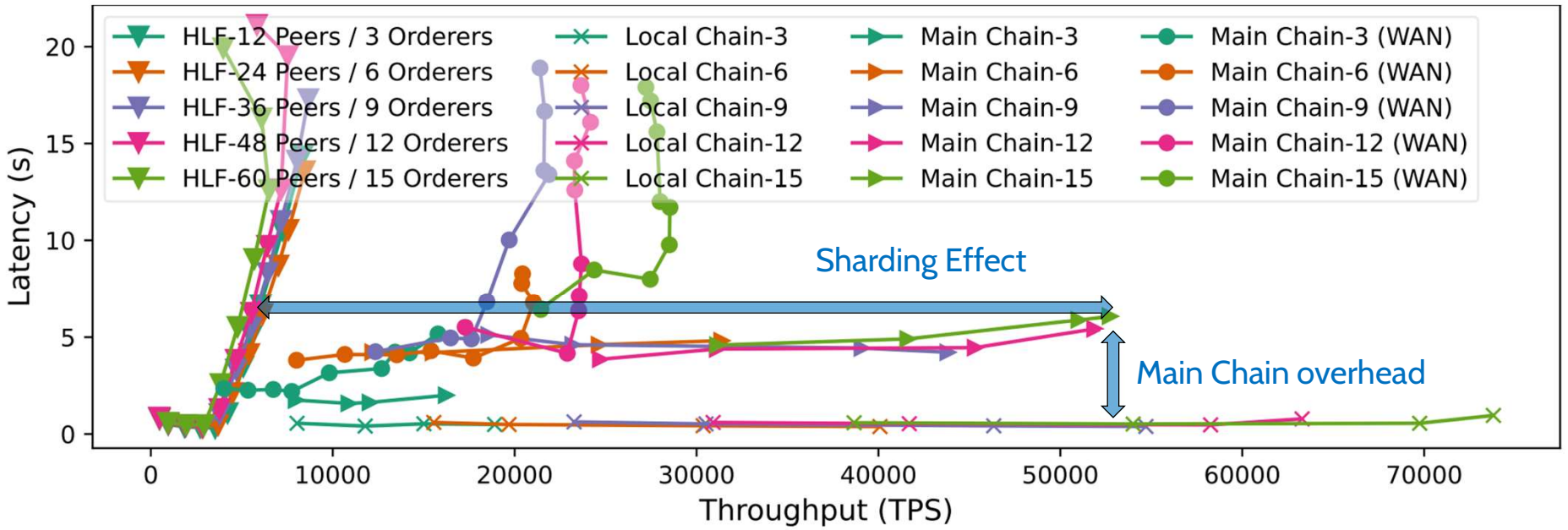


Main Chain Performance

Evaluation: vs. HLF and WAN

- Scaling up to 15 zones with 75 nodes with local block size: 500 TXs
- Hyperledger Fabric (HLF) with the same scale/config.
- Wide Area Network (Simulation based on AWS inter-region latencies*)

All local TXs



*cloudping.co

Our Approach & Contributions

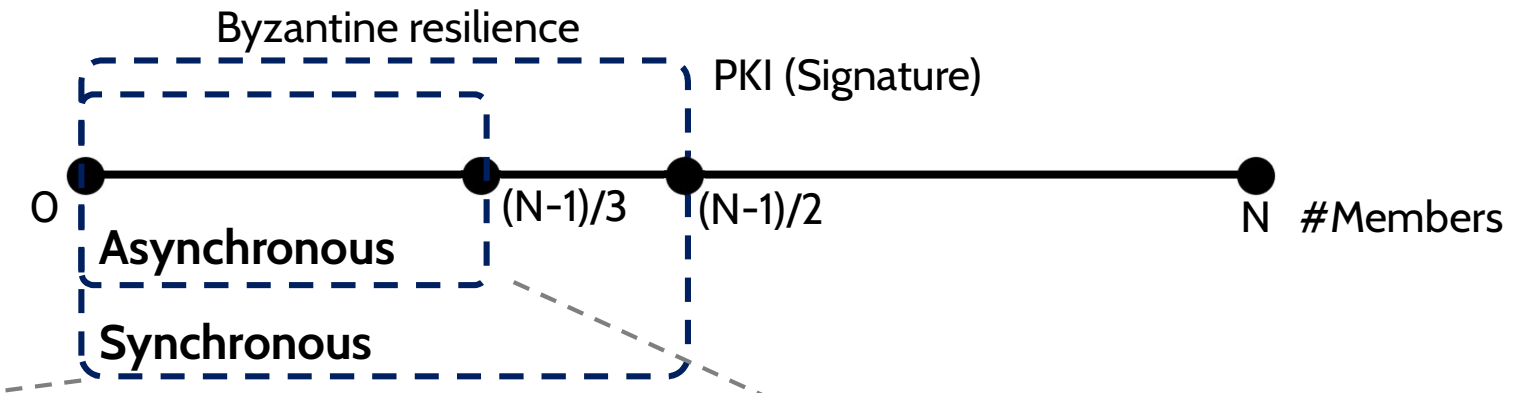
Sharding-based Hierarchical Blockchain

DyloChain
- Synchronous Main Chain
- Order-Execute-Order-Validate (O-X-O-V)
- State Reshard Protocol

PyloChain
- **Partially Synchronous Main Chain**
- Improving Main Chain Scalability
- Enhancing Cross Shard TX Efficiency

Distributed Network Assumptions for Main Chain Design

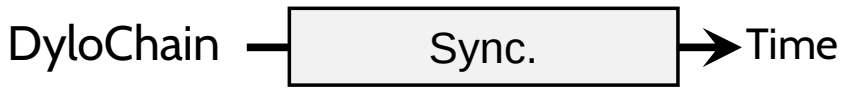
The Byzantine Generals Problem, 1982



Synchronous

Reliable network (known msg. delay bound)
-> Higher resilience, simple, but requires a precise timeout
-> Leveraging sophisticated latency prediction techniques[1)2)]

e.g., XFT (OSDI'16), VFT (EuroSys'15), SyncHotStuff (SP'20), Multi- BFT (CCS'21)

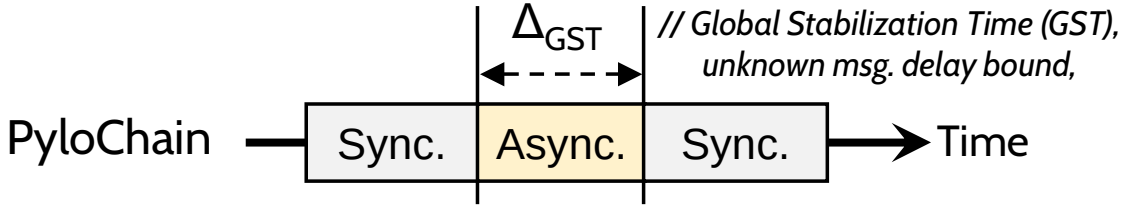


1) Swift: Delay is Simple and Effective for Congestion Control in the Datacenter (SIGCOMM'20)
2) Better never than late: meeting deadlines in datacenter networks (SIGCOMM'11)

Asynchronous

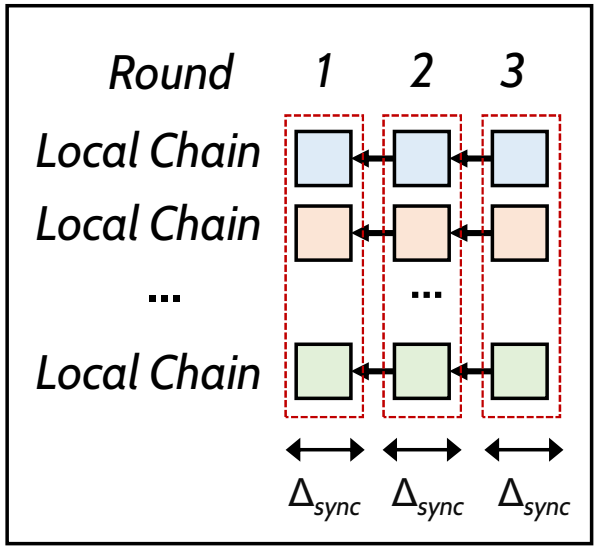
Unreliable network (no msg. delay bound)
-> More practical, but impossibility result for liveness (FLP)
-> Introduce only a **partially synchronous** network

e.g., PBFT (OSDI'99), Zyzzyva (SOSP'07), HotStuff (PODC'19), Bullshark (CCS'22)



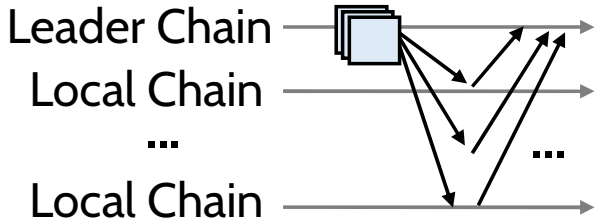
Redesigning Main Chain

Synchronous Main Chain



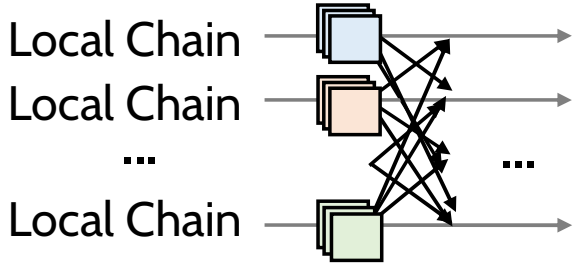
Partially Synchronous Main Chain: Design Choices

#1 Leader-based (e.g., PBFT)



- Leader bottleneck
- Complex leader change under failures

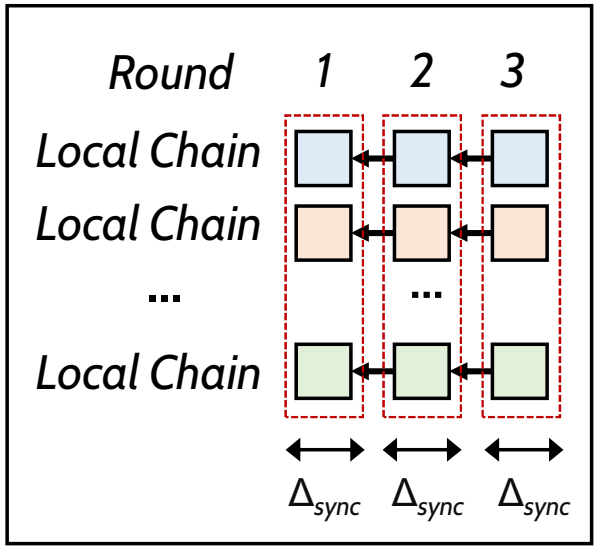
#2 Leader-less (e.g., Bullshark)



- Equal participations
- No leader change
- Highly concurrent

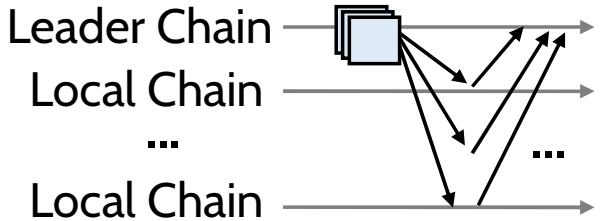
Redesigning Main Chain

Synchronous Main Chain



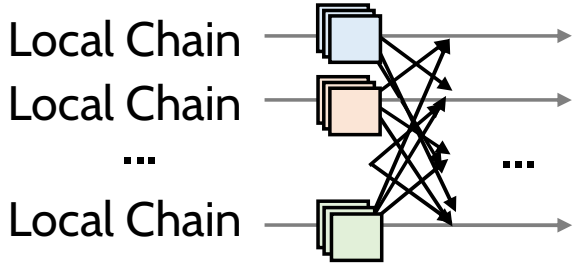
Partially Synchronous Main Chain: Design Choices

#1 Leader-based (e.g., PBFT)



- Leader bottleneck
- Complex leader change under failures

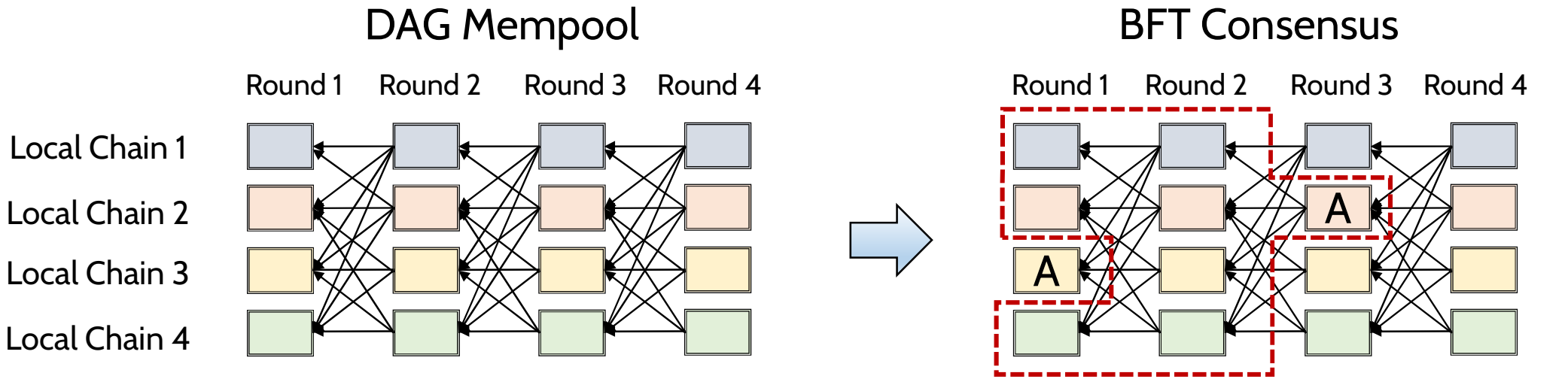
#2 Leader-less (e.g., Bullshark)



- **Equal participations**
- **No leader change**
- **Highly concurrent**

Designing Leader-less Main Chain

- **For a leader-less main chain, PyloChain chooses a DAG-based consensus**
  - Recent breakthrough in blockchain consensus scalability (e.g., 125 kTPS)
  - e.g., Narwhal/Tusk (EuroSys'22), Bullshark (CCS'22)
  - Idea: Separation of reliable block distributions and their efficient ordering

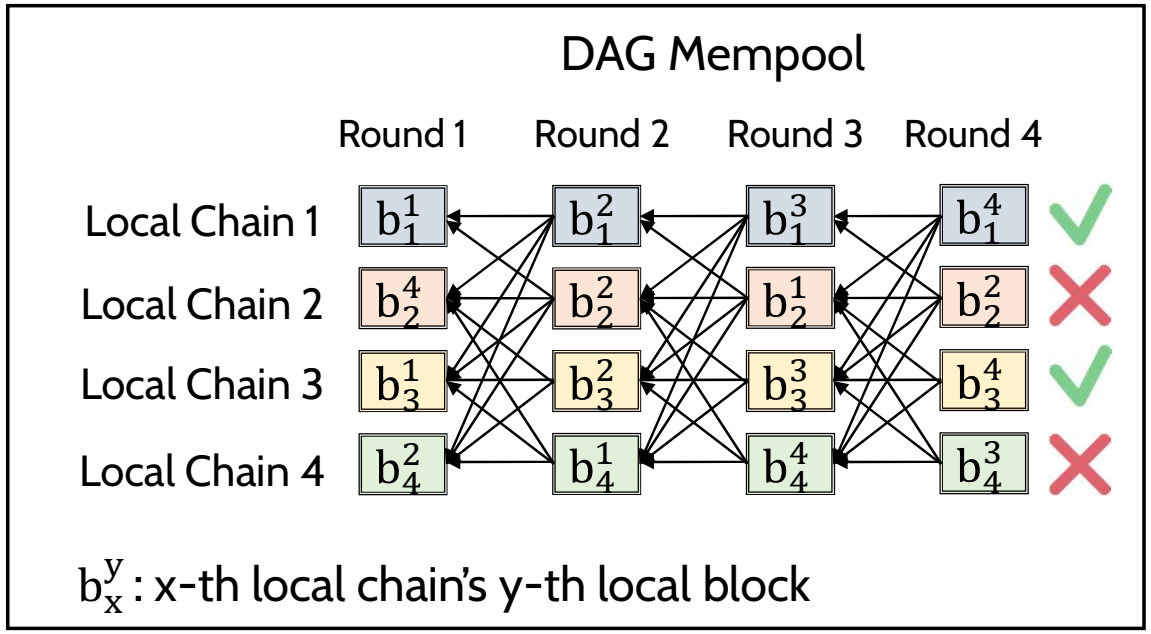


— Highly concurrent & parallel block distributions
— Guarantees local block availability

— Deterministic total order
— Efficient (only local computation)

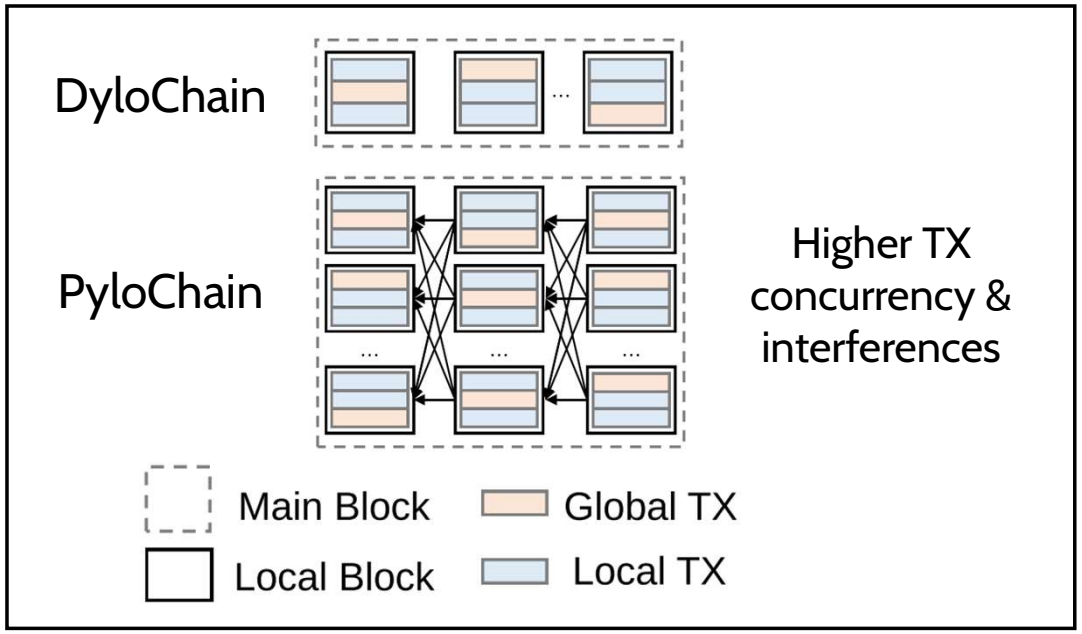
DAG-based Main Chain: Issues

#1 Disrespecting locally agreed-upon sequence



Making DAG mempool align local blocks with monotonically increasing order

#2 Larger main block size



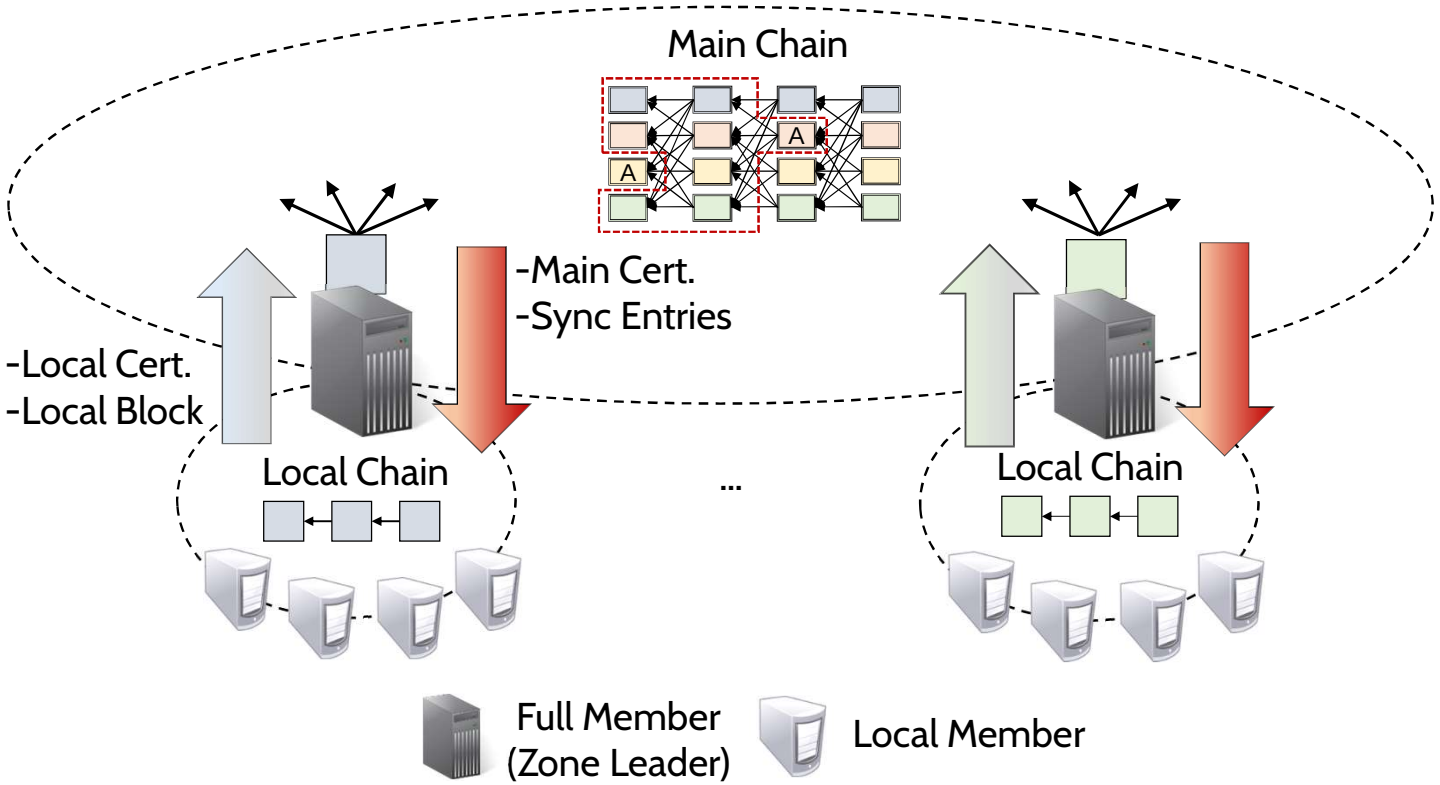
Scheduling Global TXs to avoid conflicts within a main block

PyloChain: Assumptions and Model

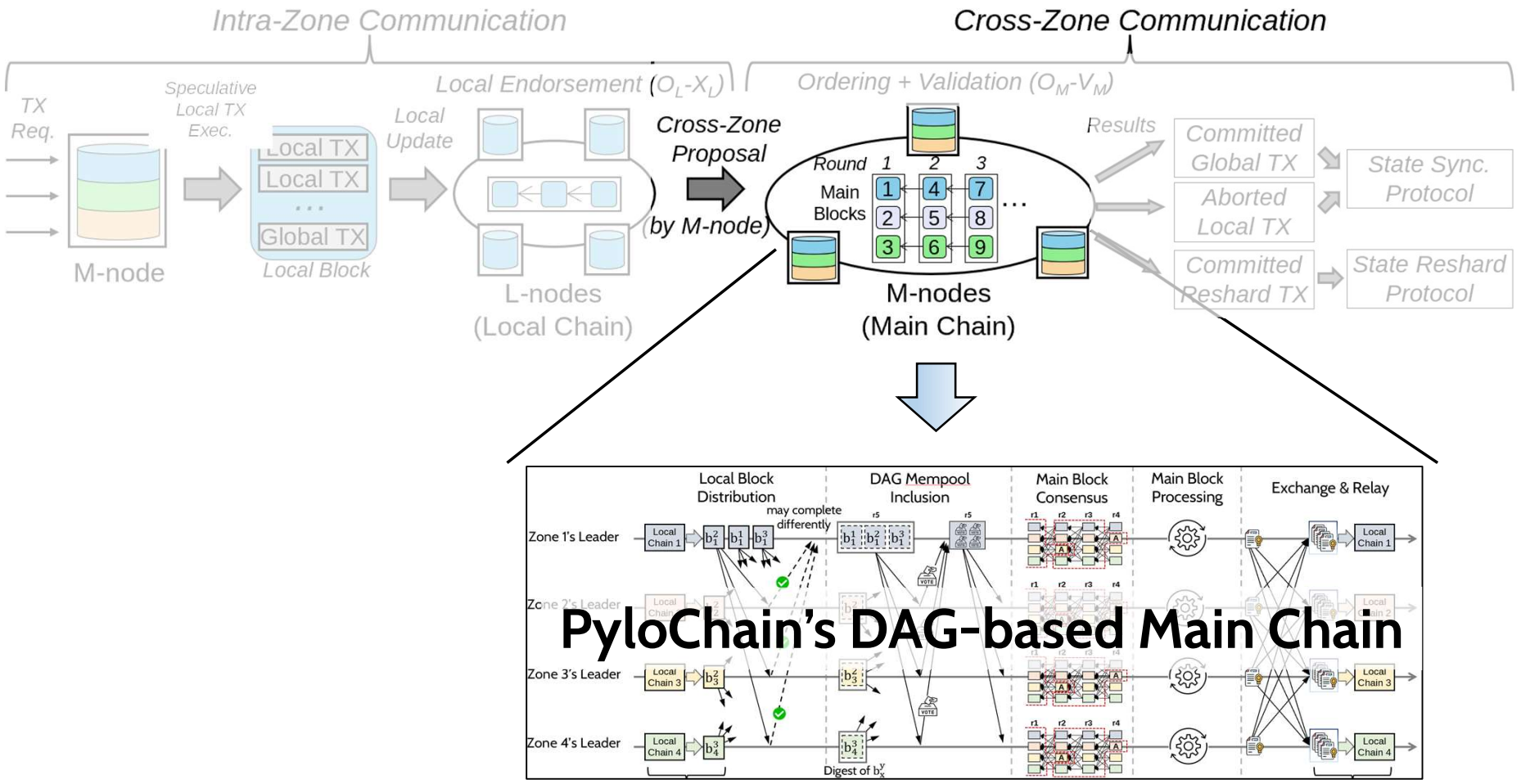
- **Permissioned**
  - Known identities with a public/private key pair
- **Network**
  - Main Chain: Partially Synchronous communication
  - Local Chain: Partially synchronous communication
- **Failure**
  - Main Chain: f_F out of $3f_F + 1$
  - Local Chain: f_L, out of $3f_L + 1$
- **Transaction**
  - Read Set = A set of (key, value, version)
  - Write Set = A set of (key, value)

version → (BlockNumber, TXOffset)

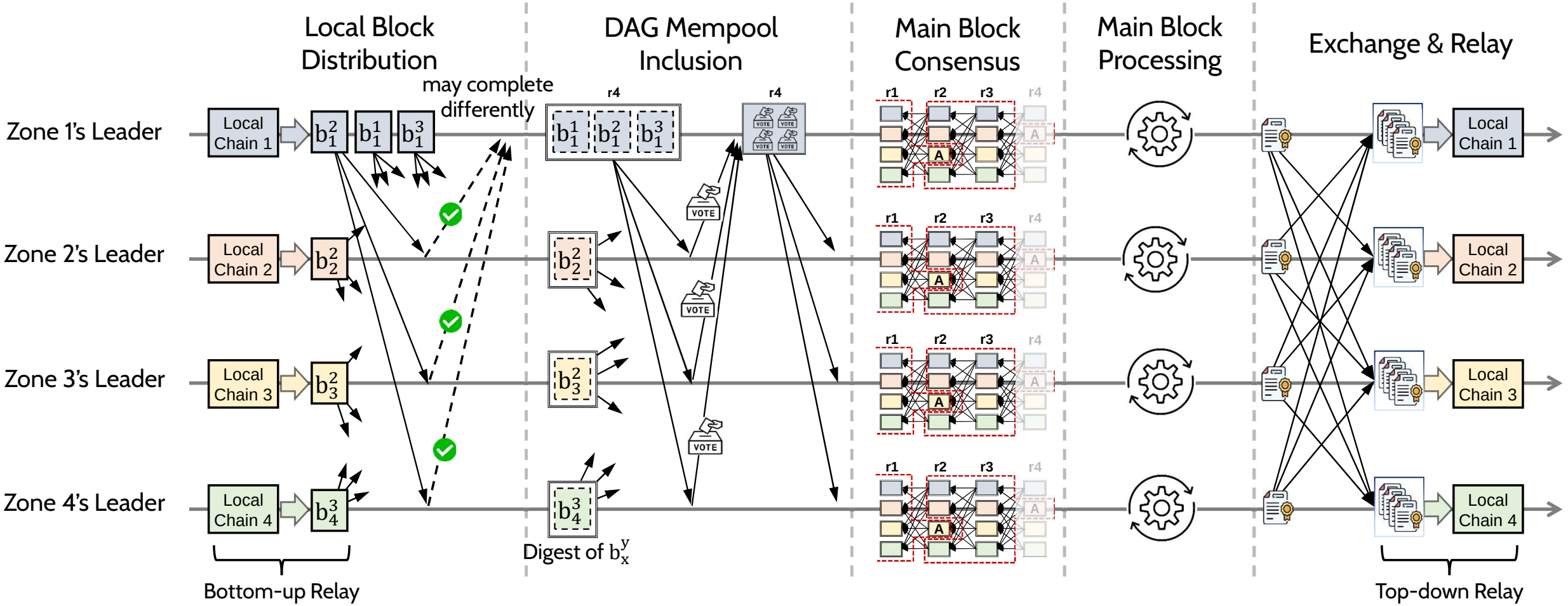
PyloChain: An Overview



PyloChain: An Overview

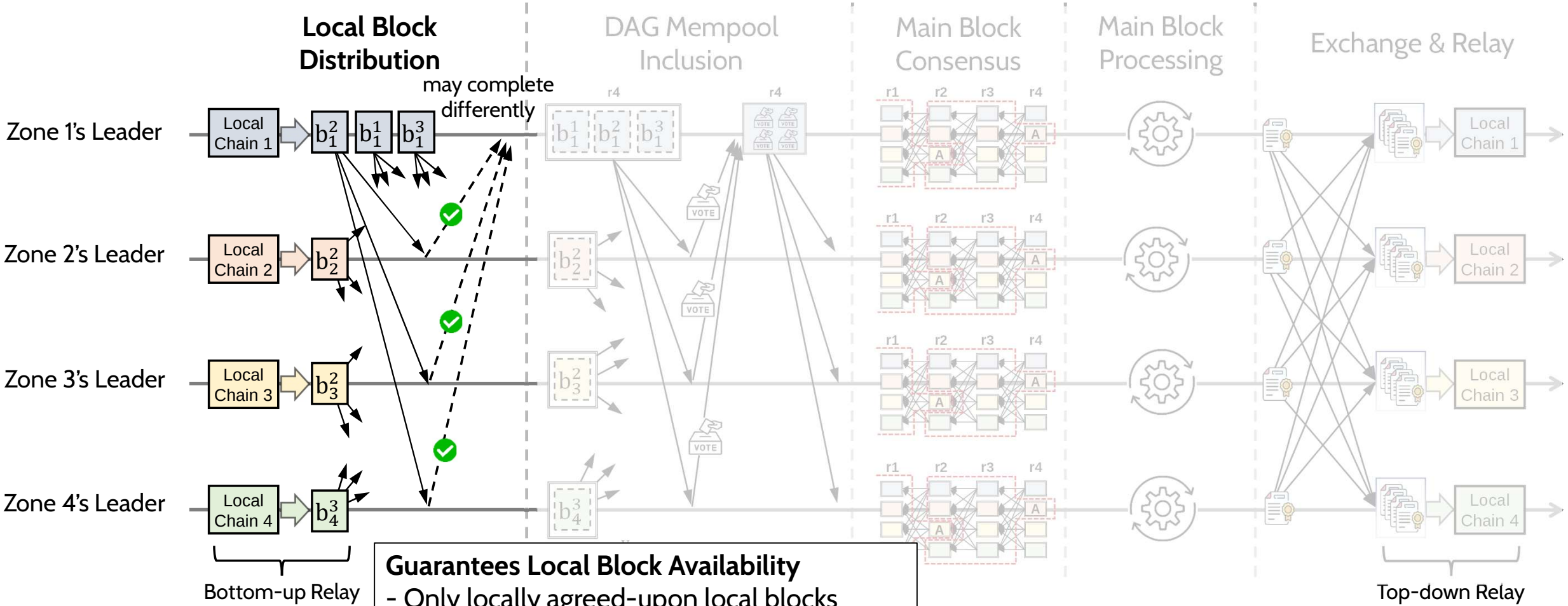


PyloChain: The Main Chain Protocol



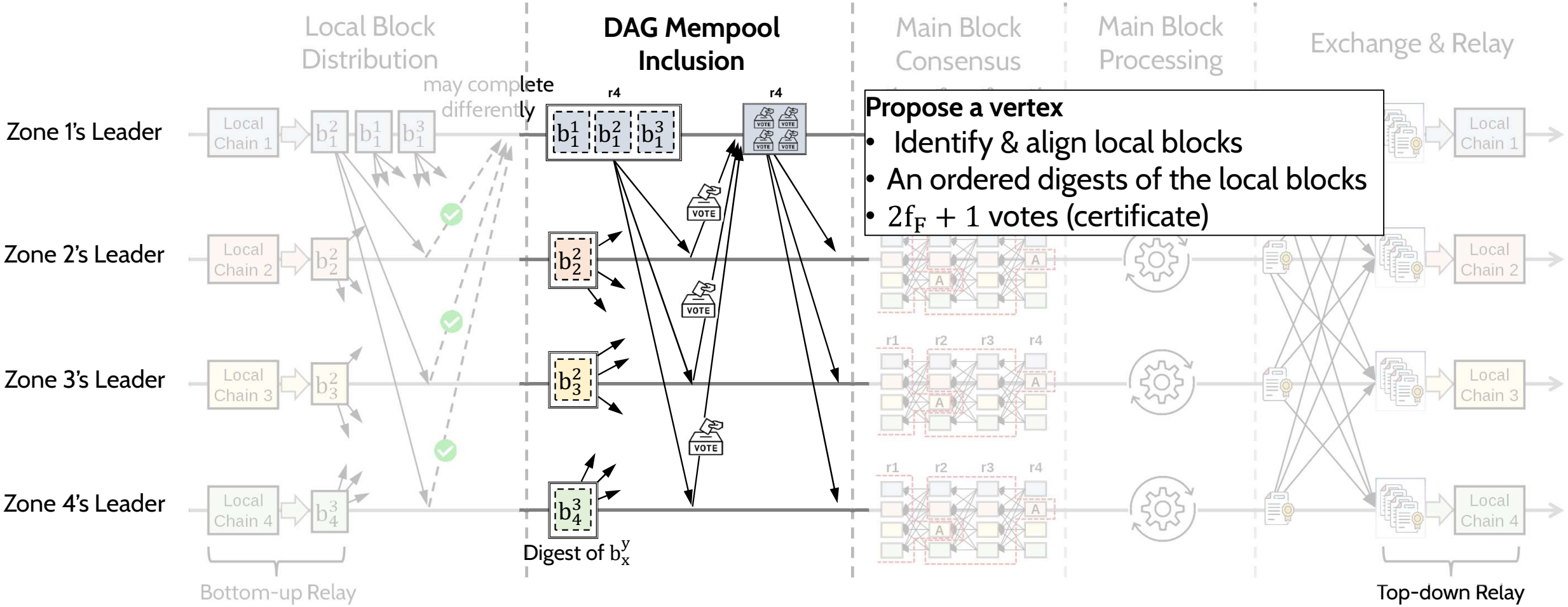
b_x^y : x-th local chain's y-th local block

PyloChain: The Main Chain Protocol

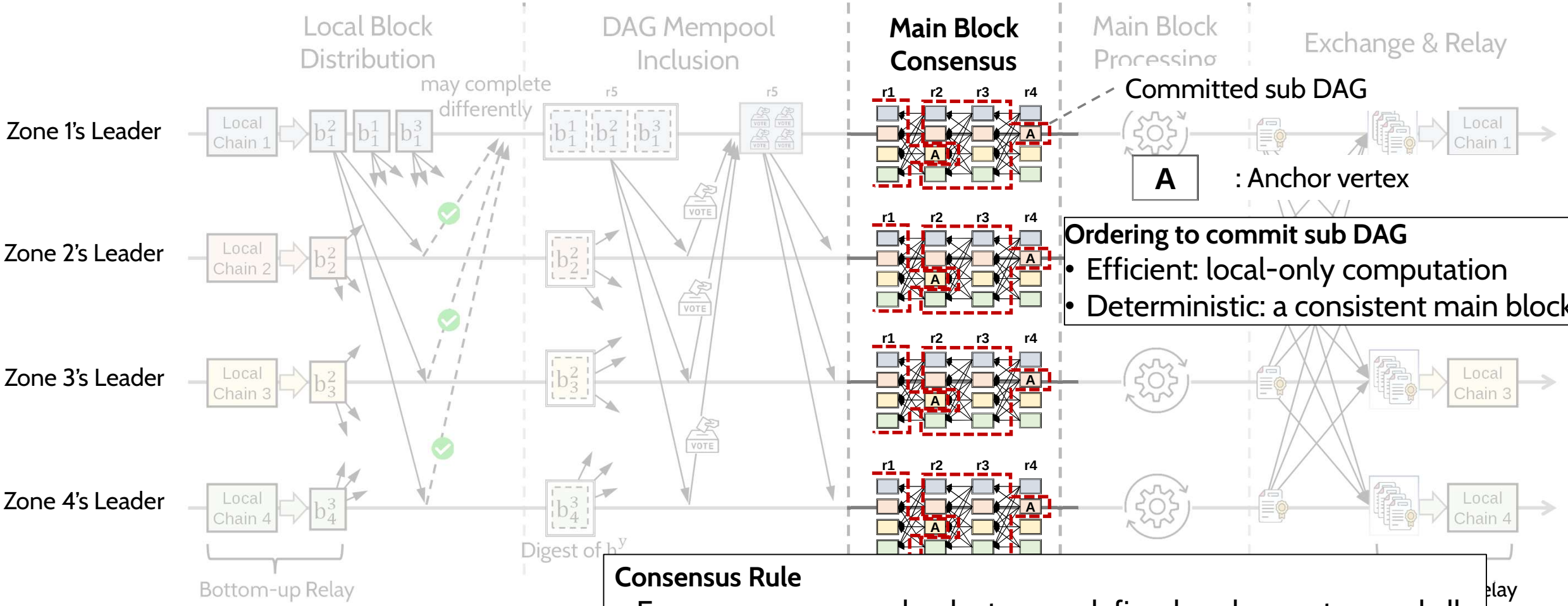


Guarantees Local Block Availability
- Only locally agreed-upon local blocks
- Concurrent distributions
- $2f_F + 1$ acknowledgements

PyloChain: The Main Chain Protocol



PyloChain: The Main Chain Protocol



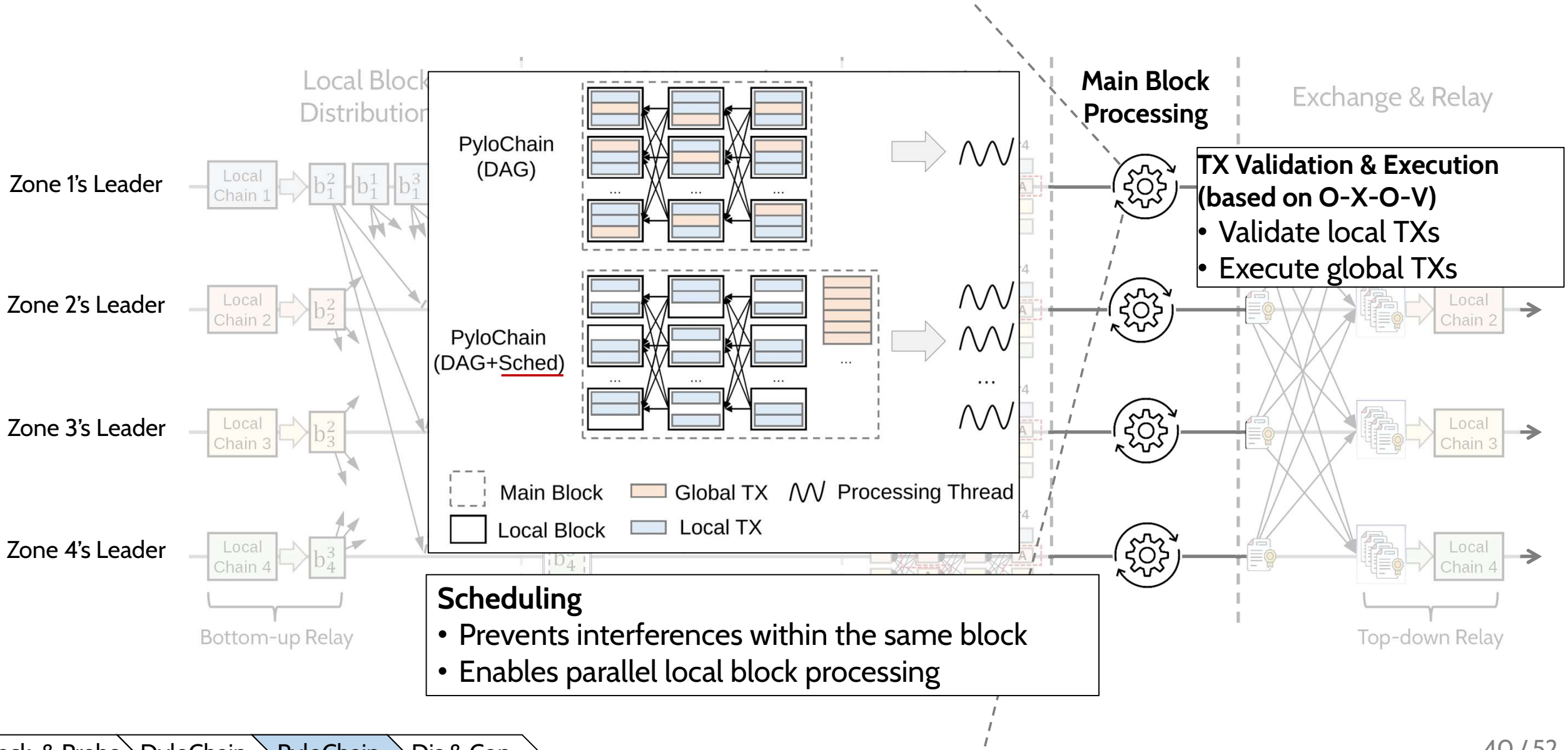
Ordering to commit sub DAG

- Efficient: local-only computation
- Deterministic: a consistent main block

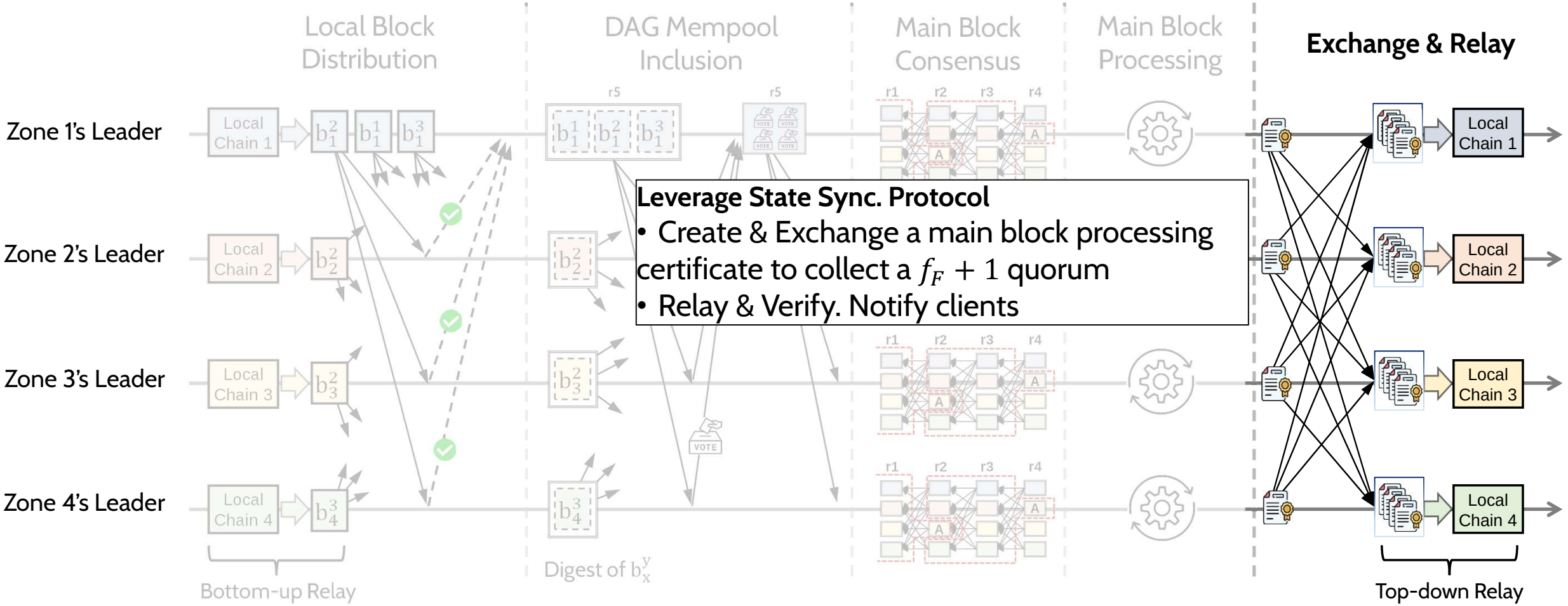
Consensus Rule

- For every even round, select a pre-defined anchor vertex, and all its uncommitted previous vertices (called causal histories)
- Then, topologically sorting them (e.g., depth first search)

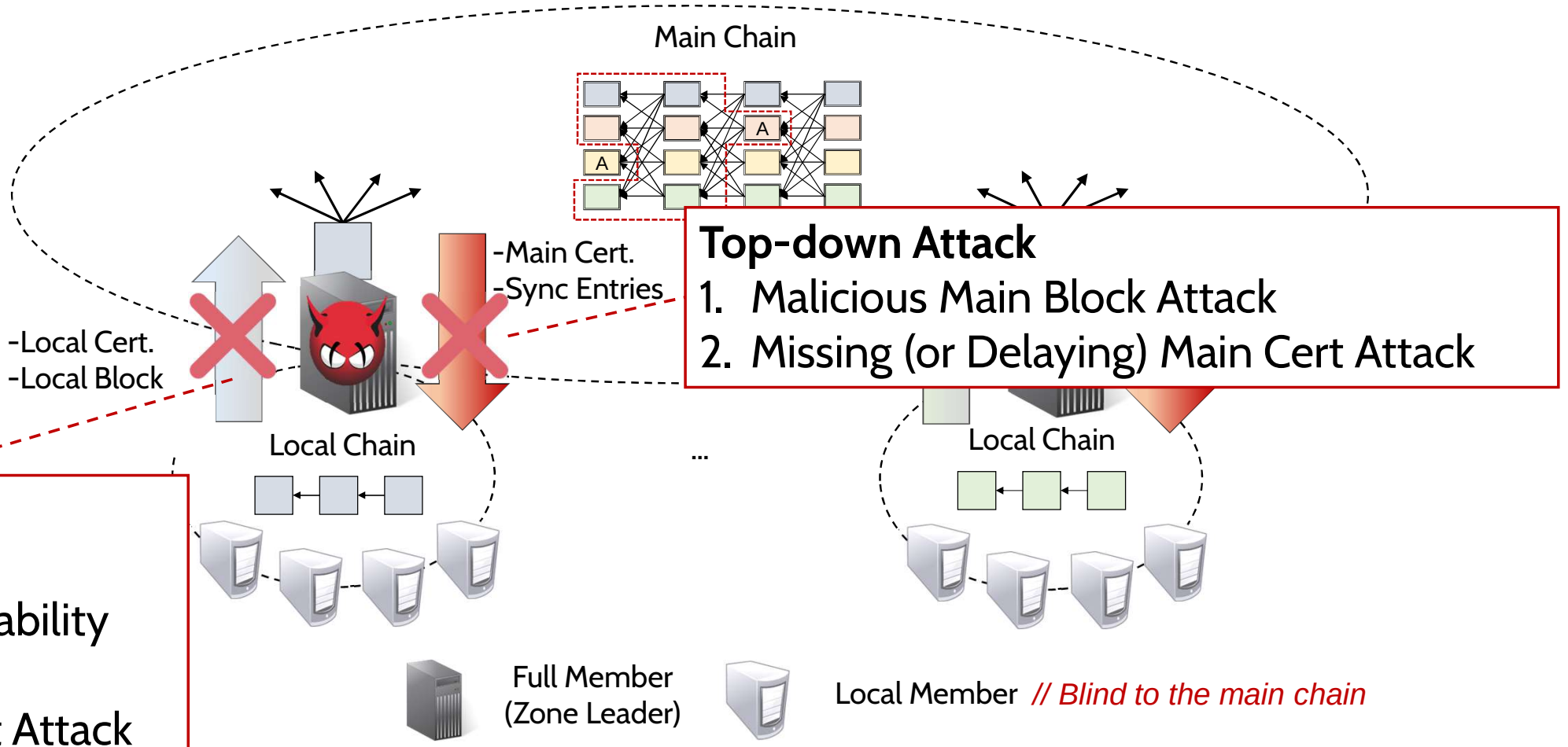
PyloChain: The Main Chain Protocol



PyloChain: The Main Chain Protocol



Auditing Malicious Zone Leader Problem



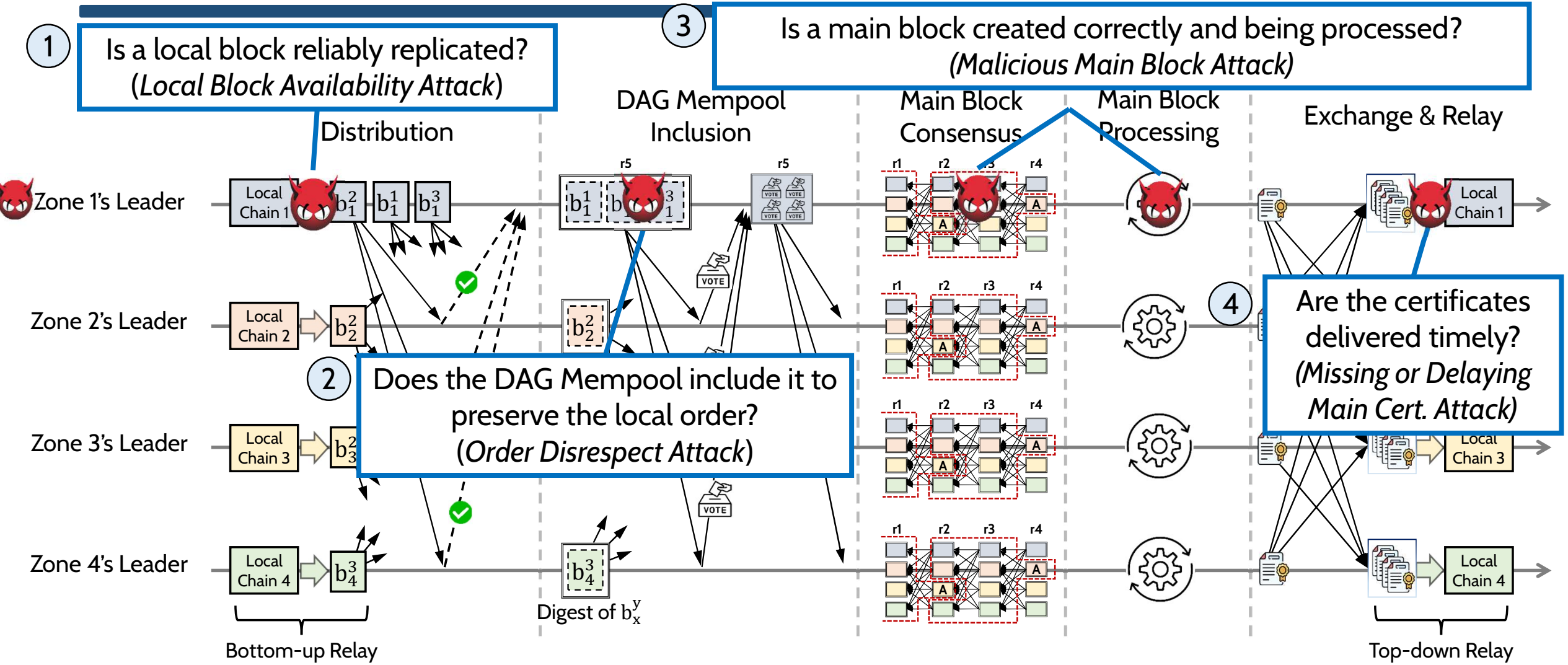
Top-down Attack

1. Malicious Main Block Attack
2. Missing (or Delaying) Main Cert Attack

Bottom-up Attack

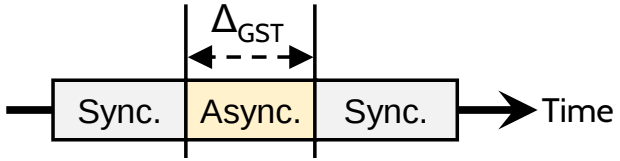
1. Fork Attack
2. Local Block Availability Attack
3. Order Disrespect Attack

Auditing Malicious Zone Leader Problem

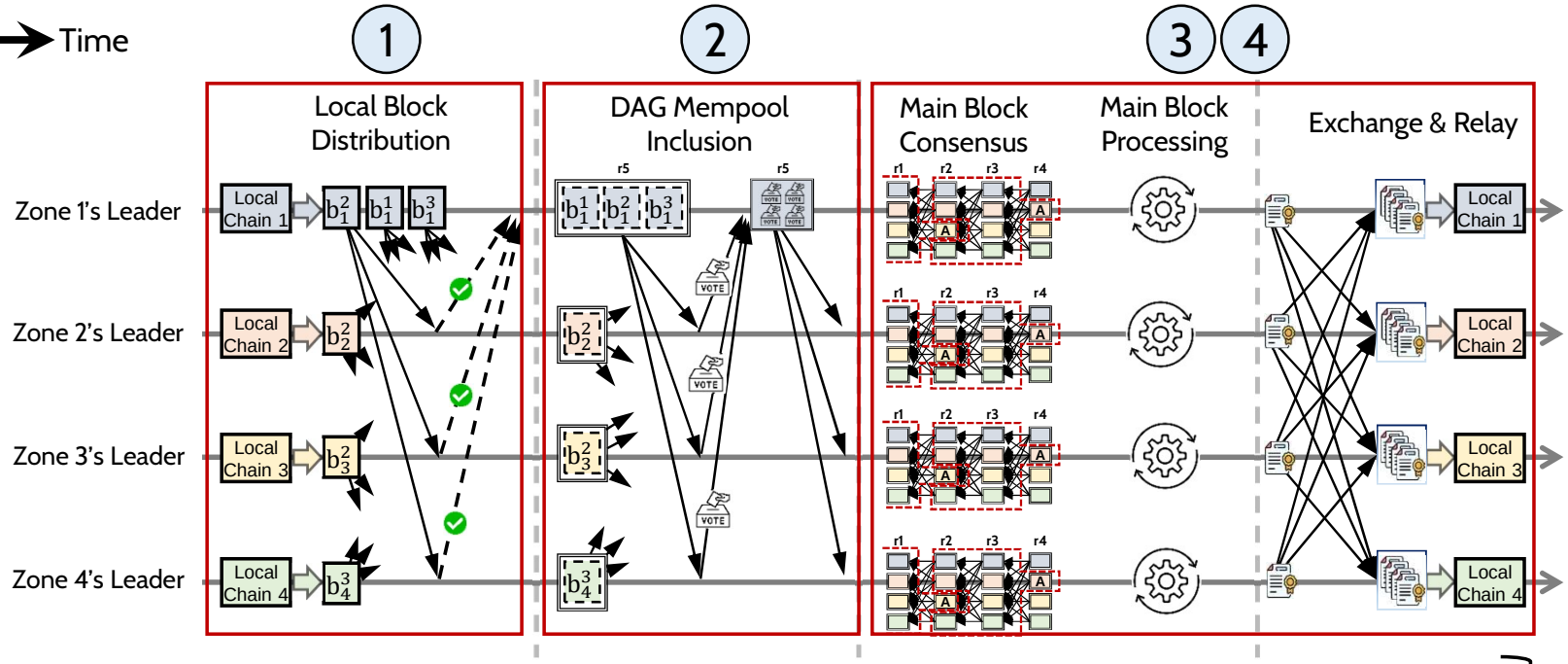


Approach: Externalizing fine-grained semantics of main chain operations

Auditing Malicious Zone Leader Problem



Externalizing each step to local members



Externalized Data	$2f_F + 1$ Acks.	$2f_F + 1$ Votes.	$f_F + 1$ Main Block Processing Certificates
Expected Latency (Sync.)	$3\Delta_{sync}$	$3\Delta_{sync}$	$6\Delta_{sync}$
Expected Latency (Async. + GST)	$3\Delta_{sync} + \Delta_{GST}$	$3\Delta_{sync} + \Delta_{GST}$	$6\Delta_{sync} + \Delta_{GST}$

After detection, recovery (or change) protocol is triggered

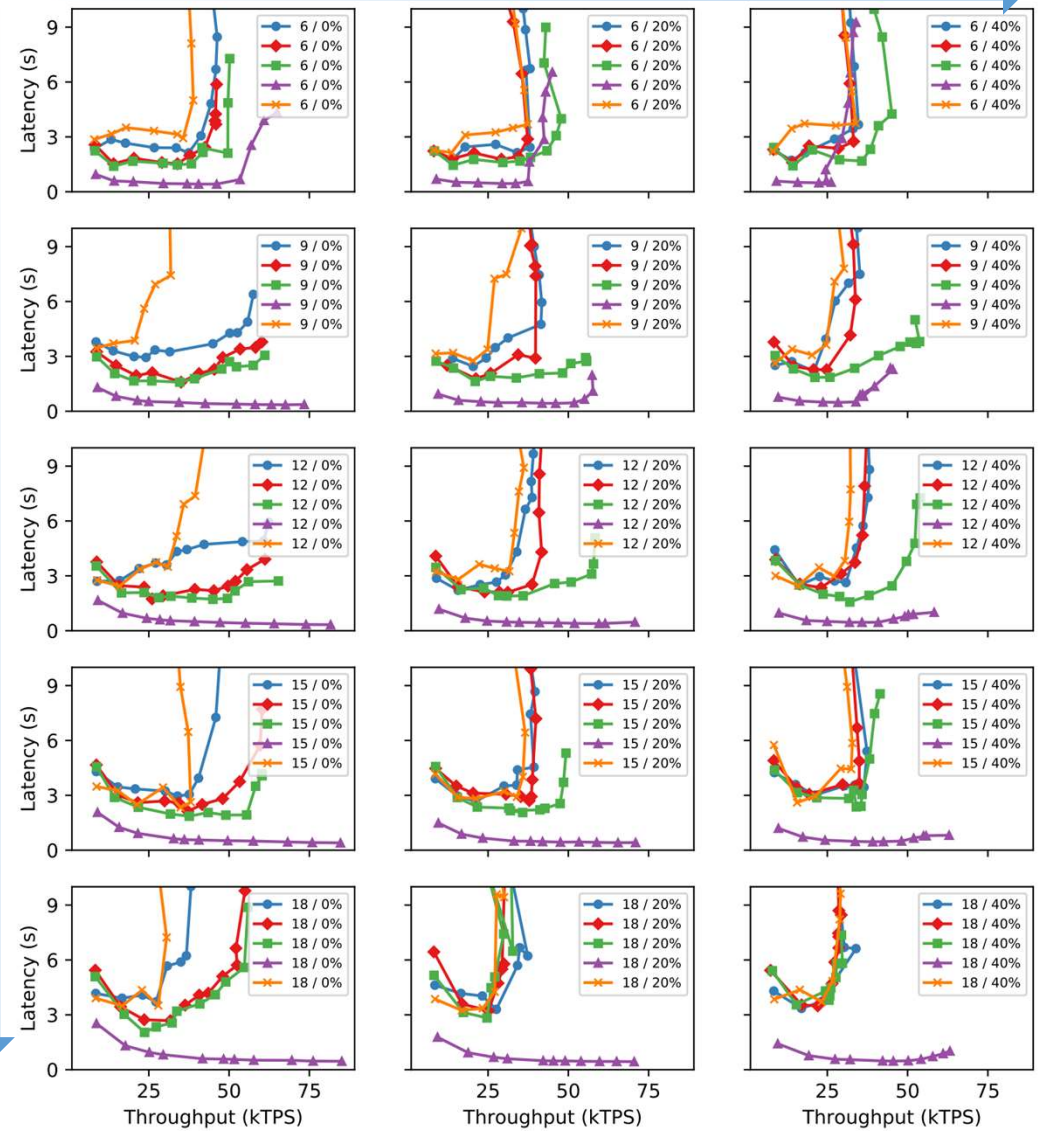
Experimental Setup

- **Building upon DyloChain**
  - Bullshark* -based Main Chain
- **Implementing other sharding schemes**
  - Availability Sharding: Replicates local blocks from DAG mempool to all local members
  - Performance Sharding: No main chain & 2PC for cross-shard TXs
- **Networks: Up to 18 zones with 90 members**
  - 18 full members + 72 local members
- **Workload**
  - SmallBank with 300,000 accounts (equally distributed & uniformly accessed)
  - Local Block Size: 500 TXs, %Global TX = [0%, 20%, 40%]
  - TX send rates (10k ~ 110k) are equally distributed

*github.com/facebookresearch/narwhal

Experimental Result : Overall Performance

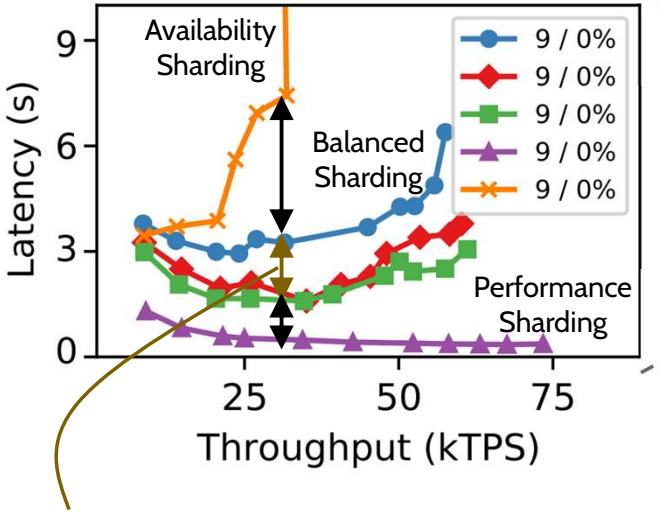
Global TX Overhead

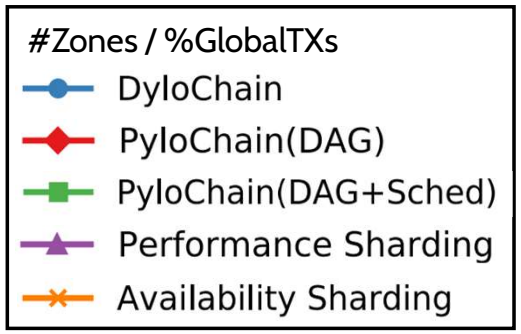


#Zones / %GlobalTXs
- DyloChain
- PyloChain(DAG)
- PyloChain(DAG+Sched)
- Performance Sharding
- Availability Sharding

Sharding Effect
(peak at 12 zones)

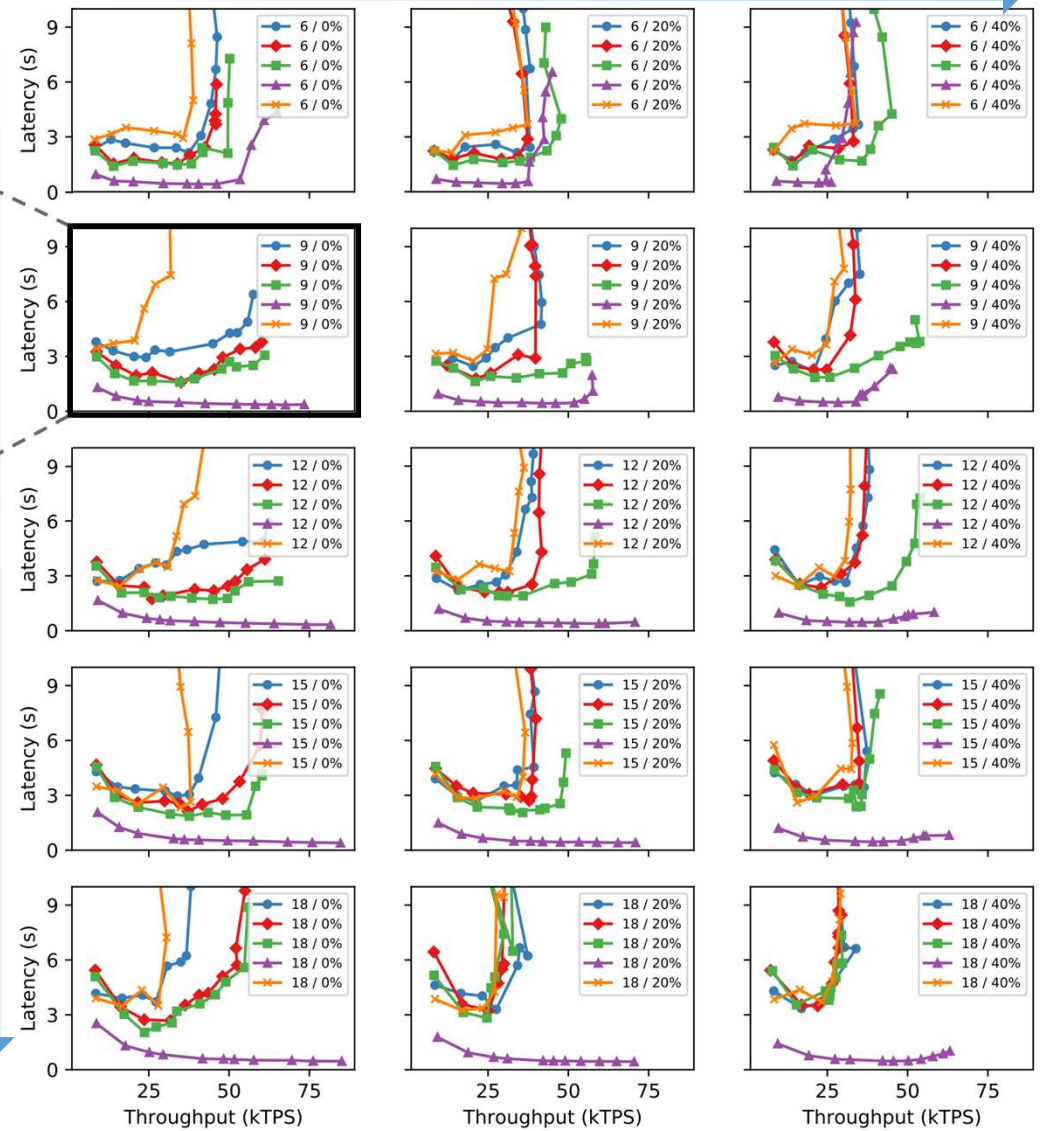
Experimental Result : Overall Performance



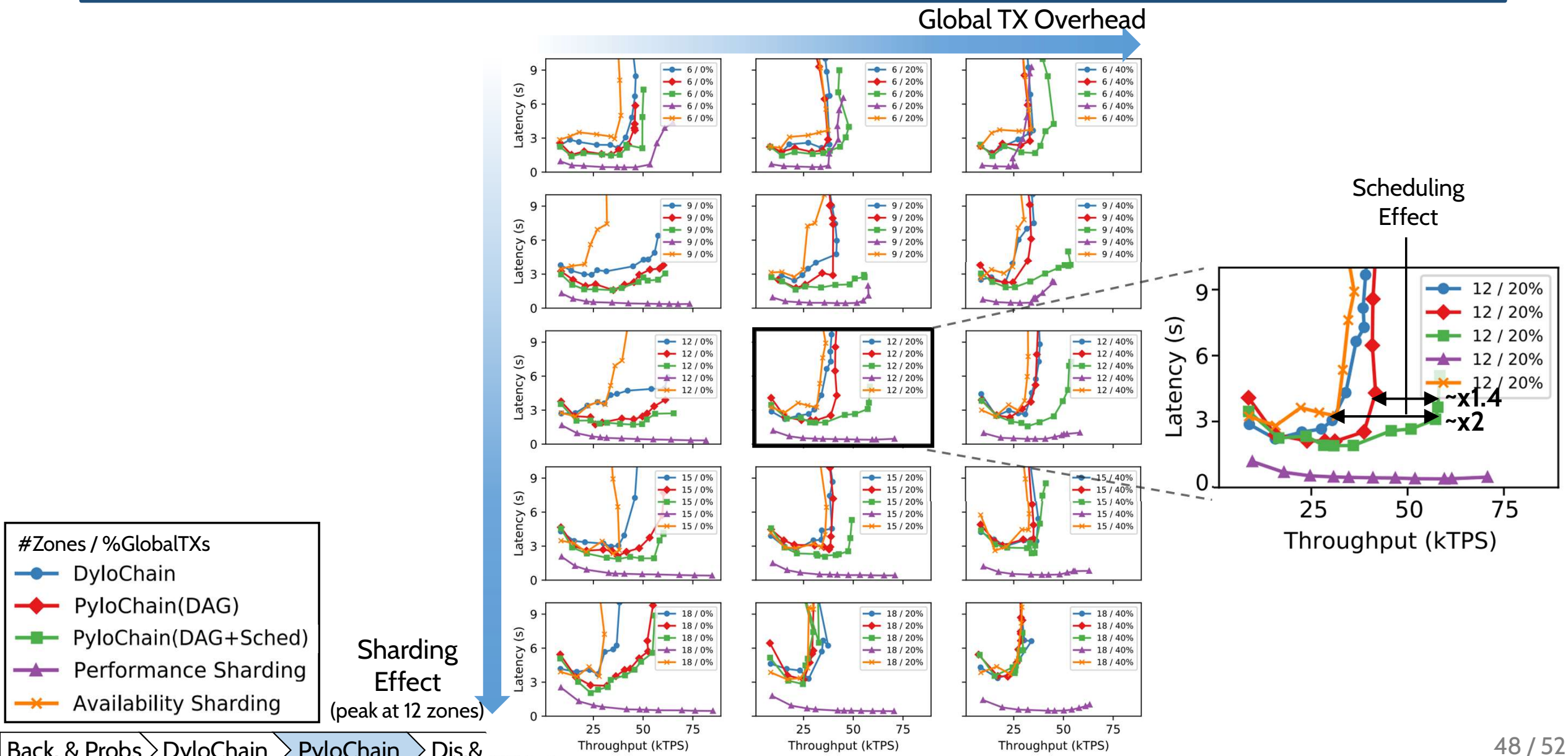


Sharding Effect
(peak at 12 zones)

Global TX Overhead

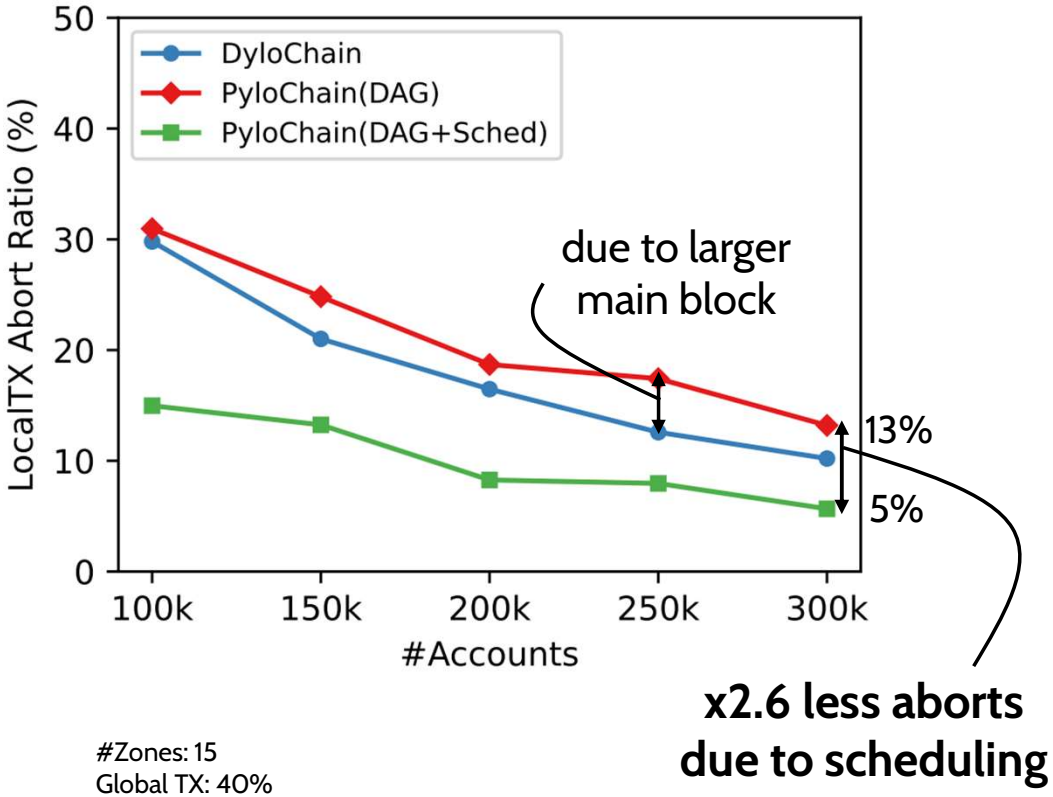


Experimental Result : Overall Performance



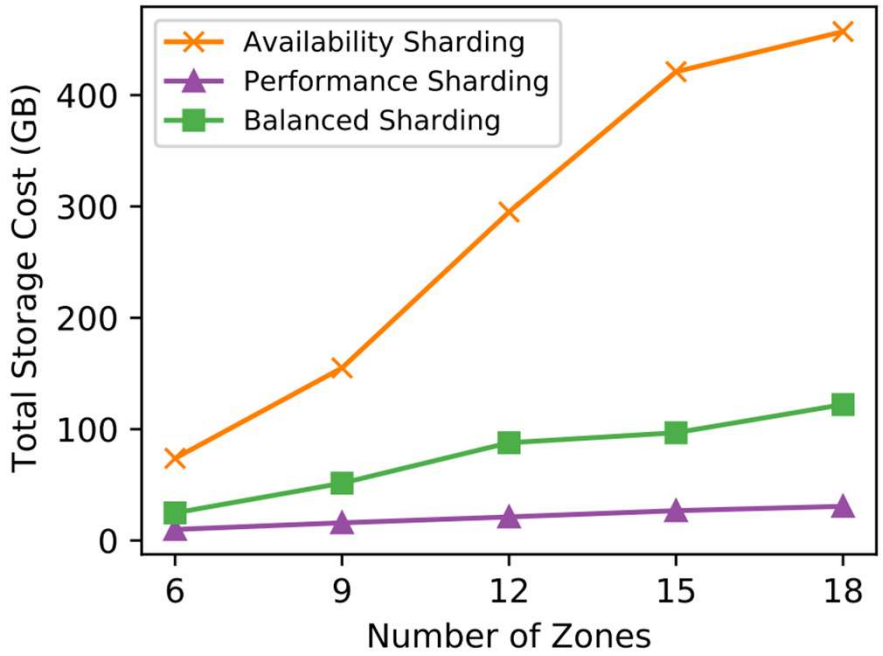
Experimental Result : Analysis

Interference Analysis



#Zones: 15
Global TX: 40%
SendRate: 80000

Storage Analysis



Global TX: 0%
SendRate: 1000 per clients

Related Works

	Sharding Scheme	Hierarchy	Dynamic Locality	Cross-Shard TX	Scalability
HLF[1)]	N/A	X	N/A	N/A	Very Low
Ziziphus[2)]	Performance	O	Enforced	Convert to Local TX	High
Saguaro[3)]	Performance	O	Selective	2PC or Optimistic	High
Meepo[4)]	Availability	X	X	Epoch-based	Low
Ours	Balanced	O	Selective	Main Chain	Balanced

1) Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the Thirteenth EuroSys Conference (Porto, Portugal) (EuroSys '18). Association for Computing Machinery, New York, NY, USA, Article 30, 15 pages. https://doi.org/10.1145/3190508.3190538
2) Mohammad Javad Amiri, Daniel Shu, Sujaya Maiyya, Divyakant Agrawal, and Amr El Abbadi. 2023. Ziziphus: Scalable Data Management Across Byzantine Edge Servers. In 2023 IEEE 39th International Conference on Data Engineering (ICDE). 490–502. https://doi.org/10.1109/ICDE55515.2023.00044
3) Mohammad Javad Amiri, Ziliang Lai, Liana Patel, Boon Thau Loo, Eric Lo, and Wenchao Zhou. 2023. Saguaro: An Edge Computing-Enabled Hierarchical Permissioned Blockchain. In 2023 IEEE 39th International Conference on Data Engineering (ICDE). 259–272. https://doi.org/10.1109/ICDE55515.2023.00027
4) Peilin Zheng, Quanqing Xu, Zibin Zheng, Zhiyuan Zhou, Ying Yan, and Hui Zhang. 2021. Meepo: Sharded Consortium Blockchain. In 2021 IEEE 37th International Conference on Data Engineering (ICDE). 1847–1852. https://doi.org/10.1109/ICDE51399.2021.00165

Discussions & Future Work

- **Byzantine Shard Resilient Cross-Shard Protocol**
  - Adversaries can fully corrupt local chains up to f_F
  - However, they cannot compromise the main chain due to its $3f_F + 1$ (or $2f_F + 1$) model
  - Yet, Application-level state corruption (e.g., fake account values) remains unhandled

- **Parallel Global TX Executions**
  - Executing non-conflicting global TXs in parallel
  - Main chain can easily leverage existing methods (e.g., OptME)

- **Across Main Blocks Scheduling**
  - Further reduce interferences with multiple main blocks
  - But, fairness issues and higher latency

Conclusions

- **Sharding-based Hierarchical Blockchain for Large-Scale TX Processing**

  Challenge #1. Overcoming Limited Availability

  Challenge #2. Efficient Handling of Cross Shard TXs

  Challenge #3. Supporting Dynamic Locality

DyloChain
- Synchronous Main Chain
- Order-Execute-Order-Validate (O-X-O-V)
- State Reshard Protocol

PyloChain
- Partially Synchronous Main Chain
- DAG-based Main Chain for Scalability
- Scheduling Cross Shard TXs for Efficiency