Mir-BFT:
High-Throughput Robust BFT for Decentralized Networks

Marko Vukolić
IBM Research - Zurich

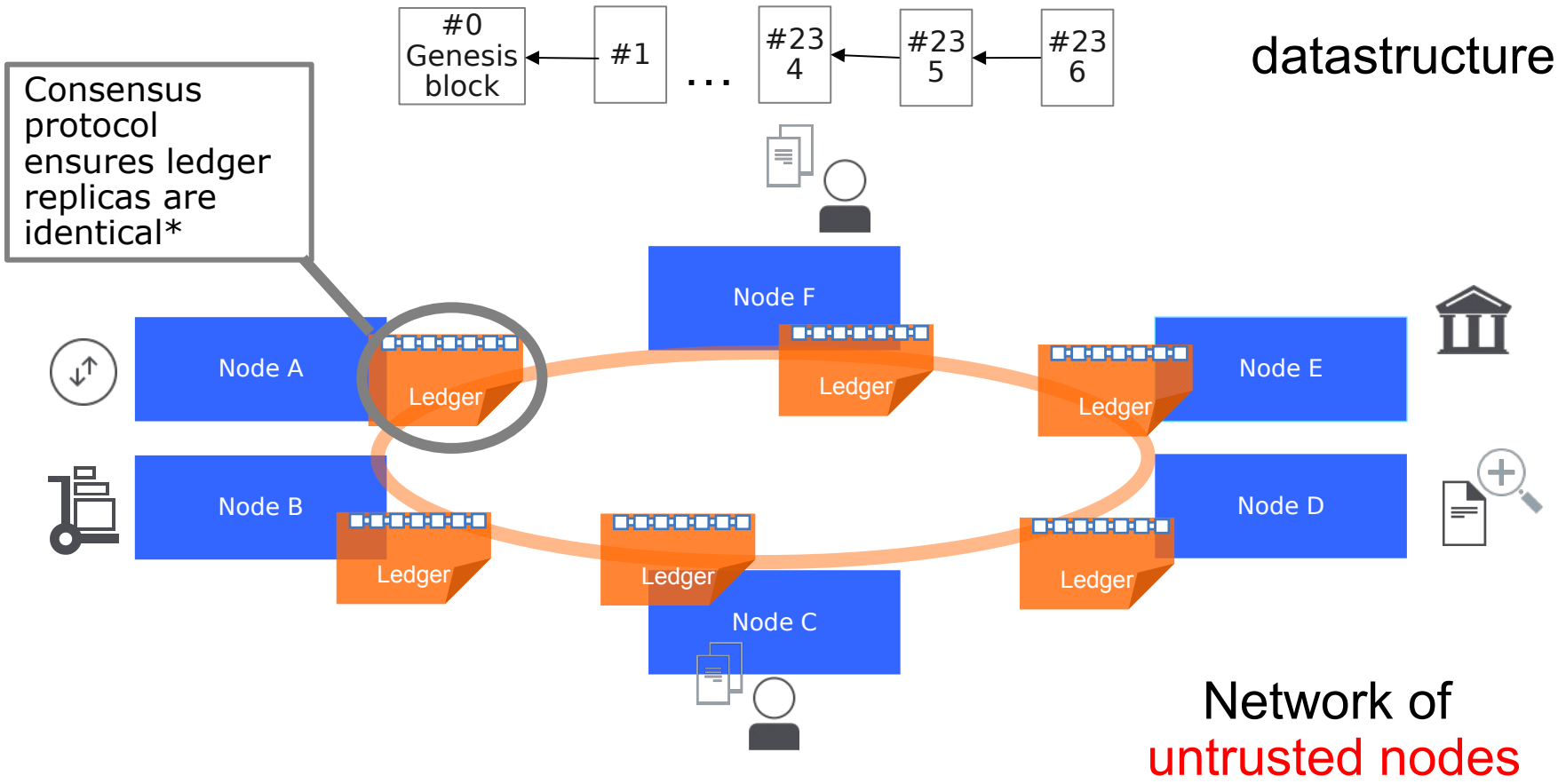
(originally a joint work with
Chrysa Stathakopoulou, Tudor David and Matej Pavlovic)

June 10, 2021

© 2021 IBM Corporation

Consensus establishes total order of transactions in a blockchain

- **Blockchain: A chain (sequence) of <u>blocks</u> of transactions**
  - Each block consists of a list of transactions
  - Blockchain establishes total order of transactions



Consensus for Blockchains

Introduced with Bitcoin's consensus mechanism (Proof-of-Work)

Inherent Problems

- Higher energy consumption than Switzerland, Austria, …
- No-finality: Forks when two nodes solve the PoW puzzle at roughly the same time
- Low throughput (7 tps)
- High latency (10 minutes – 1 hour)

Proof-of-Stake Permissionless and Permissioned Blockchains

- Rely on Byzantine Fault Tolerant (BFT) consensus to circumvent PoW issues
  - PBFT, Tendermint, BFT-SMaRt, Algorand, HotStuff, HoneyBadger, etc.

This work

Mir-BFT is a robust BFT total order (consensus) protocol

with the highest throughput on WANs

with up to 100 (and probably more) nodes

Suitable for Permissioned (e.g., Hyperledger Fabric)

and Proof-of-Stake Permissionless Blockchains

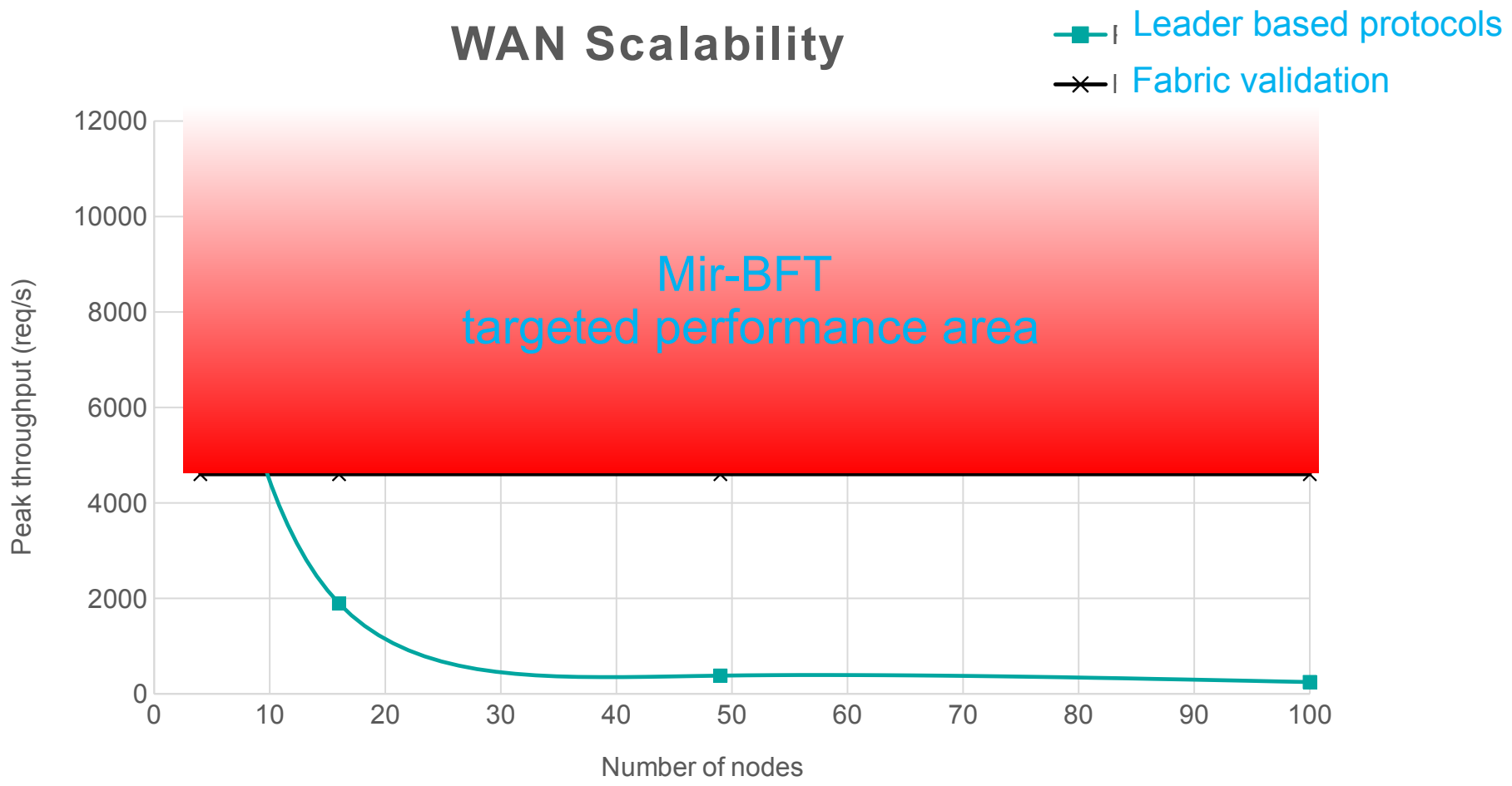
BFT Problem Statement + Model (this talk)

- **N nodes, F of them may be Byzantine faulty (deviate from protocol, malicious)**
  - N ≥ 3F+1
  - Any number of clients can be Byzantine

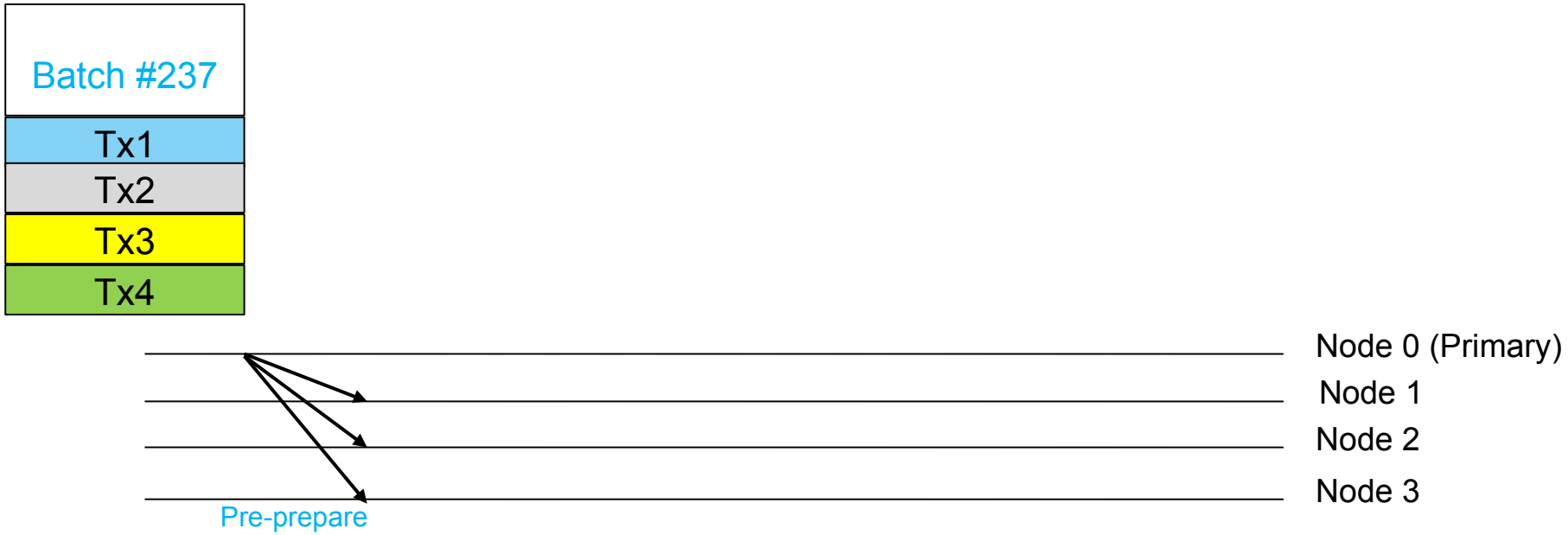
- **Asynchronous* (eventually synchronous) network**
  - messages may be dropped, reordered, delayed
  - After some time GST (unknown to nodes) the network is assumed to be synchronous

- **Problem**: **Total Order broadcast**, informally:
  - **Agreement:** all correct nodes deliver the same transactions in the same order
  - **Validity:** they only deliver transactions proposed by some client
  - **No-Duplication:** Transactions are not duplicated
  - **Liveness:** every transaction proposed by a correct client is eventually delivered by all correct nodes

Motivation

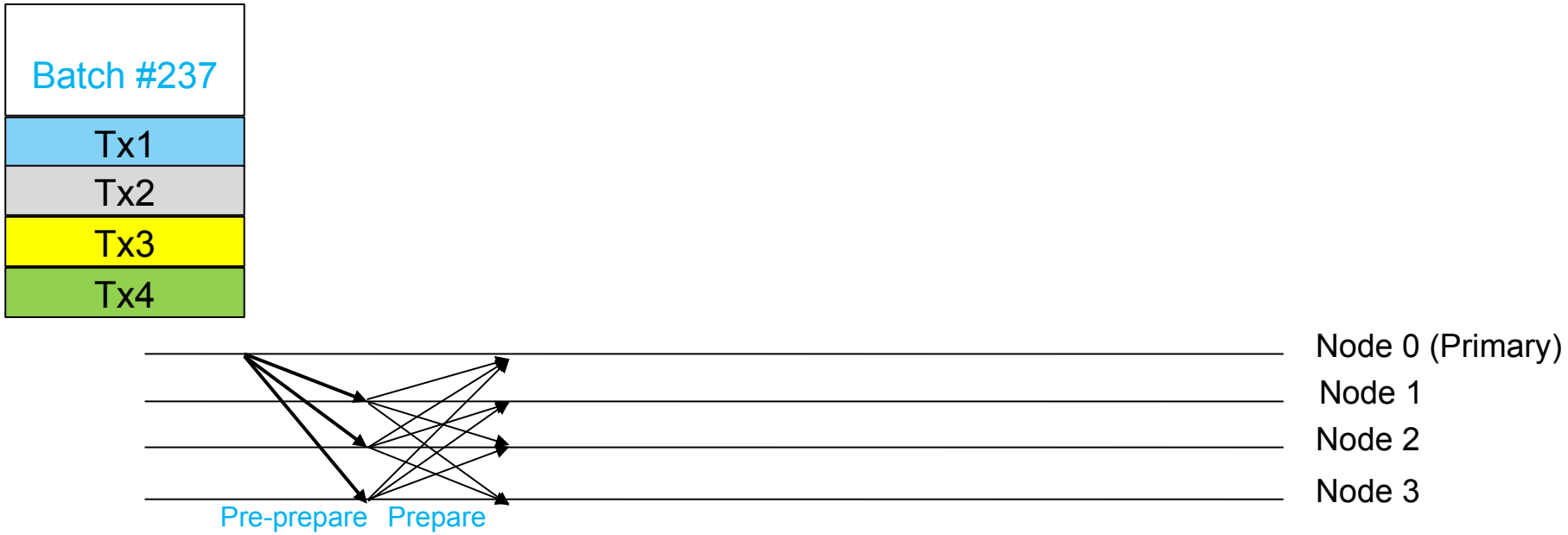


Understanding BFT bottlenecks: Leader-based protocols



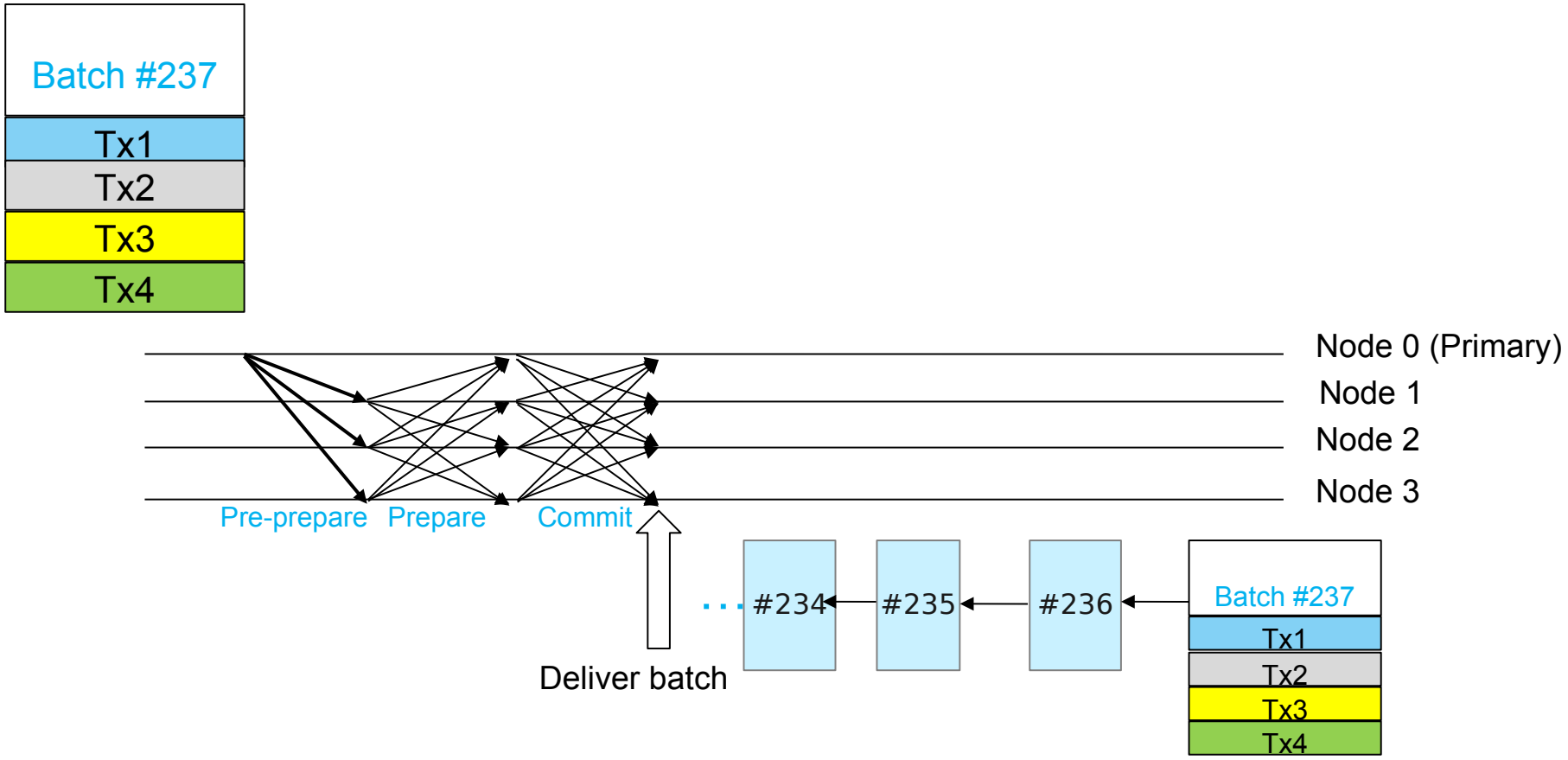
PBFT [Castro/Liskov99], Aardvark [Clement et al. 2009]

Understanding BFT bottlenecks: Leader-based protocols



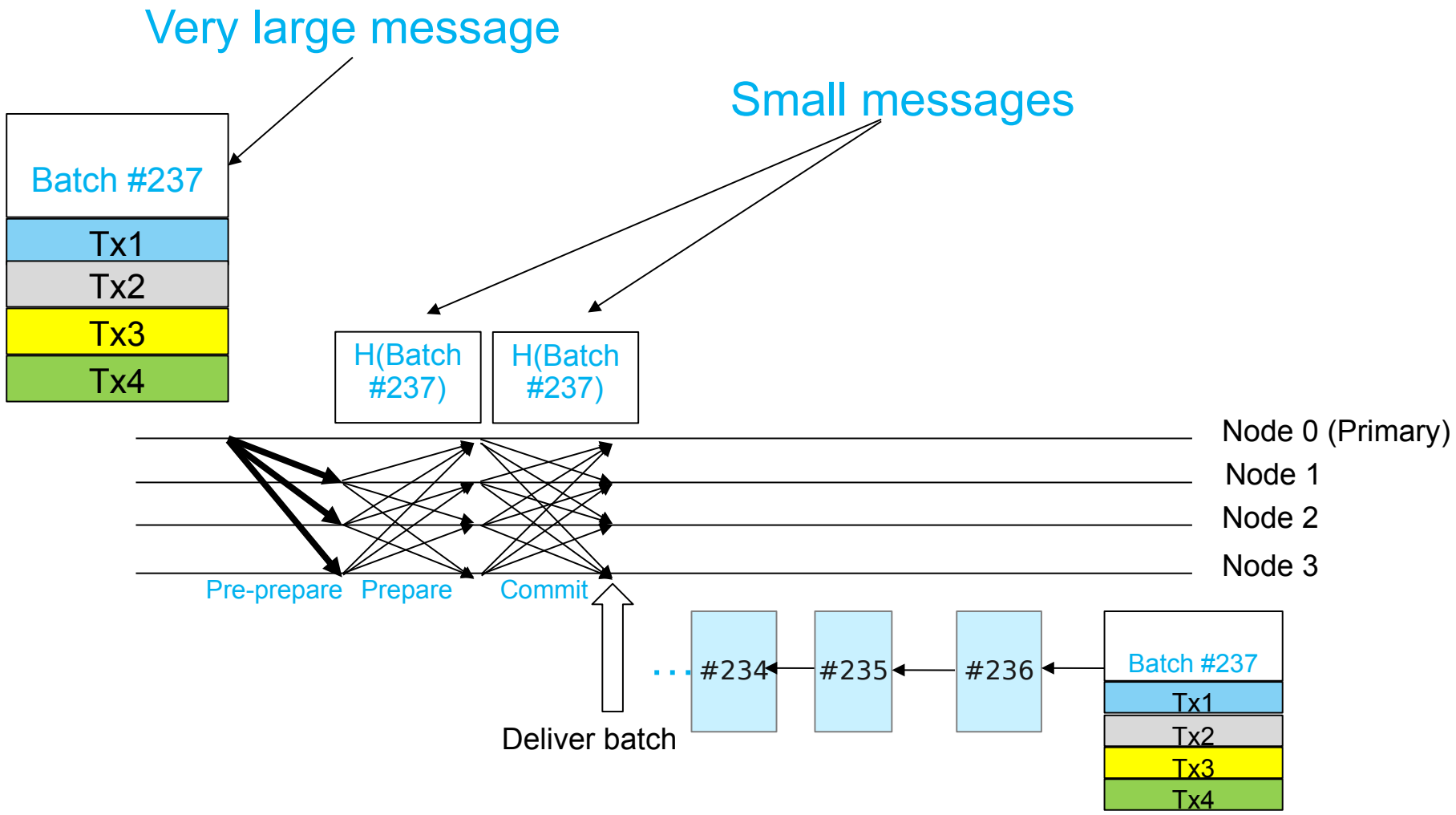
PBFT [Castro/Liskov99], Aardvark [Clement et al. 2009]

Understanding BFT bottlenecks: Leader-based protocols



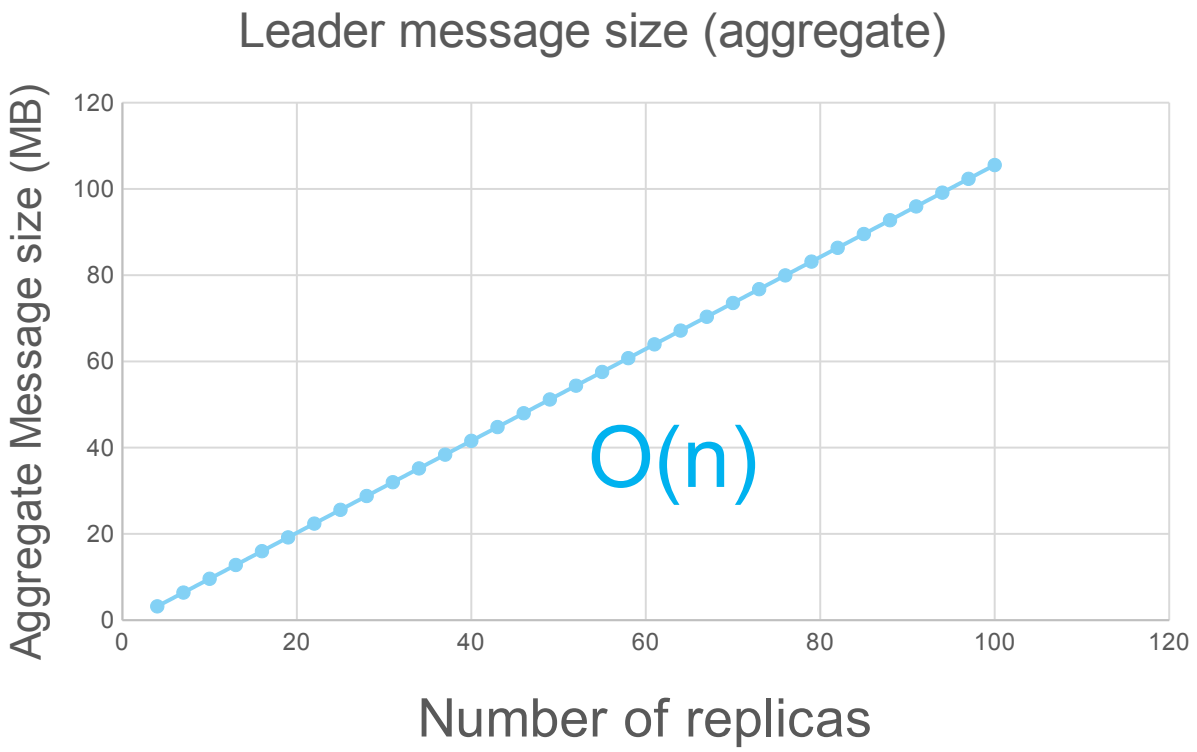
PBFT [Castro/Liskov99], Aardvark [Clement et al. 2009]

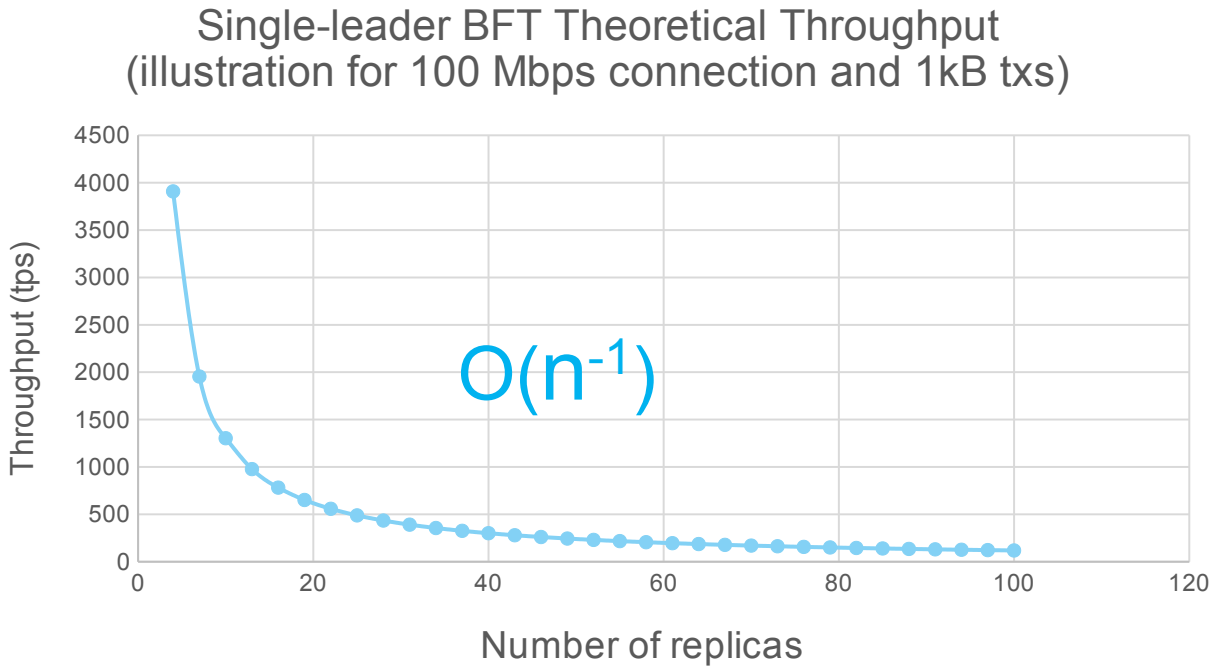
Understanding BFT bottlenecks: Leader-based protocols



PBFT [Castro/Liskov99], Aardvark [Clement et al. 2009]

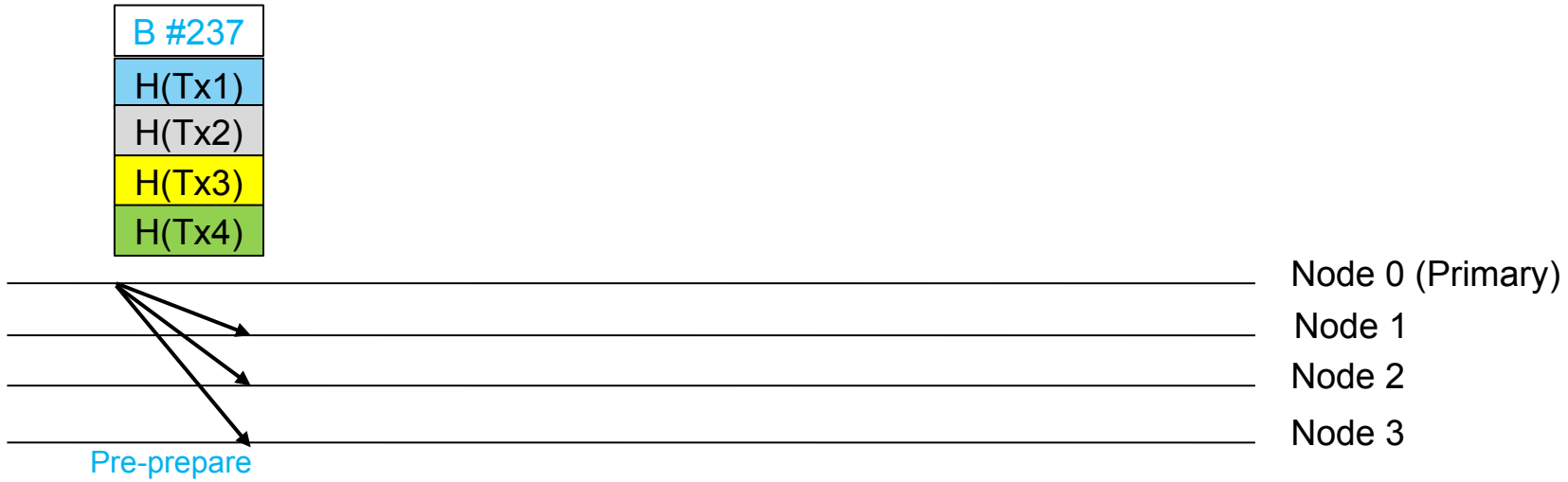
Understanding BFT bottlenecks: leader-based protocols





Robust BFT

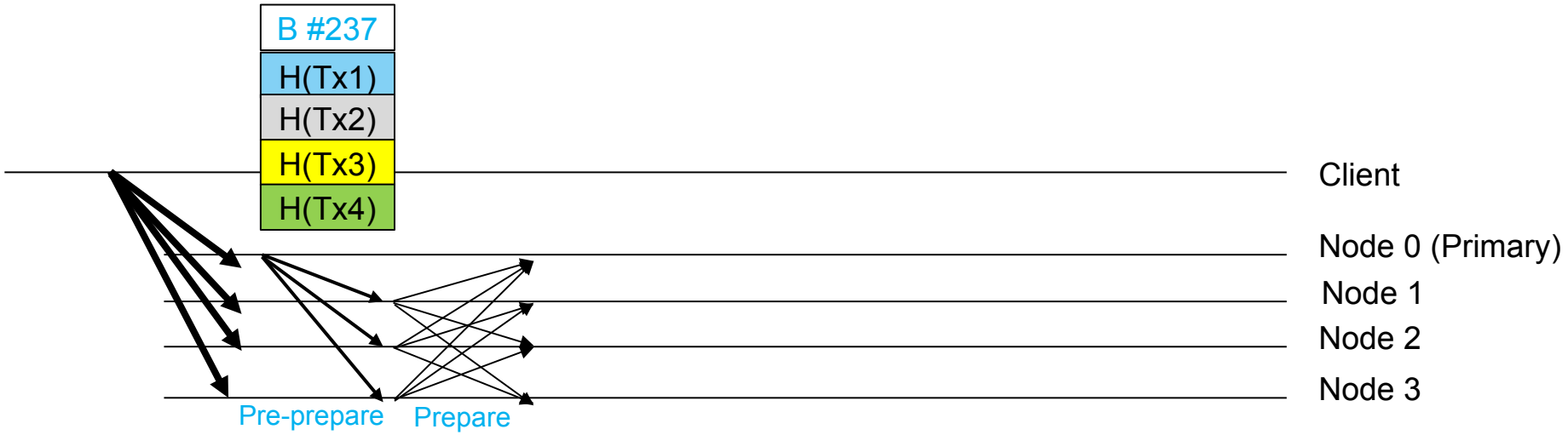
- **Requries sustainable performance of a BFT protocol despite active attacks**
- **Concept Introduced by Aardvark protocol (Clement et al. NSDI 2009)**
  - **showing pitfalls of fragile optimizations**
  - **e.g., client request dissemination (or MAC based client authentication)**



Fragile optimizations: PBFT [Castro/Liskov2002], HotStuff [Yin et al. 2009]

Robust BFT

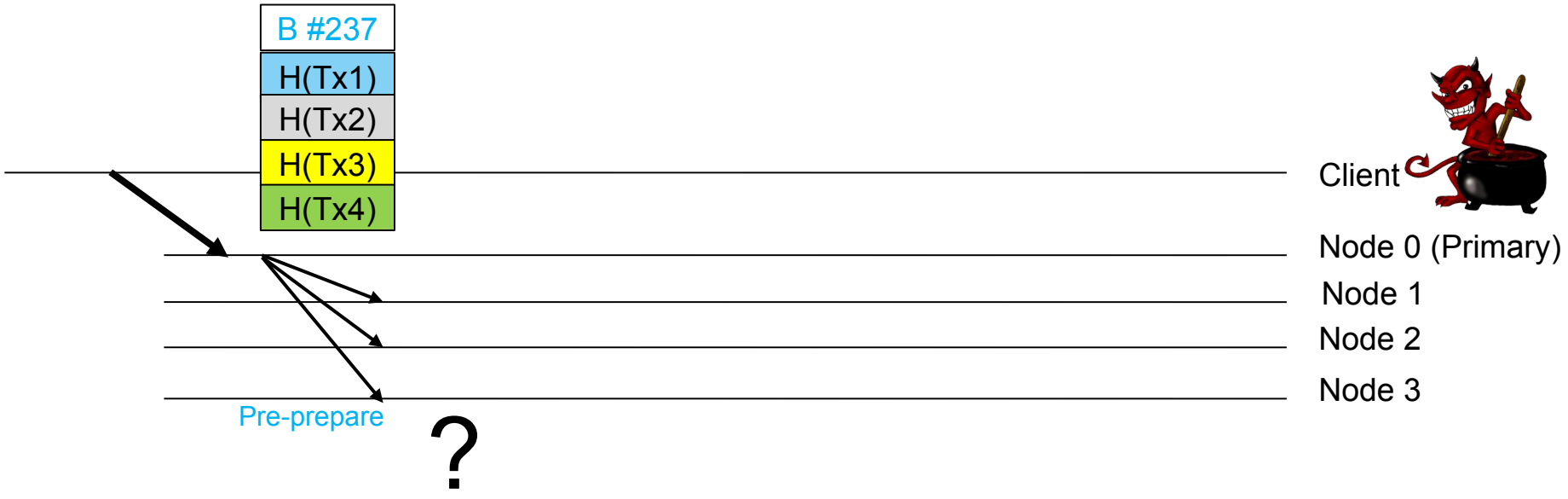
- **Requries sustainable performance of a BFT protocol despite active attacks**
- **Concept Introduced by Aardvark protocol (Clement et al. NSDI 2009)**
  - **showing pitfalls of fragile optimizations**
  - **e.g., client request dissemination (or MAC based client authentication)**



Fragile optimizations: PBFT [Castro/Liskov2002], HotStuff [2019]

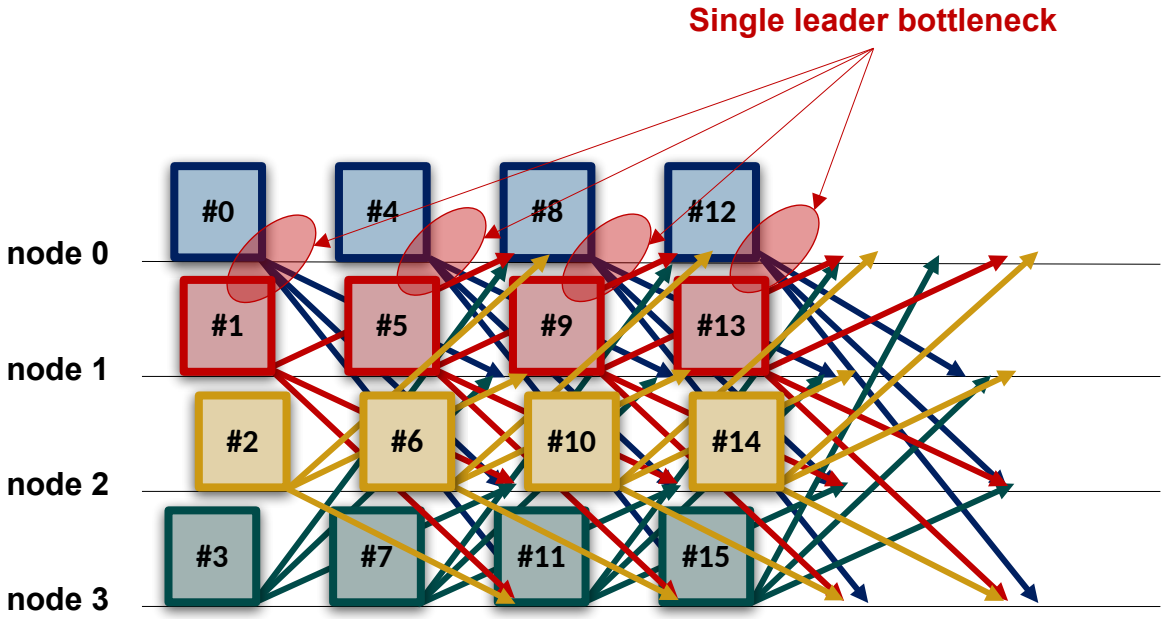
Robust BFT

- **Requries sustainable performance of a BFT protocol despite active attacks**
- **Concept Introduced by Aardvark protocol (Clement et al. NSDI 2009)**
  - **showing pitfalls of fragile optimizations**
  - **e.g., client request dissemination (or MAC based client authentication)**



Fragile optimizations: PBFT [Castro/Liskov2002], HotStuff [2019]

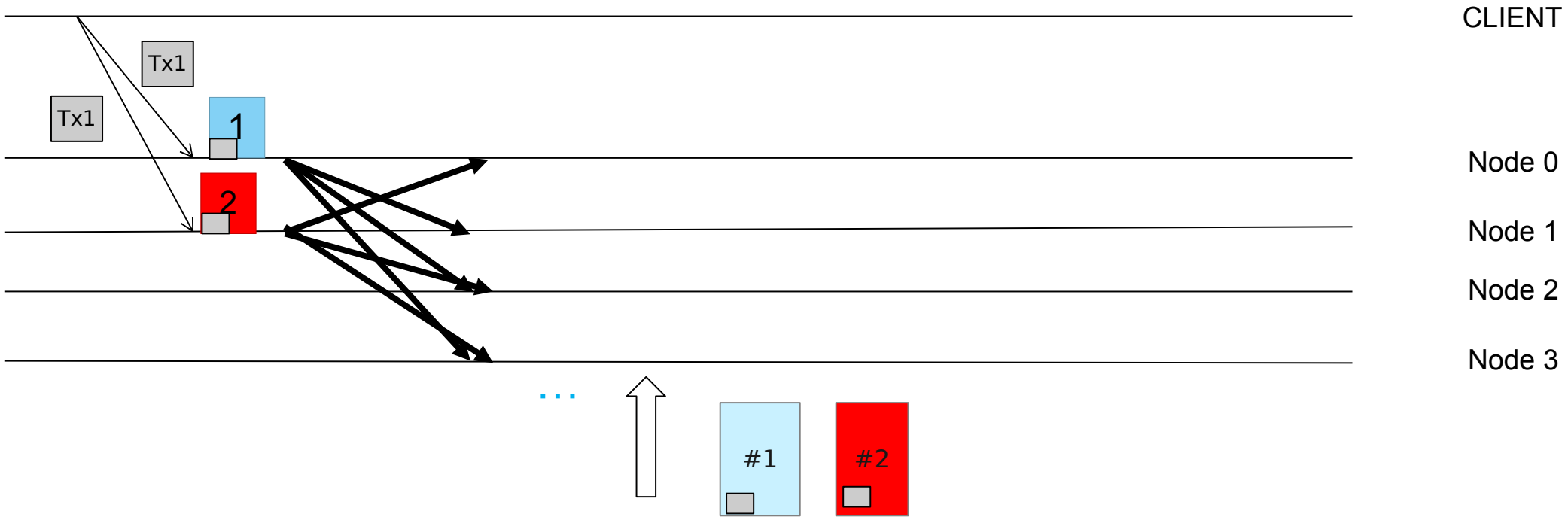
Mir: Robust Scaling of Classical BFT



PBFT: Single leader

Mir: Parallel leaders

Main Challenge with Multiple Leaders: Request duplication



Mir BFT Total Order Broadcast protocol: The main principles

- **Multiple leaders**
  - Have multiple leaders propose requests in parallel (vs PBFT single leader)
  - Sharding block/batch sequence numbers across leaders, multiplexing several single leader instances
- **Prevents request duplication**
  - Prevent duplicates using rotating assignment of partitioned hashspace
- **Incrementally built on PBFT/Aardvark**
  - Mir is a strict generalization of PBFT/Aardvark: **Critical for easier reasoning about correctness**
  - Changes only to PBFT/Aardvark leader(s) election part
  - Asynchronous* & optimally resilient: n=3f+1 nodes in total up to f can be Byzantine
- **Implementation & Optimizations**
  - Parallelized networking and implementation
    - Maintain multiple gRPC connections between each pair of nodes (OSNs)
  - Signature verification sharding (SVS) optimization
    - make sure at least one correct node verifies client sig, as opposed to all nodes)

Preventing Transaction Duplication

Mir Transaction Sharding principles

- **Partition transactions into "buckets"** using a cryptographic hash function H
  - Each transaction is mapped to a unique bucket

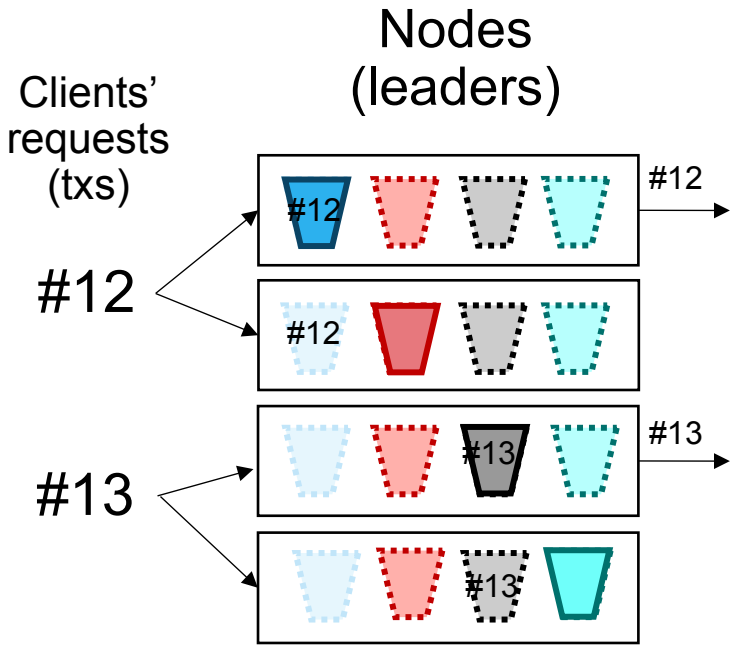
- Each leader has a different **active bucket**
  - Leader proposes only transactions from an active bucket

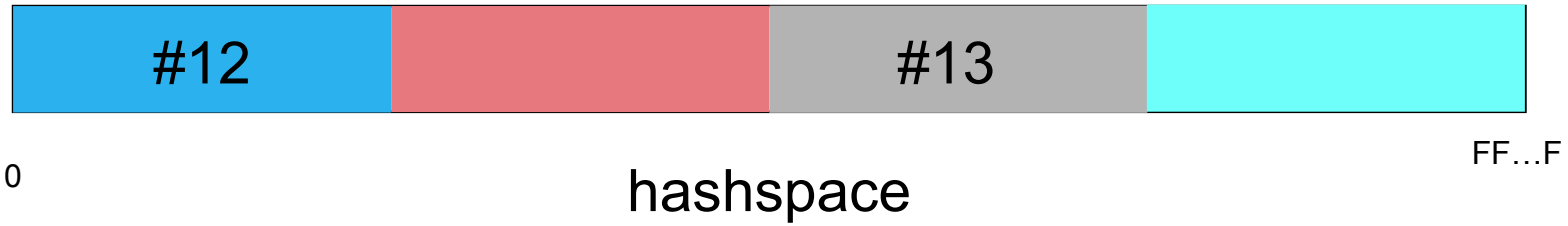
- Periodic active **bucket re-assignment** (**"rotation"**)

Preventing Transaction Duplication: Bucket assignement

Mir Transaction Sharding principles

- **Partition transactions into "buckets"** using a cryptographic hash function H
  - Each transaction is mapped to a unique bucket
- Each leader has a different **active bucket**
  - Leader proposes only transactions from an active bucket
- Periodic active **bucket re-assignment** (**"rotation"**)

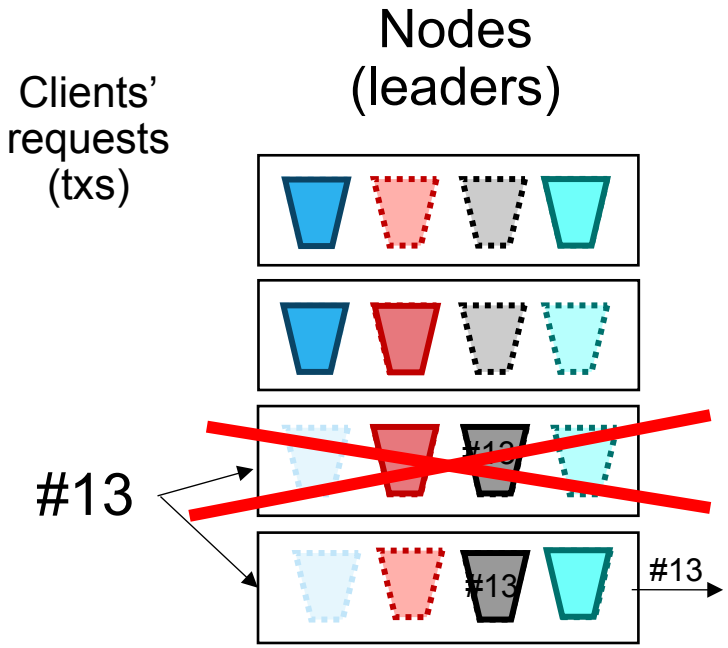


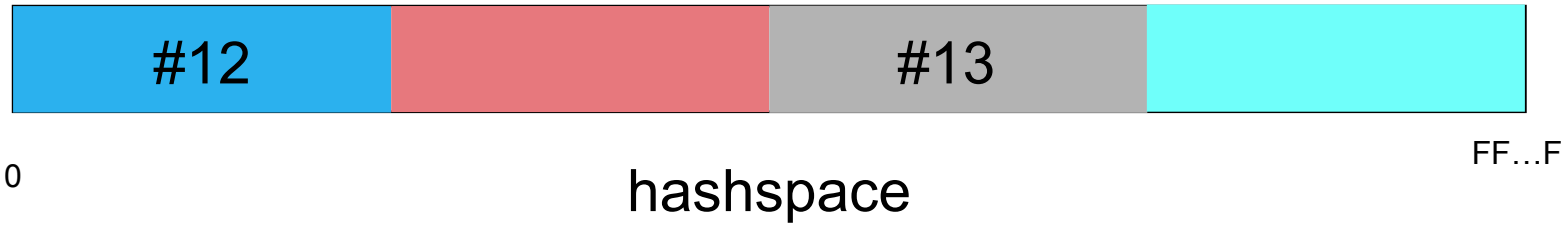


Preventing Transaction Duplication: Bucket Rotation

Mir Transaction Sharding principles

- **Partition transactions into "buckets"** using cryptographic hash function H
  - Each transaction is mapped to a unique bucket
- Each leader has a different **active bucket**
  - Leader proposes only transactions from an active bucket
- Periodic active **bucket re-assignment** (**"rotation"**)

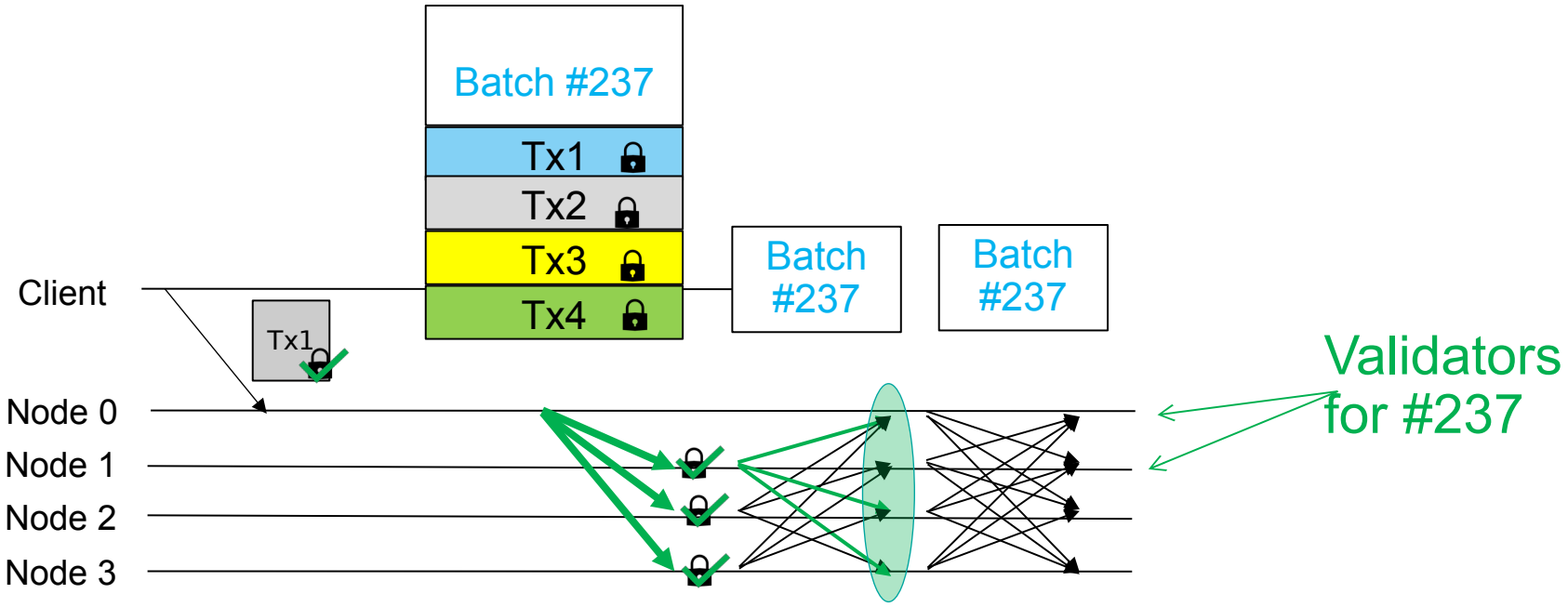




Growing and Shrinking Leader Set (see paper for details)

- **Primary of an epoch announces (reliably broadcasts) the epoch leader set**
  - Subject to constraints
- **Stable epoch: Number of leaders equal to number of nodes**
  - Unbounded epoch duration, moving to **recovery epoch** only in case of faults/partitions
  - Periodic bucket rotation (to protect against censorship attacks)
- **Recovery epoch: Number of leaders is smaller than n**
  - Recovery epoch is limited duration (measured in number of blocks)
- **Gracious epoch change**
  - If “Things are good” (a **complete** recovery epoch e)
  - Then number of leaders in epoch e+1 **does not reduce**
  - **Leader set grows** if the new epoch primary perceives more nodes as alive (until we reach a stable epoch)
- **Ungracious epoch change**
  - Only in case of faults/partitions
  - **The size of the leader set reduces**

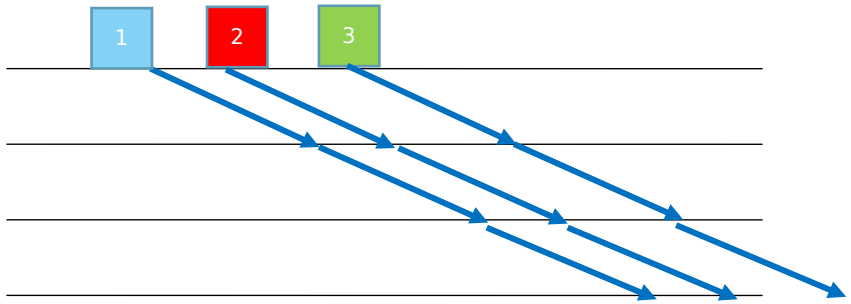
Signature Verification Sharding (SVS) optimization



correct validator for each transaction

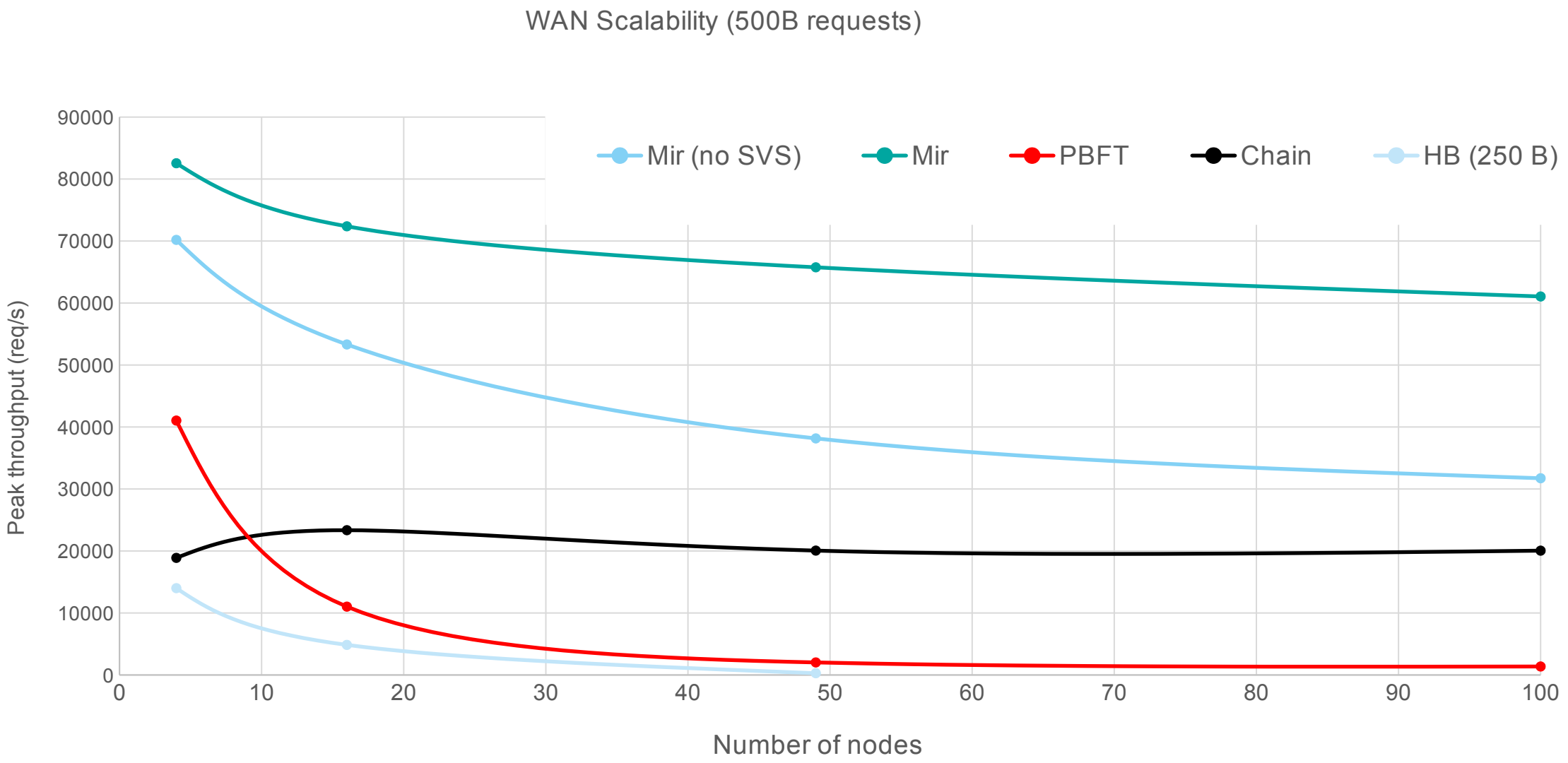
Performance Evaluation Setup

Chain

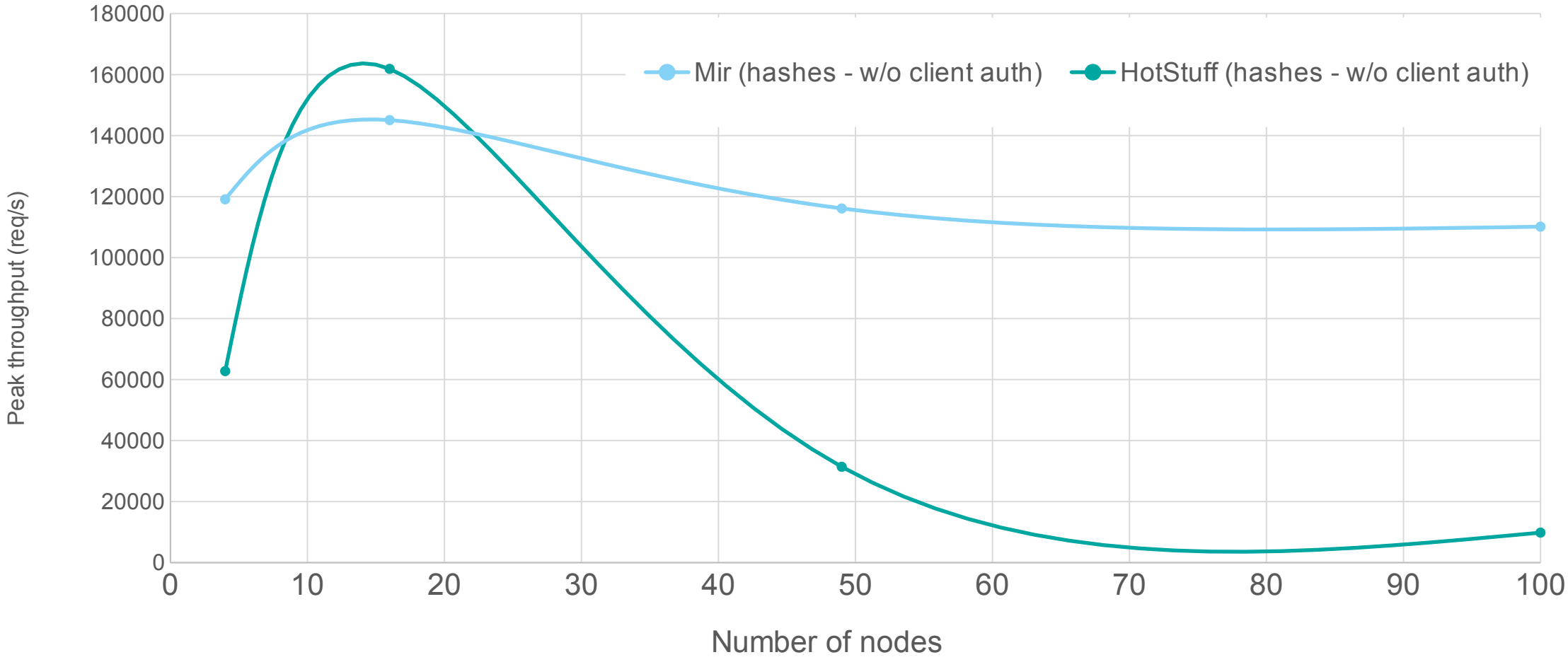


- **IBM Cloud (Softlayer)**
- **Baselines**
  - PBFT/Aardvark (on Mir codebase for fair comparison)
  - Chain (best-case, fault-free only, optimistic Chain BFT replication), Aublin et al. TOCS 2015
  - Honeybadger (HB) BFT (CCS 2016)
  - Libra HotStuff (PODC 2019)
- **WAN**
  - (up to) 16 datacenters across the world
  - 32 vCPU / 32 GB RAM 2.0Ghz VMs
  - 1Gbps links nominal full duplex bandwidth
  - Up to 100 nodes
  - 500 byte transactions (Bitcoin size) & 3500 bytes transactions (Fabric size)
- **LAN**

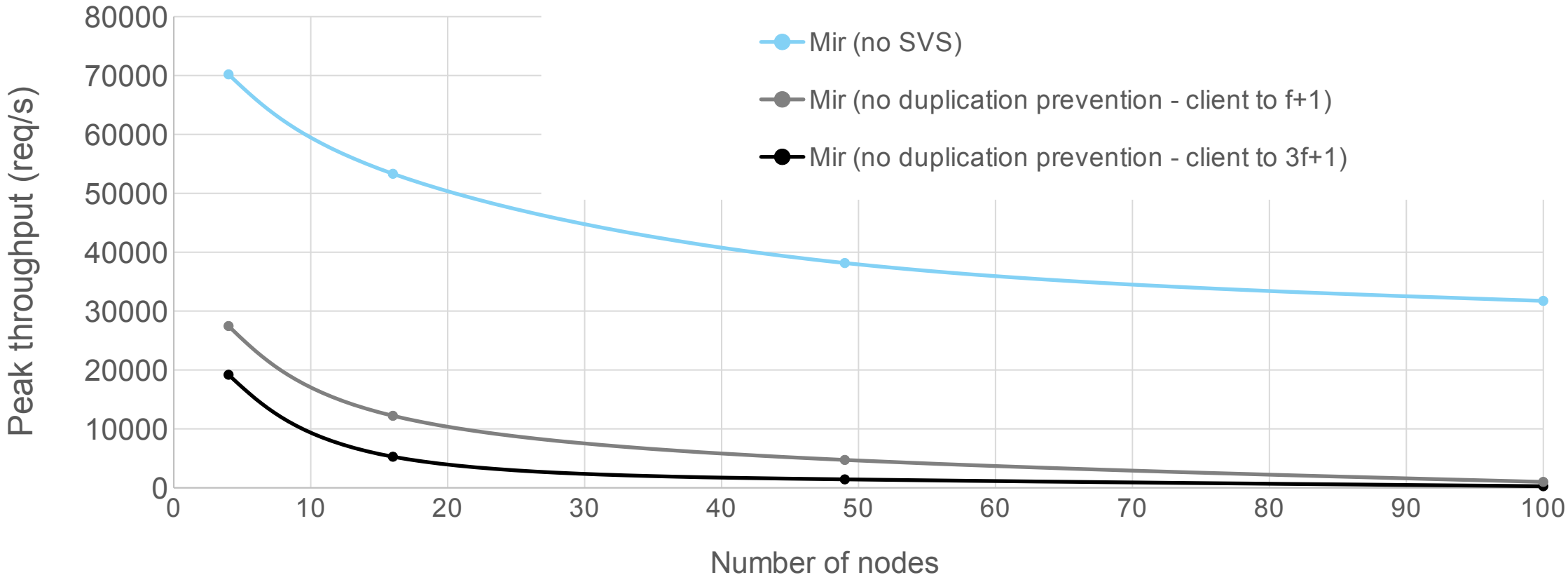
WAN throughput scalability (Bitcoin size txs. - 500 bytes)



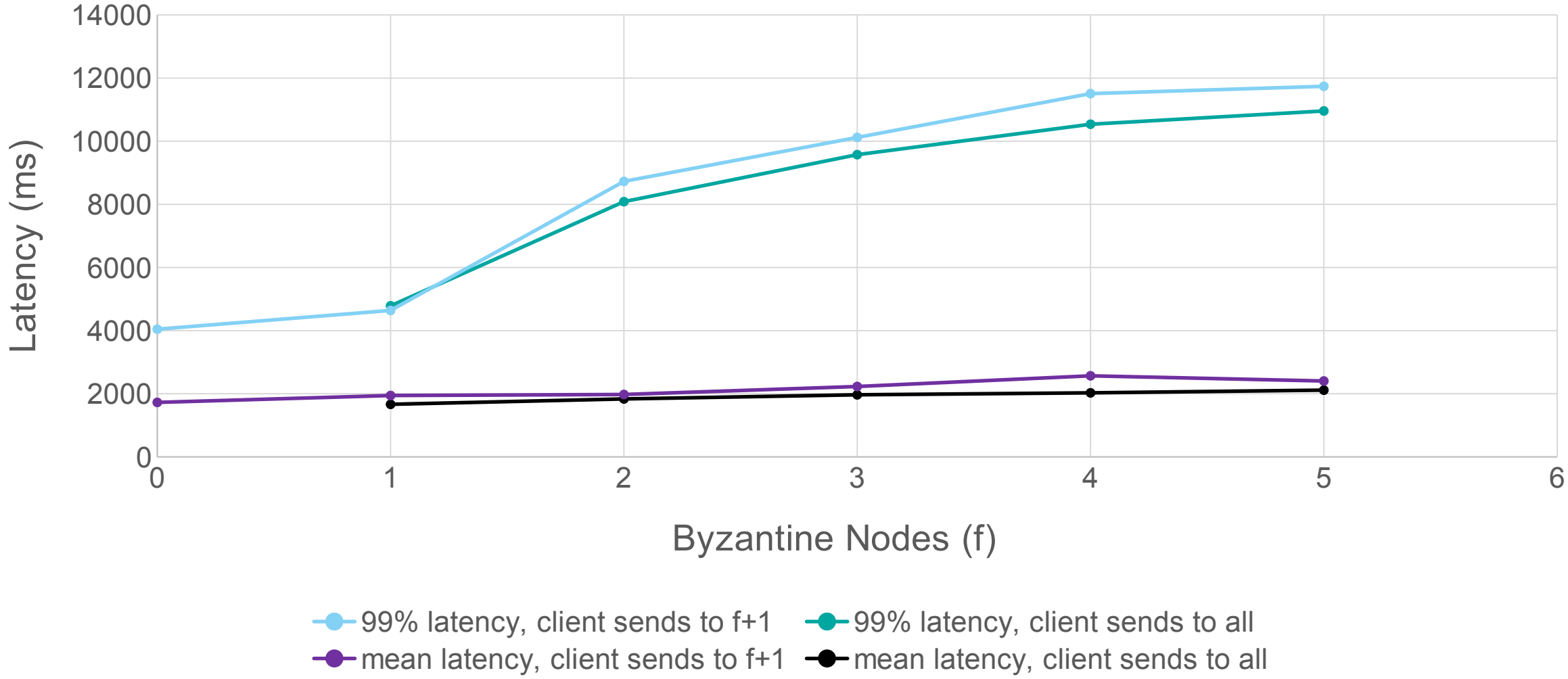
WAN Scalability vs. Hotstuff (500B, ordering hashes only, no client auth)



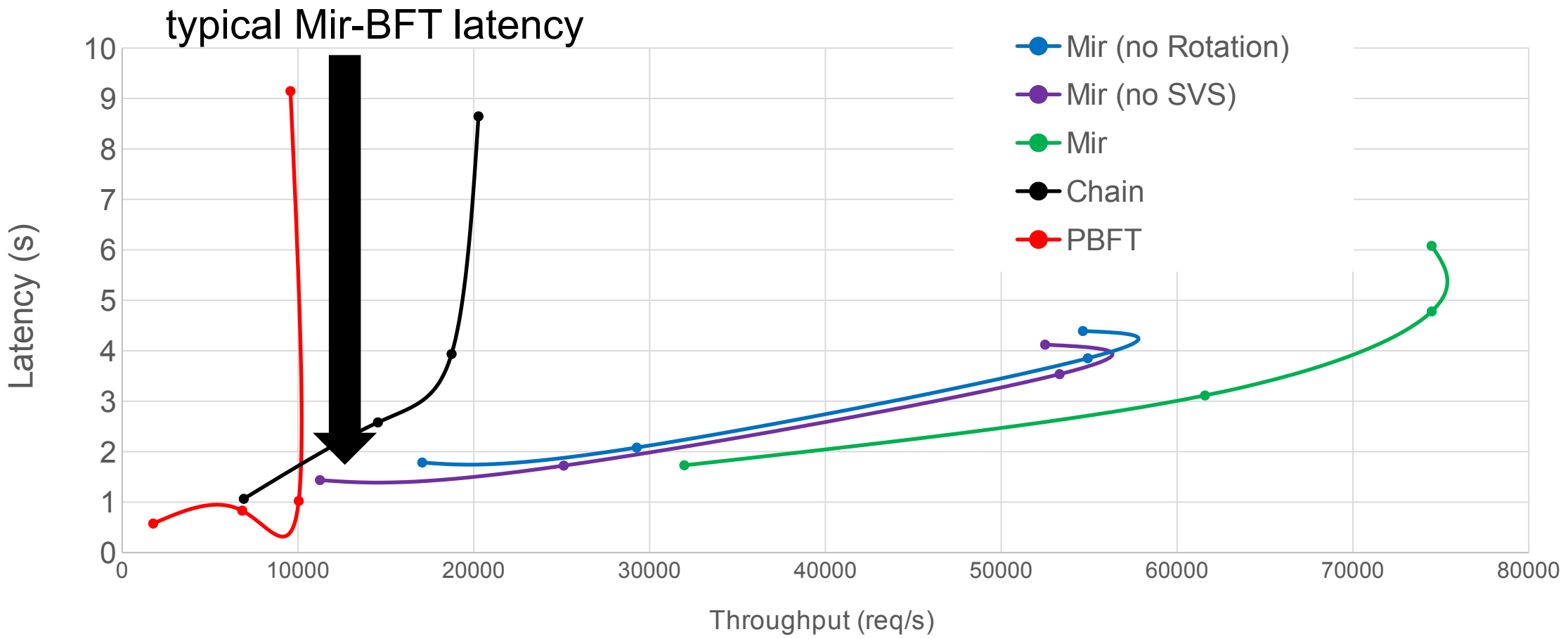
Impact of Duplication Prevention (500B requests)



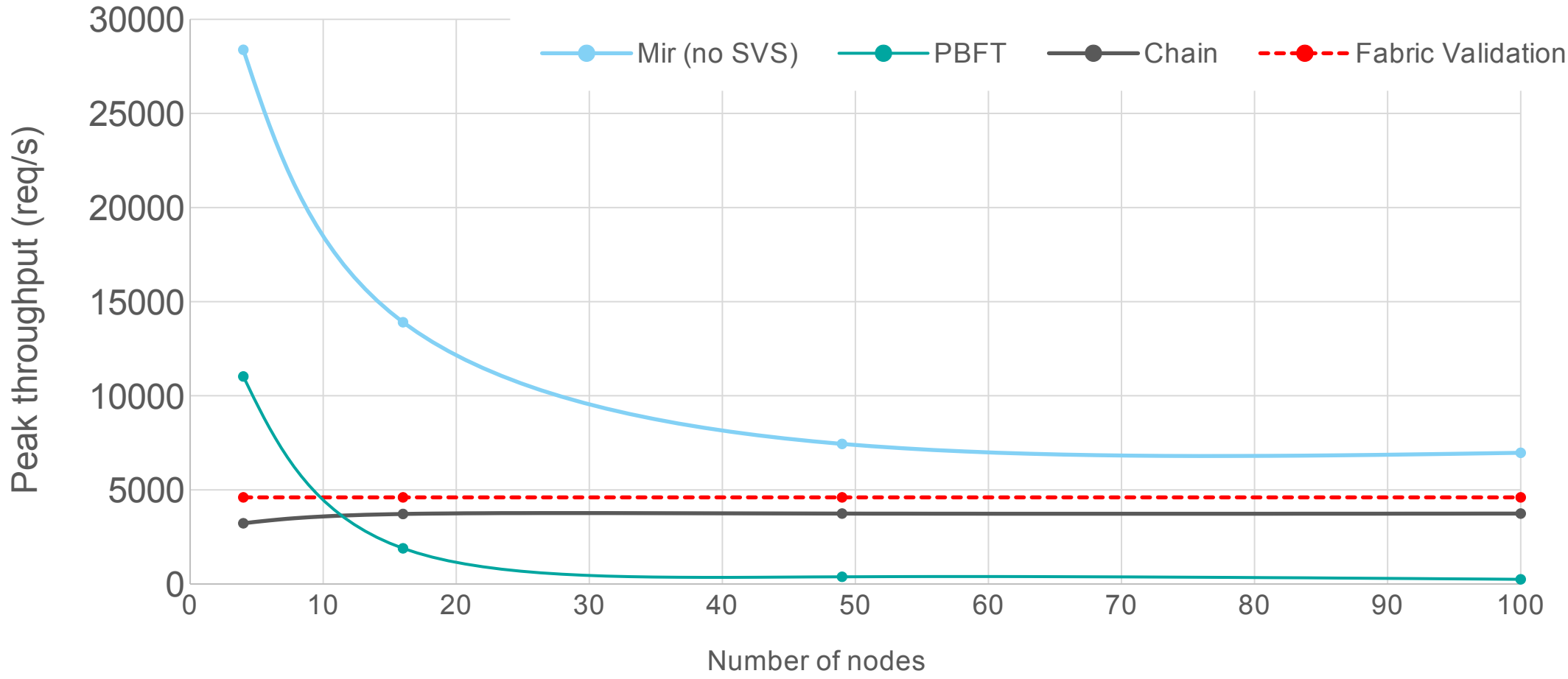
Censoring Resistance (20% dropped request by f nodes): n=16 nodes across 16 datacenters, 500 bytes



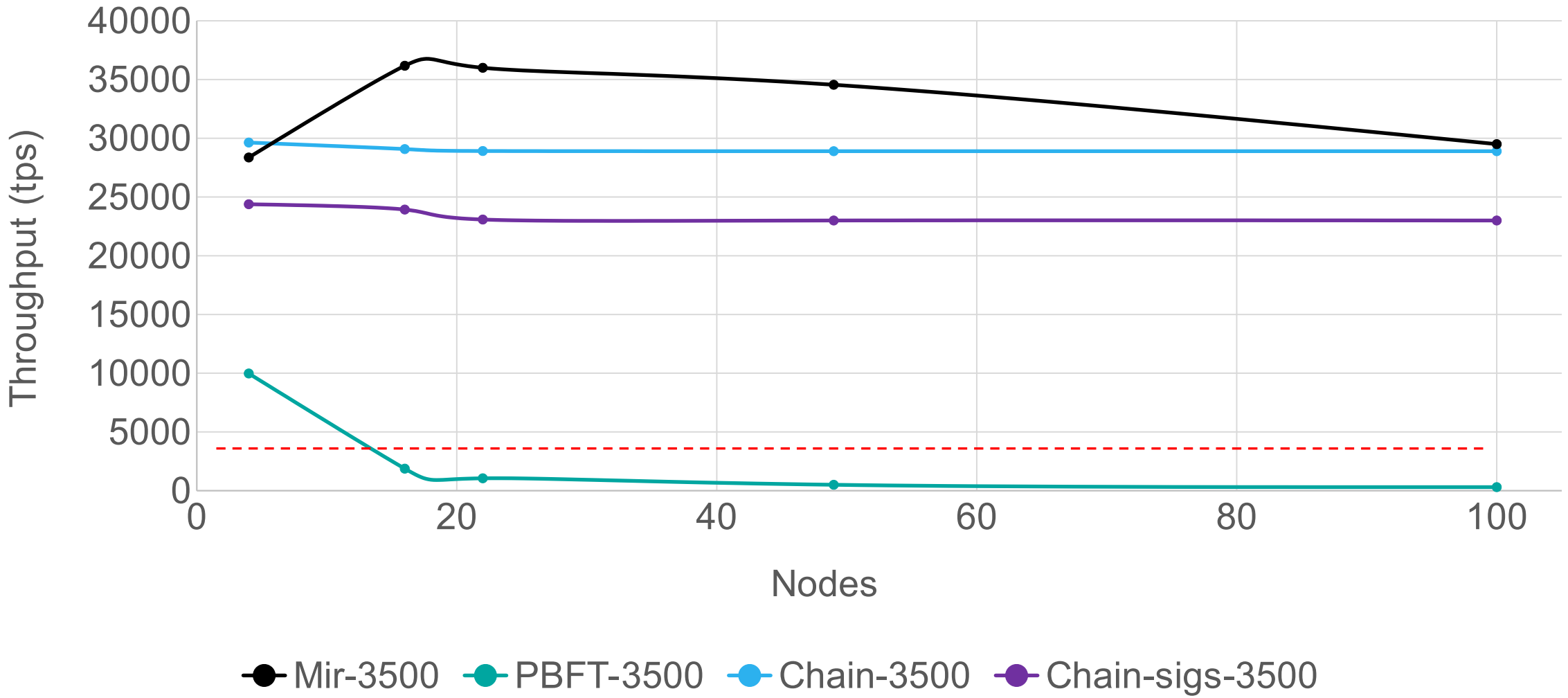
Latency, Impact of Bucket Rotation and SVS (16 nodes, WAN, 500B)



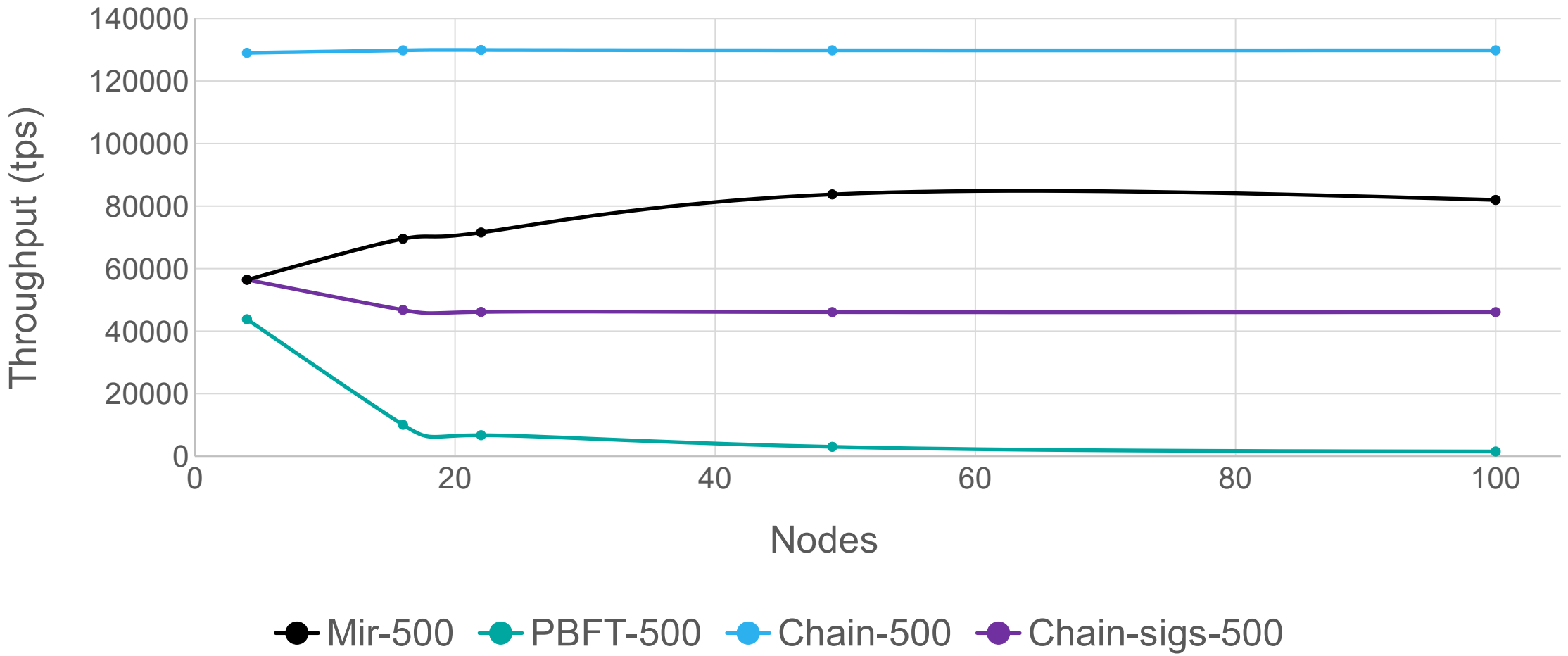
WAN Scalability (3500B requests – typical Hyperledger Fabric tx size)



LAN performance: 3500 bytes (Hyperledger Fabric size)



LAN performance: 500 bytes (Bitcoin size)



Summary

The paper is available on arXiV
https://arxiv.org/pdf/1906.05552.pdf

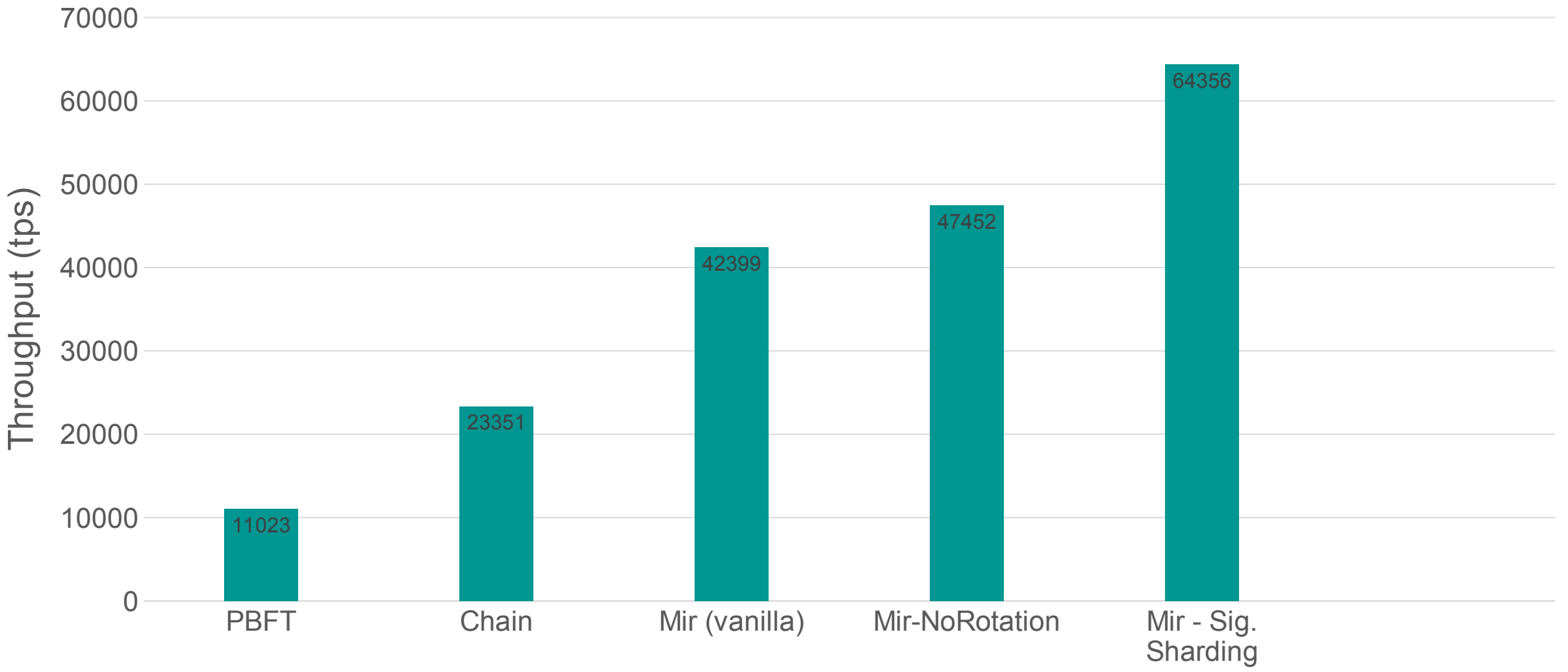
Mir-BFT

- **Excellent scalability (at least up to 100 nodes)**
- **Best performance to date on WANs, by far…**
- **Robust to performance attacks**
- **Open source, Apache 2.0**
- **a Hyperledger Lab since Apr 2021**
  - PoC, described in the paper, available on the research branch
  - Production implementation under way (main branch)
  - Planned integration into Hyperledger Fabric and possibly other Hyperledger projects

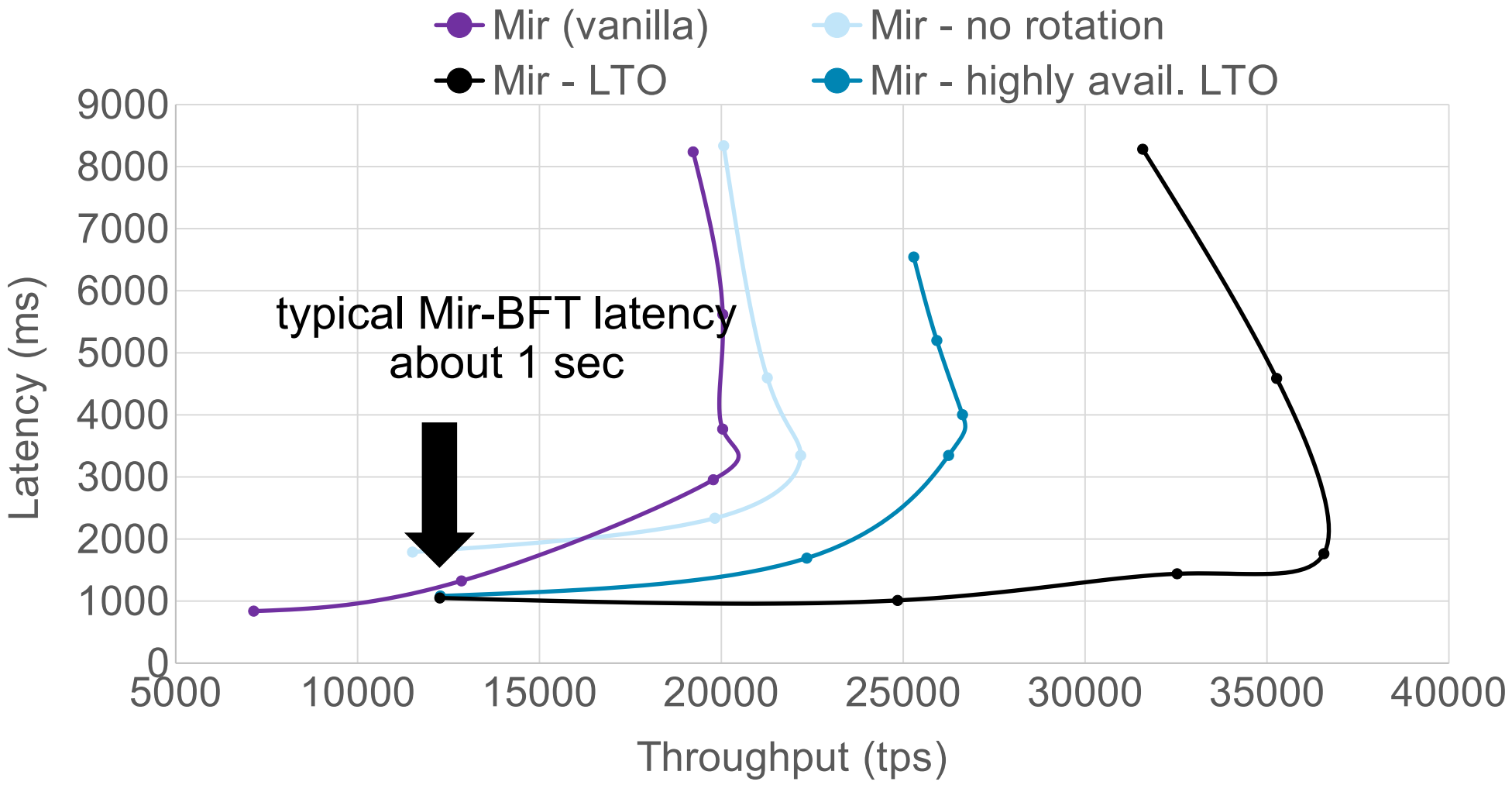
Join the community at

https://github.com/hyperledger-labs/mirbft

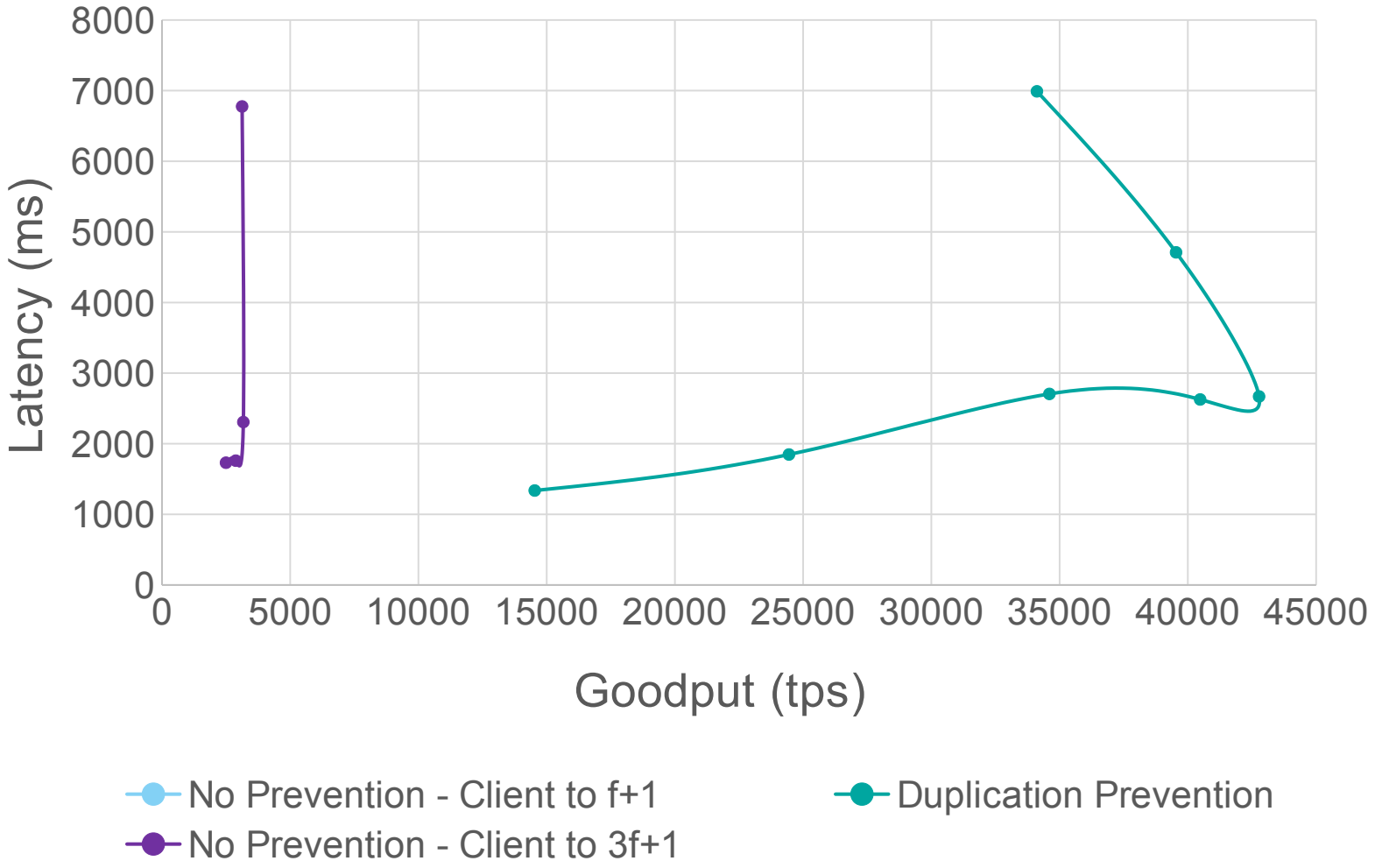
Impact of optimizations: 16 nodes across 16 datacenters, 500 bytes



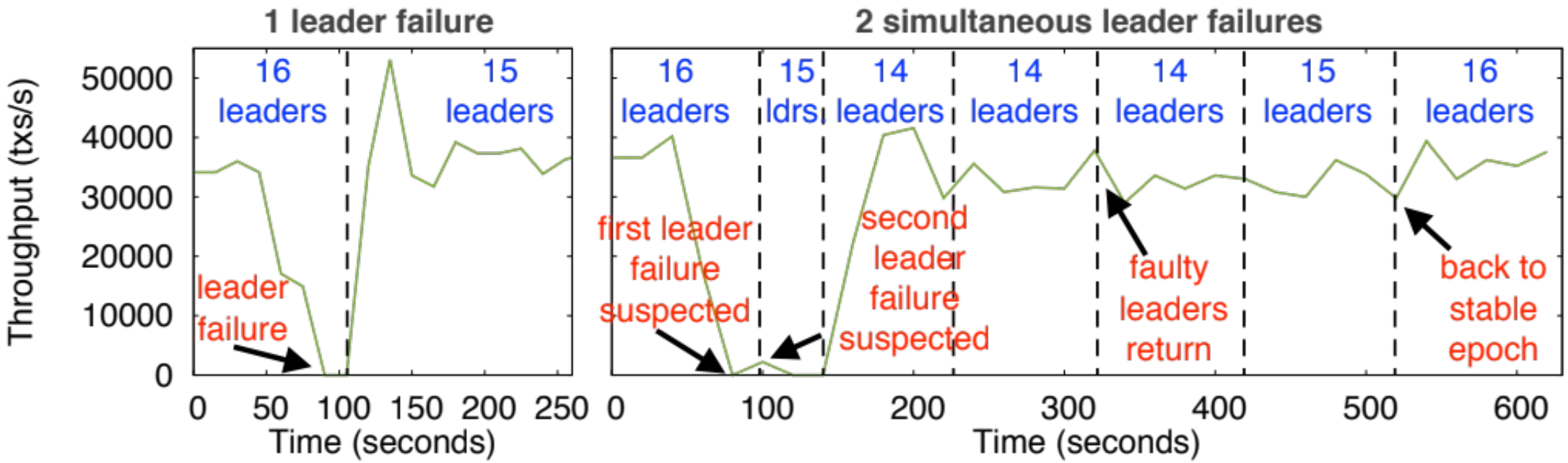
Impact of optimizations: n=16 nodes across 16 datacenters, 3500 bytes



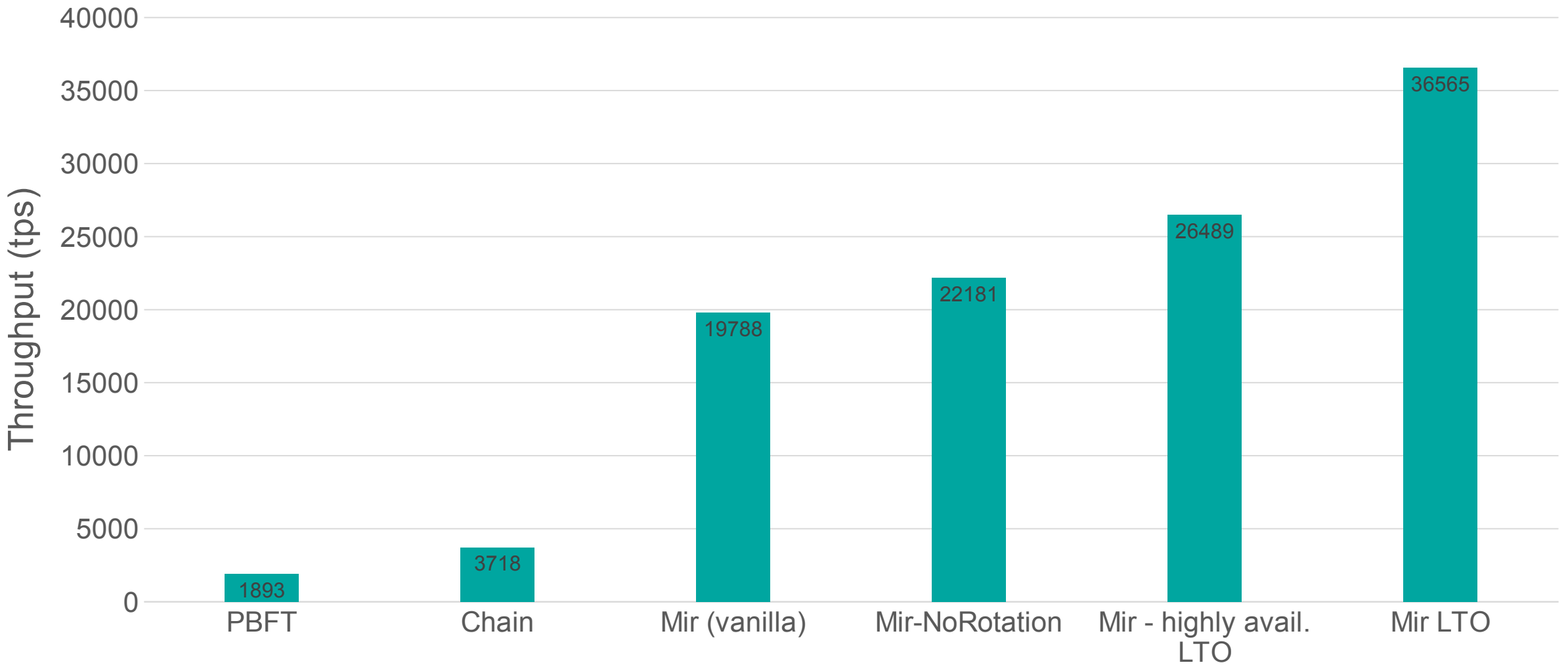
Duplication Prevention Impact: : n=16 nodes, 16 datacenters, 500 bytes



WAN performance with faults: 1-2 faults injected, 16 nodes, 500 bytes



Impact of optimizations: 16 nodes across 16 datacenters, 3500 bytes



Wrt HoneyBadger (WAN, 500 bytes)