Scaling State Machine Replication with Synchronized Clocks in the World with Byzantine Faults

Shiyu Liu
Feiran Wang, Yilong Geng, Balaji Prabhakar, Mendel Rosenblum
Stanford University

PLATFORMLAB

Huygens (NSDI 2018)

A software-based clock synchronization system

- Probe based: easy and fast to deploy
- Works with simple switches
- Only needs widely used timestamping capable NICs
- Light weight: runs distributedly in real time

Nanosecond synchronization accuracy

- Achieves accuracy of tens of nanoseconds
- Accuracy verified against NetFPGAs
- Evaluated in several production data centers in industry

Overview

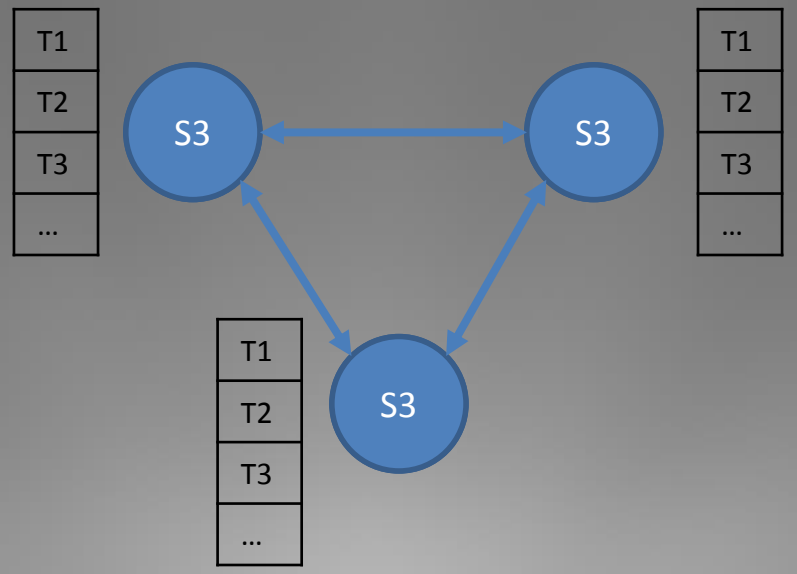
- Scaling State Machine Replication with Synchronized Clocks
  - Overview
  - Implementation in BFT-SMART
  - Experiment Evaluation

State Machine

- State
- Transaction

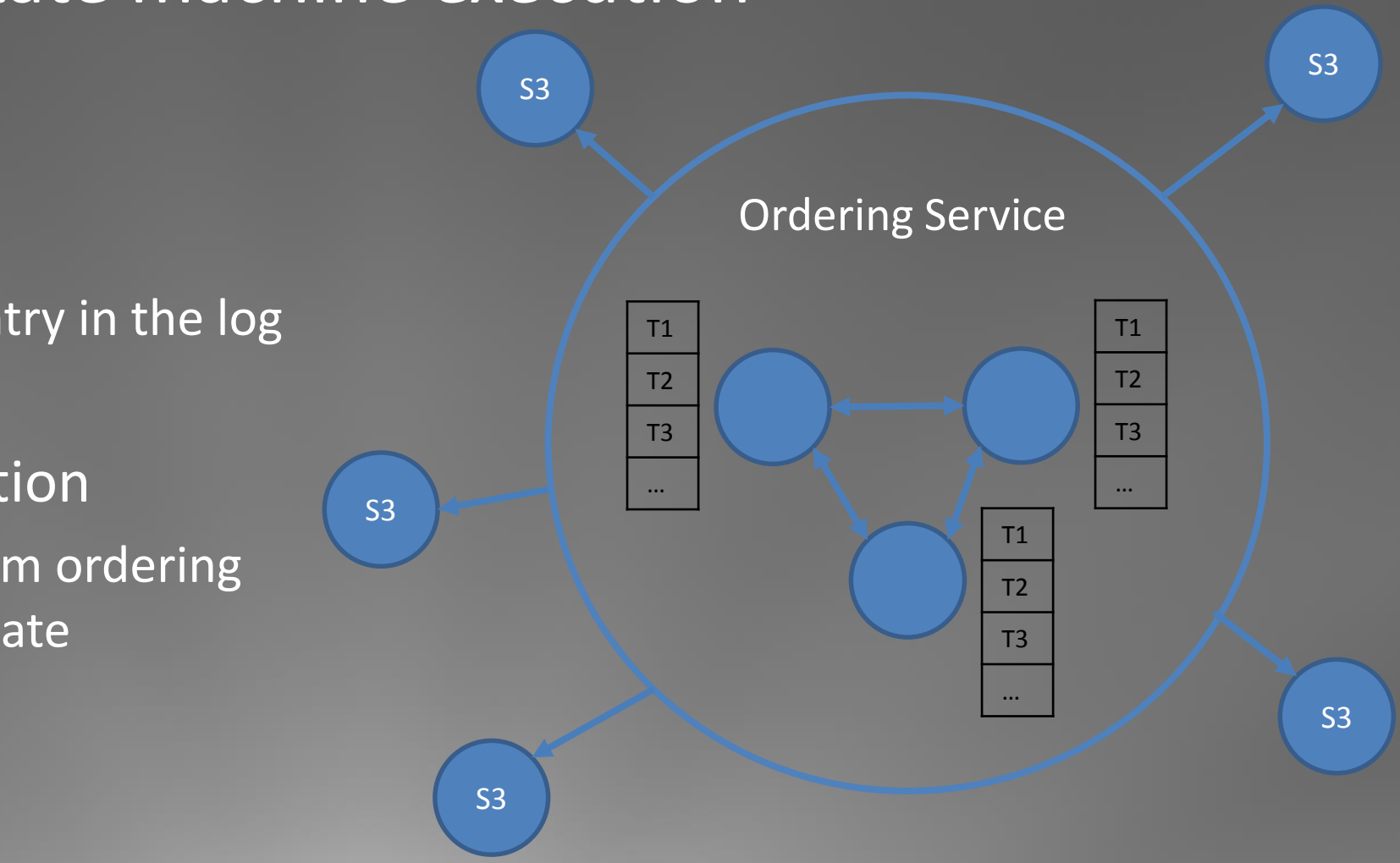
State Machine Replication

- Replicated state machine across nodes
  - For fault tolerance
  - For read throughput
- To guarantee all nodes end up at the same state:
  - All nodes agree on the order of transactions



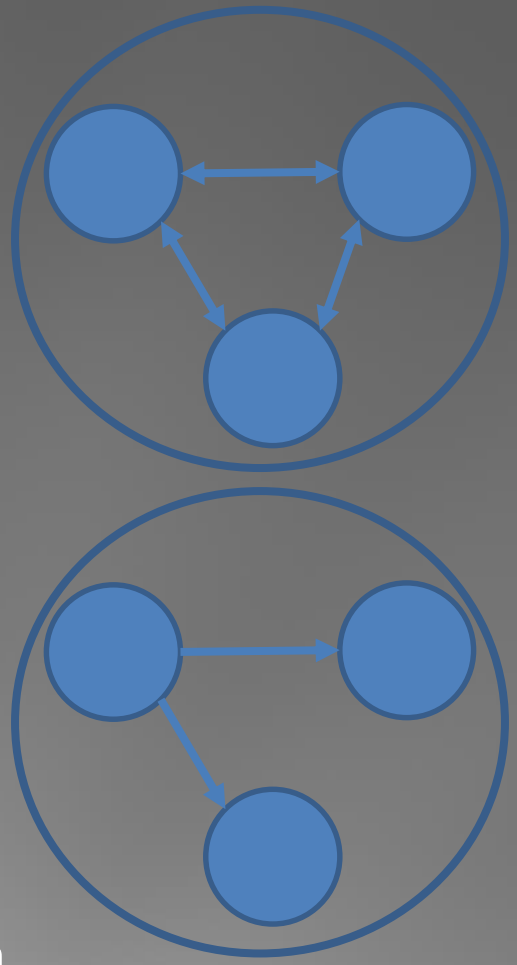
Separating ordering agreement and state machine execution

- Ordering service
  - Log replication
  - A transaction is an entry in the log
- State machine execution
  - Read transactions from ordering service and update state



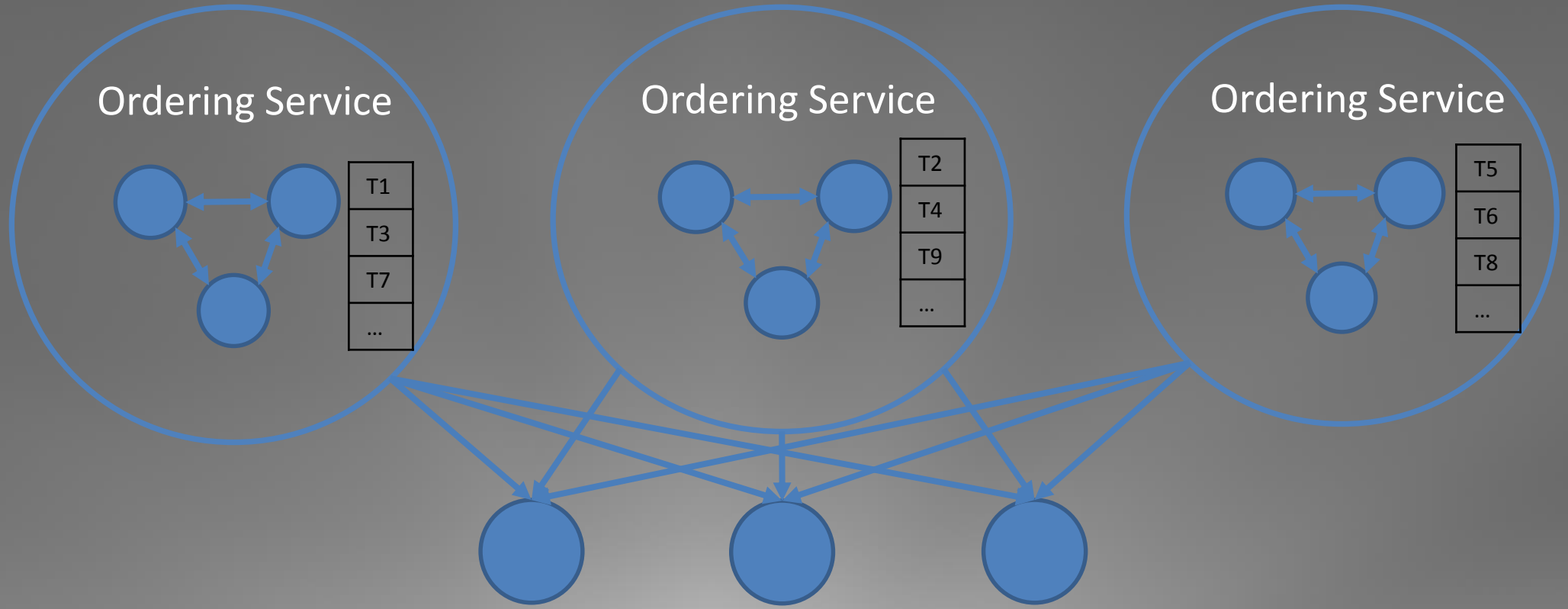
Replication algorithms

- System Hierarchy
  - Active replication
  - Primary-backup
- Fault tolerance
  - Crash Fault Tolerant
  - Byzantine Fault Tolerant
- Examples
  - Paxos, Raft
    - Tolerates n/2 crash failures
  - PBFT, BFT-SMART
    - Tolerates n/3 Byzantine failures
    - Complexity: O(n^2)
    - Write and read throughput decreases with n



Scaling Log-replication with Synchronized clocks

- Replication systems have been 1-d creatures
  - Single time axis derived from single log-replication service
- Synchronized clocks can make this multi-dimensional
  - Each log entry has a timestamp; consumers merge logs based on timestamps
  - Improves both throughput and latency due to multiple log-replication services



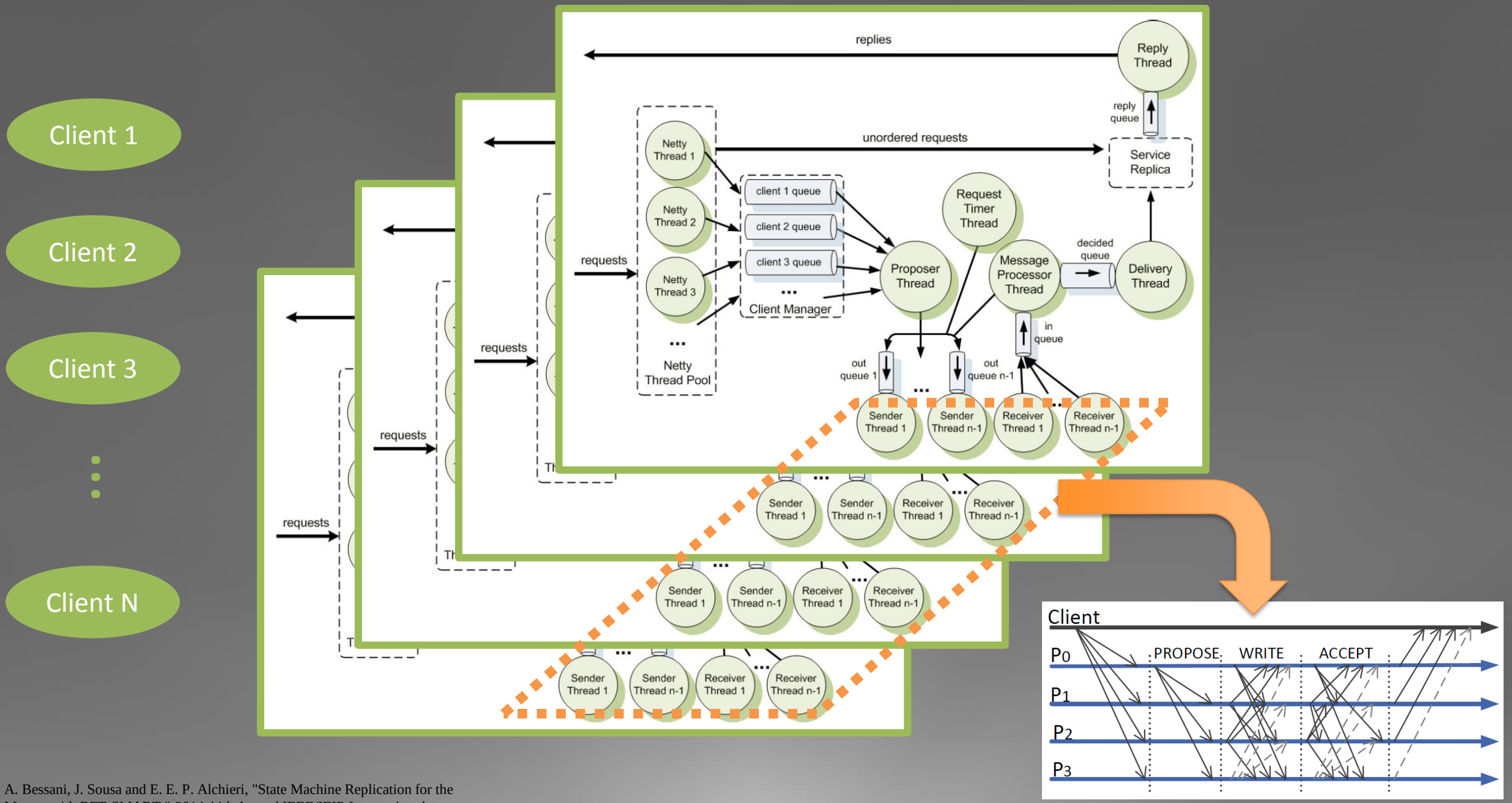
Overview

- Scaling State Machine Replication with Synchronized Clocks
  - Overview
  - Implementation in BFT-SMART
  - Experiment Evaluation

What is BFT-SMaRt

- BFT-SMaRt is an open-source library implementing robust Byzantine fault-tolerant (BFT) state machine replication (SMR).
  - Several improvements to PBFT: reliability, modularity, reconfiguration support, etc.
- Provide BFT ordering service for e.g. Hyperledger Fabric (HLF), a popular platform for distributed ledger (blockchain) solutions.
  - Ordering service is one of the key bottlenecks of the performance of distributed ledger solutions.

A. Bessani, J. Sousa and E. E. P. Alchieri, "State Machine Replication for the Masses with BFT-SMART," *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Atlanta, GA, 2014, pp. 355-362.
A. Bessani, J. Sousa and E. E. P. Alchieri, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, Las Vegas, NV, 2017



A. Bessani, J. Sousa and E. E. P. Alchieri, "State Machine Replication for the Masses with BFT-SMART," *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Atlanta, GA, 2014, pp. 355-362.

Single BFT group cannot scale: more machines increases redundancy, but throughput hurts

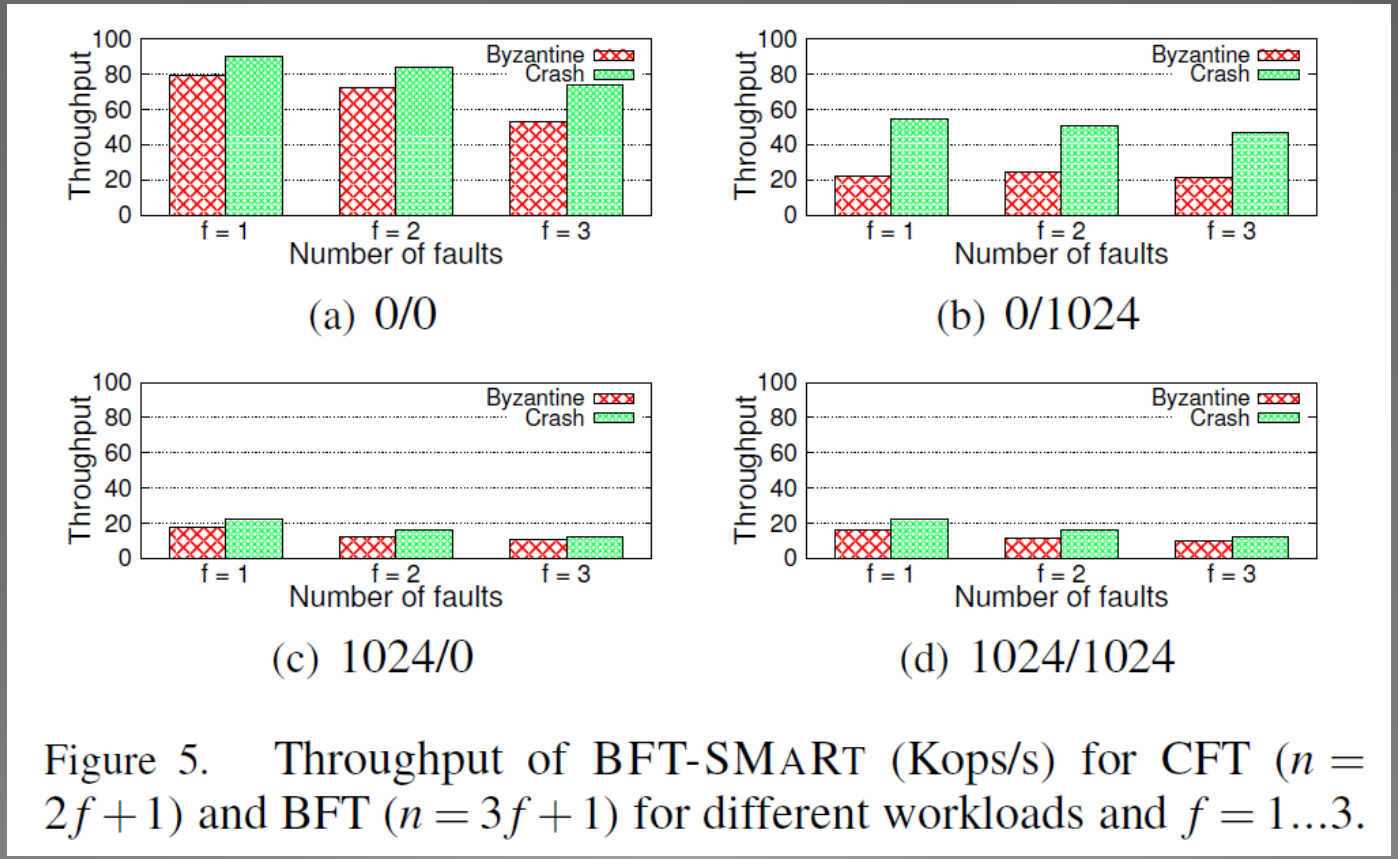


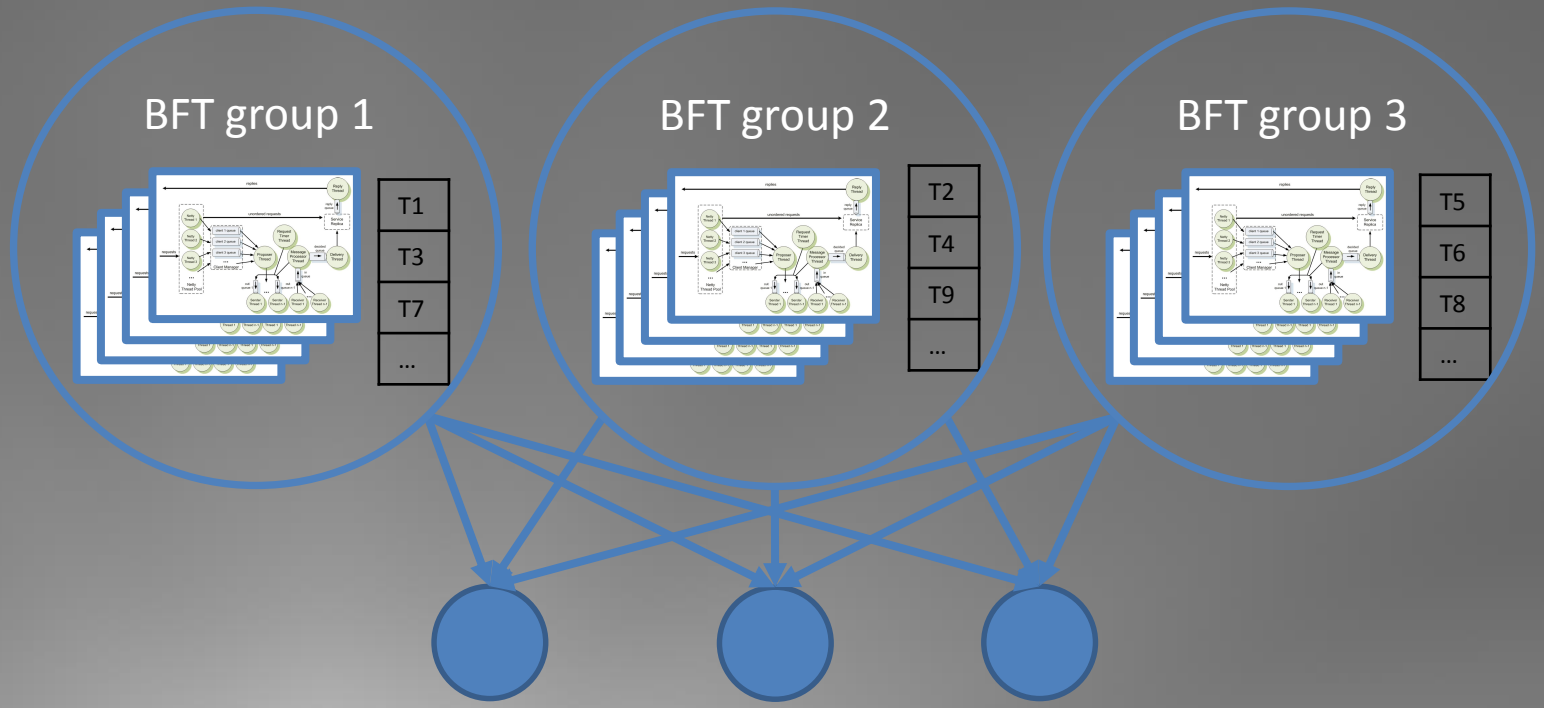
Figure 5. Throughput of BFT-SMART (Kops/s) for CFT ($n = 2f+1$) and BFT ($n = 3f+1$) for different workloads and $f = 1...3$.

A. Bessani, J. Sousa and E. E. P. Alchieri, "State Machine Replication for the Masses with BFT-SMART," *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Atlanta, GA, 2014, pp. 355-362.

Our proposal: Multiple BFT groups to provide ordering services for transaction logs

- Separating transaction ordering agreement and state machine execution.
  - Each BFT-group keeps the ordered logs of part of transactions.
  - The clients merge transaction logs by the leader-taken timestamps, then execute.

- The clients have two modes:
  - Producer
  - Consumer

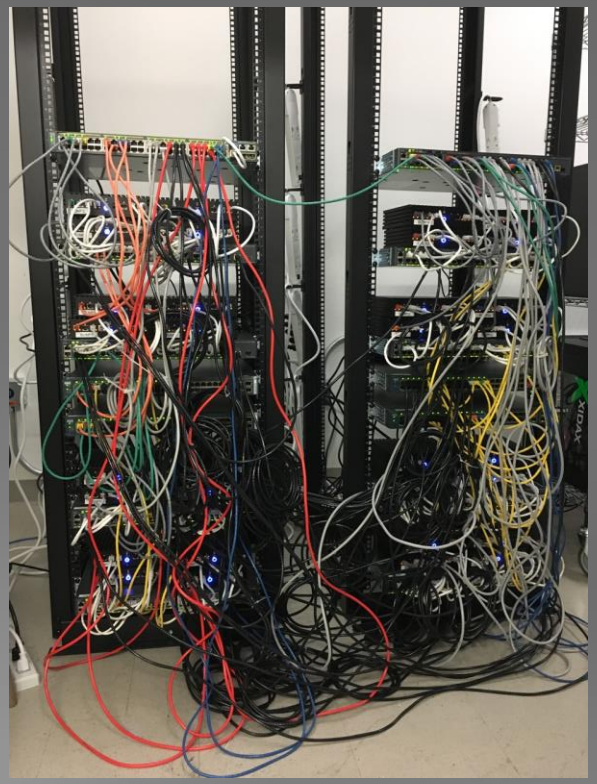


Overview

- Scaling State Machine Replication with Synchronized Clocks
  - Overview
  - Implementation in BFT-SMART
  - Experiment Evaluation

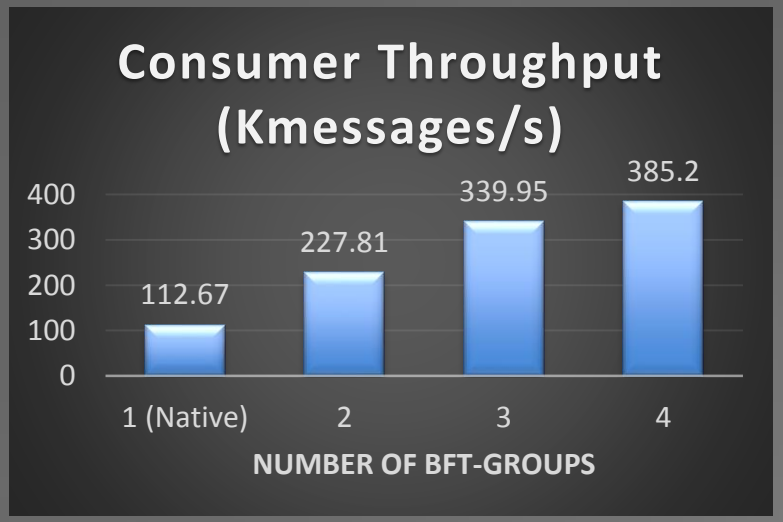
Experiment setup

- Use Stanford testbed. Run Huygens to sync clocks.
- Use BFT-smart to provide ordering service for transaction logs.
  - 4 replicas per BFT-group, each replica takes 1 server. # BFT-groups = 1,2,3,4
- Producer: 16 / 64 producers distributed on 16 servers.
  - Message size = 1KB. Batch size = 16KB
- Consumer: 64 consumers distributed on 16 servers.
  - Message size = 1KB. Batch size = 16KB



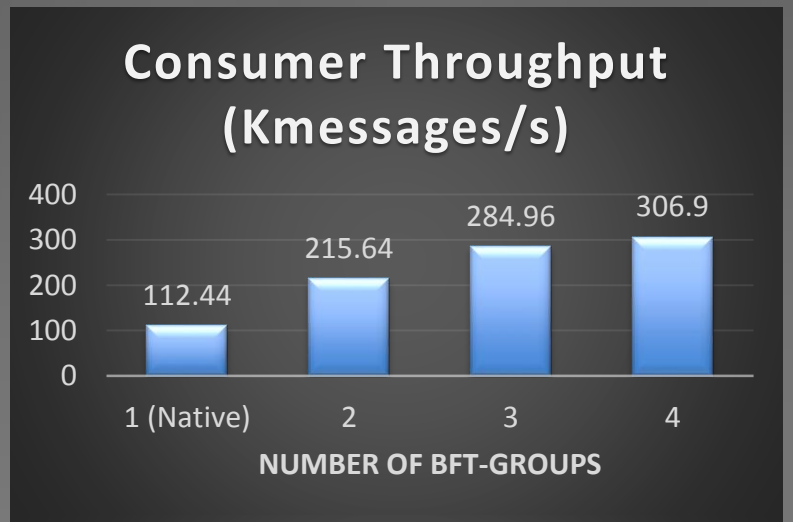
Experiment results: Consumer

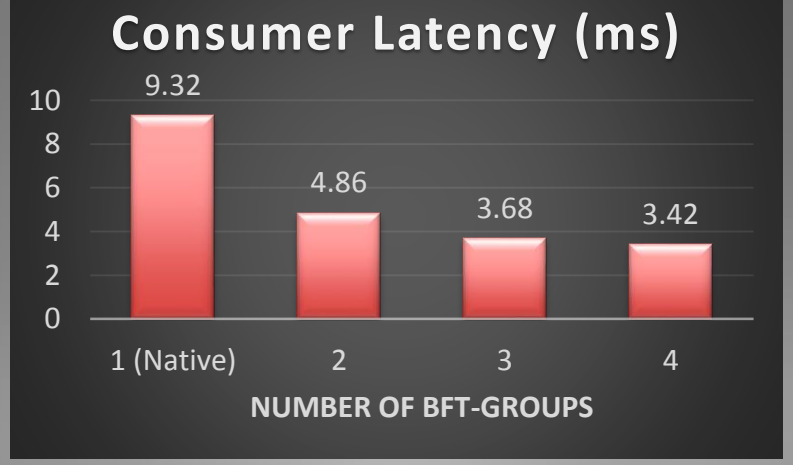
Streaming mode



Hard to define latency in streaming mode

On-demand mode

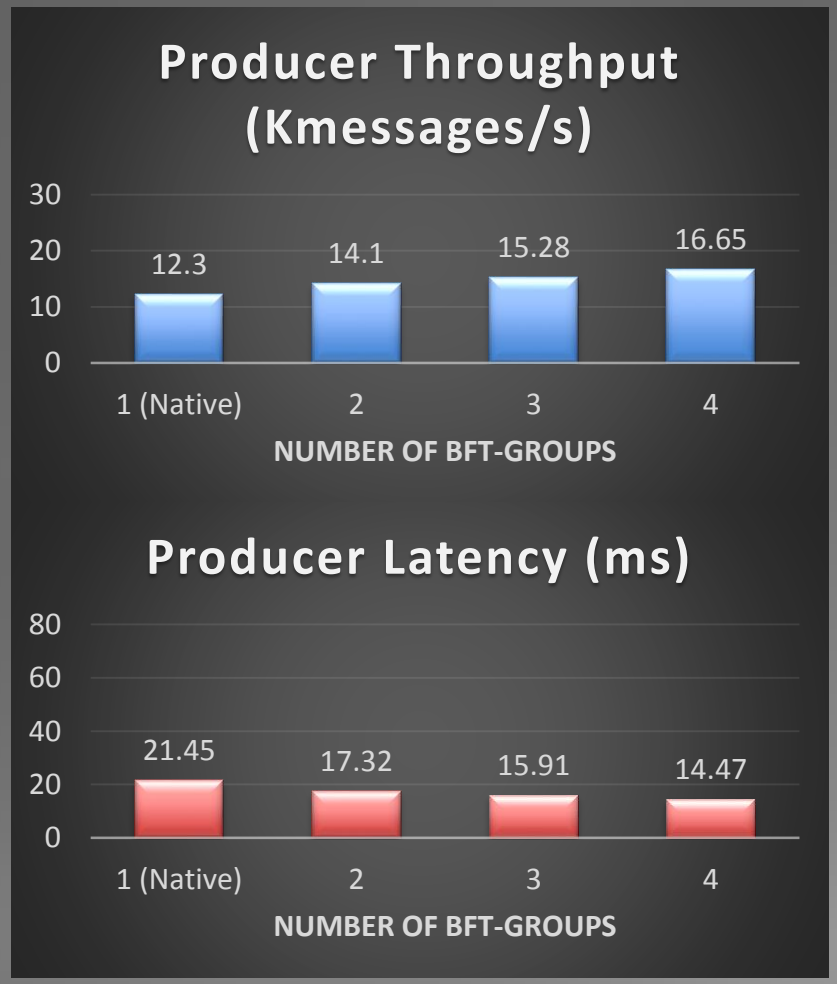




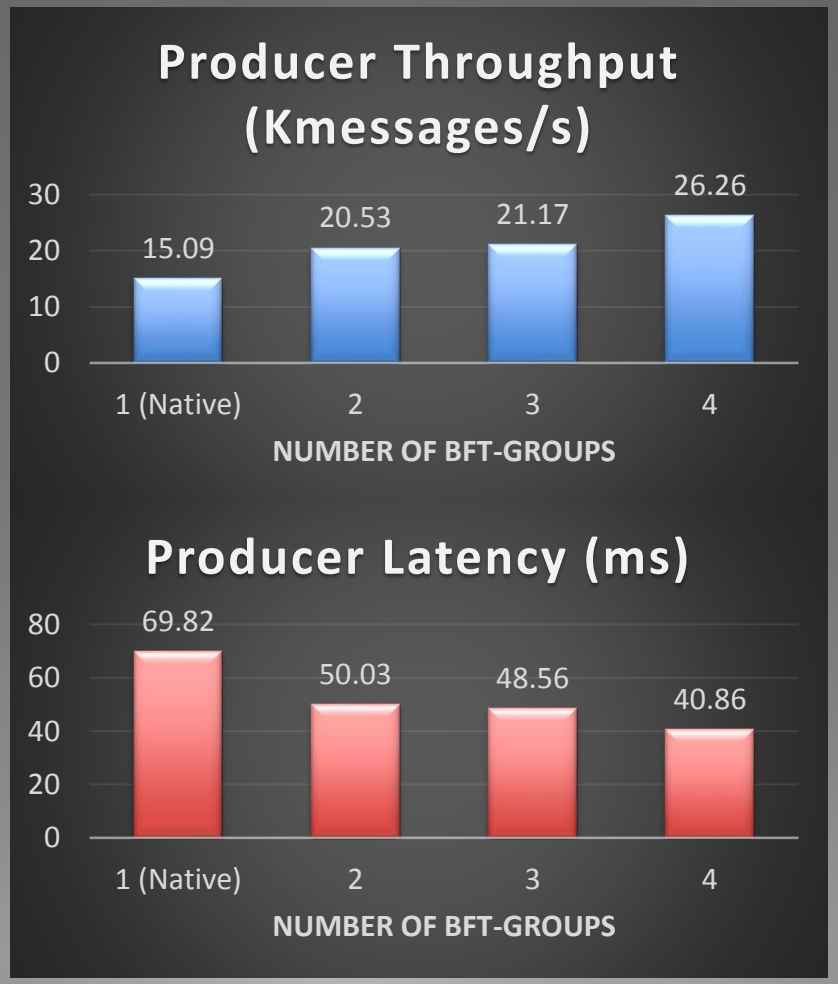
- Smaller latency and higher throughput when having multiple BFT-group.
- Linear scaling.
  - When # of BFT-groups goes up to 4, the bottleneck becomes the line rate (1Gbps) of consumers.

Experiment results: Producer

Low load
1 producer per box, 16 producers



Medium load
4 producers per box, 64 producers



- Smaller latency and higher throughput when having multiple BFT-group.
  - The load on each group is reduced.
- Not linear scaling: The BFT-groups are not saturated.
  - Limited by testbed

Summary

- Use synchronized clocks to scale state machine replication
  - Provide high throughput and low latency ordering service for distributed ledgers (blockchains)
  - General idea for both CFT and BFT replication algorithms
- Implement in BFT-SMART: significant improvements in latency and throughput for both producing and consuming