Zyzzyva, Proof of Execution, Practical Byzantine Fault Tolerance Discussion

PBFT

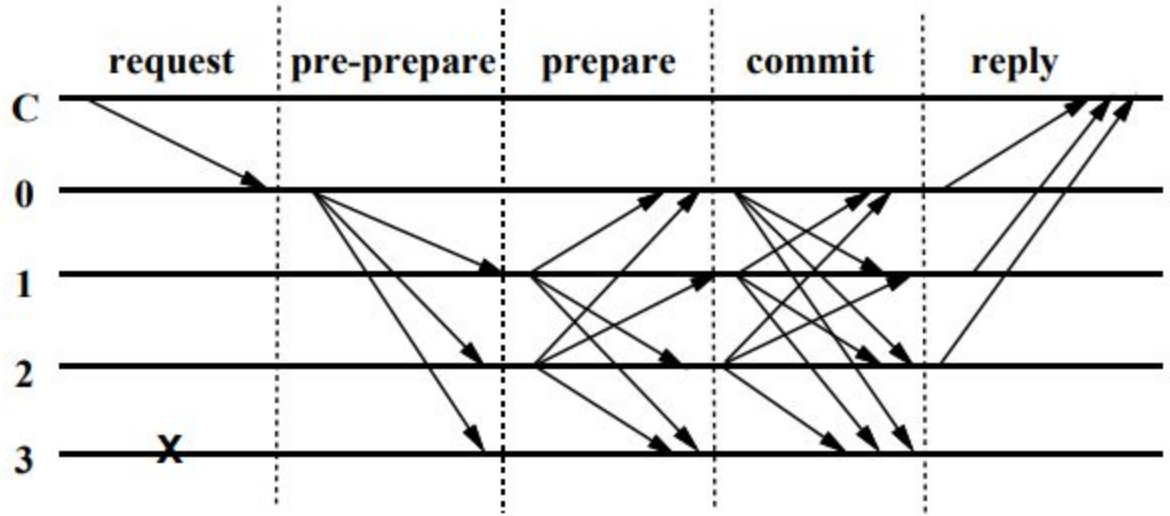
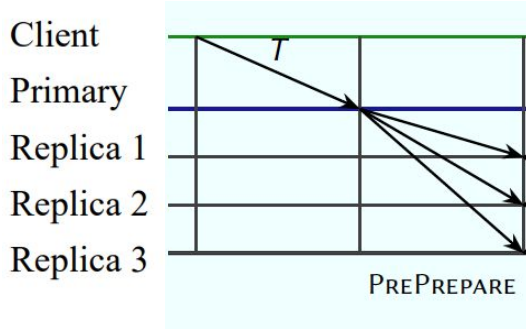
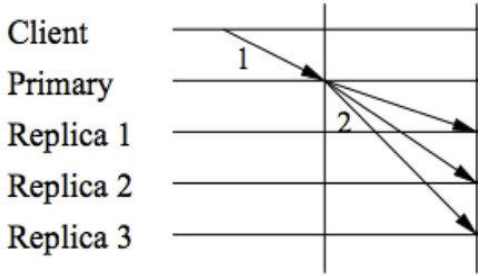


Figure 1: Normal Case Operation

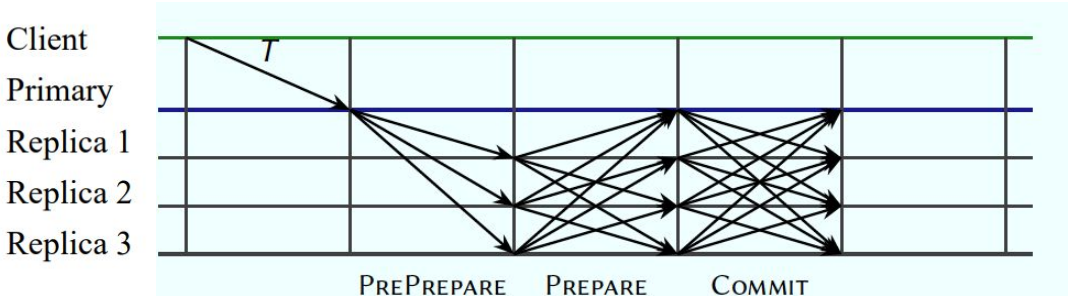
PBFT



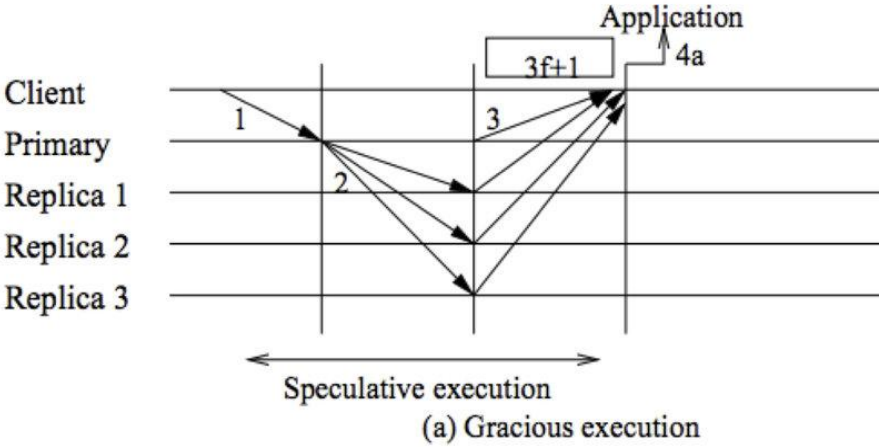
Zyzzyva



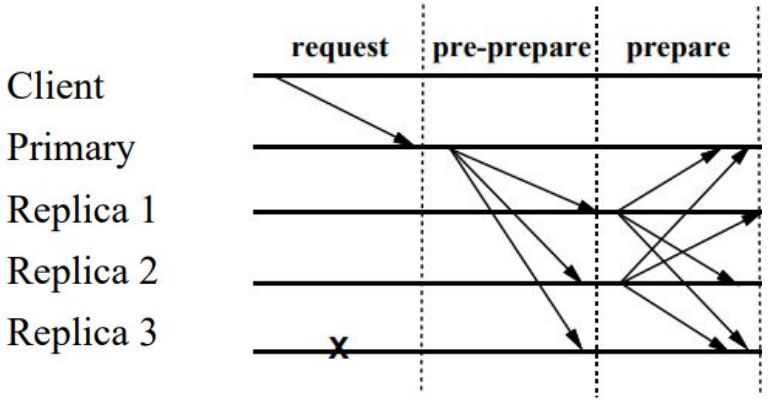
PBFT



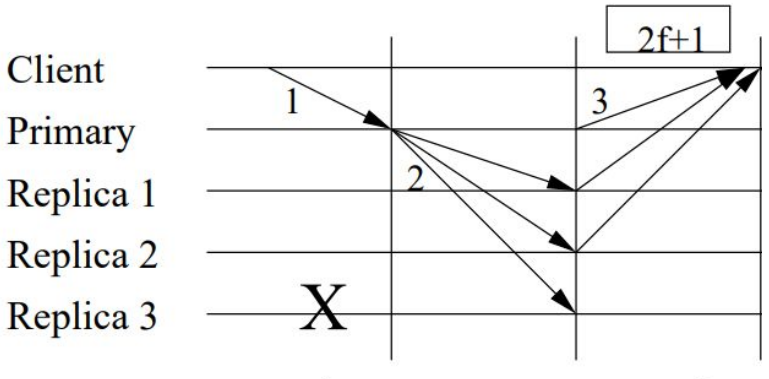
Zyzzyva



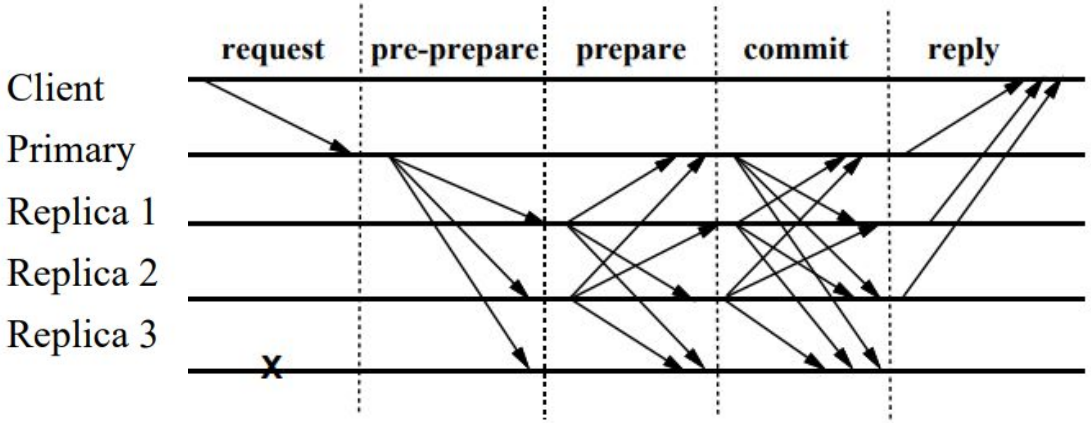
PBFT



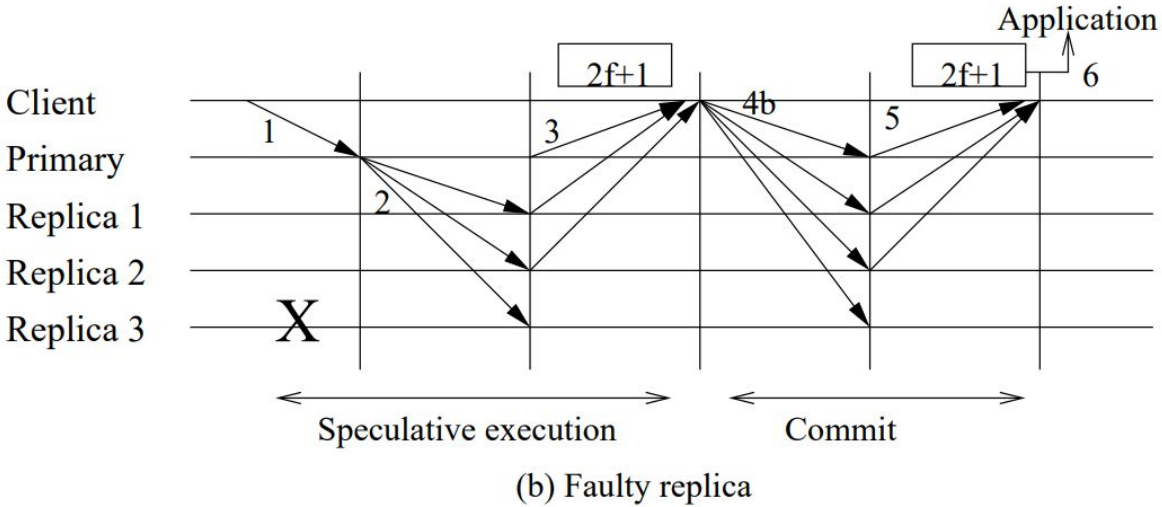
Zyzzyva



PBFT

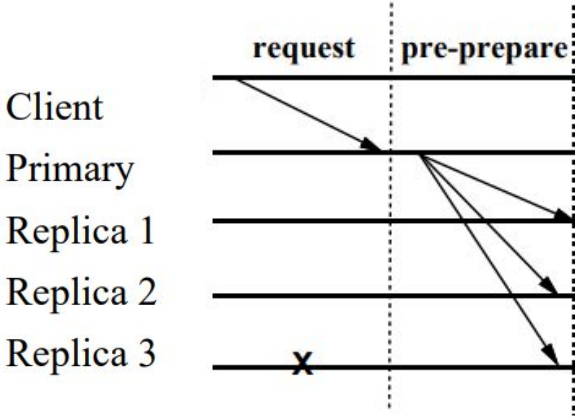


Zyzzyva

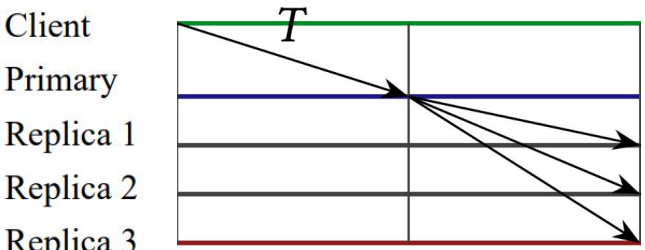


(b) Faulty replica

PBFT

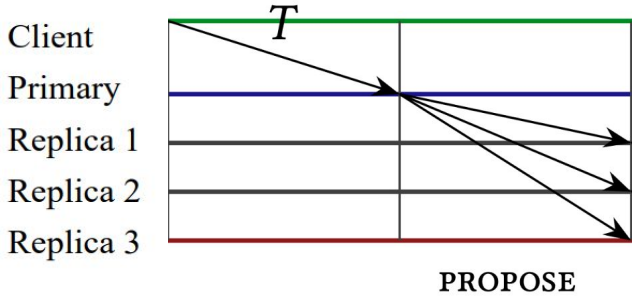


MAC

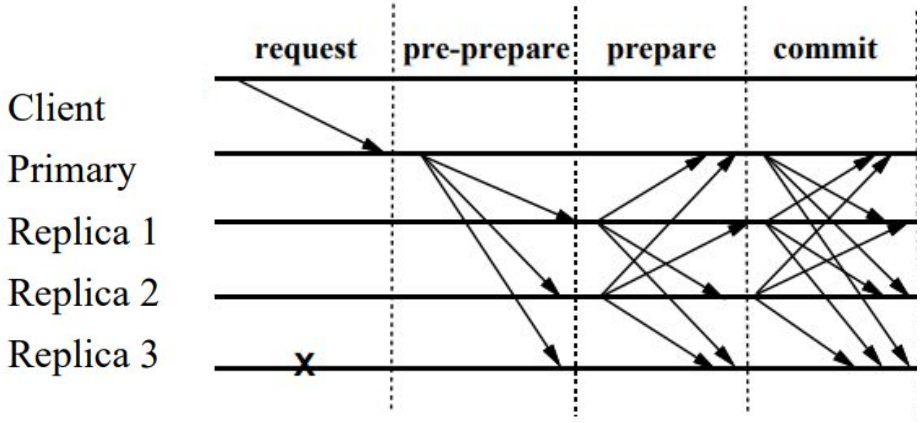


POE

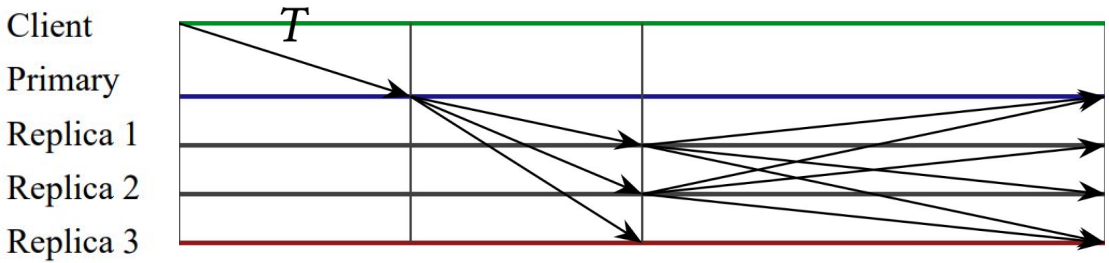
TSS



PBFT

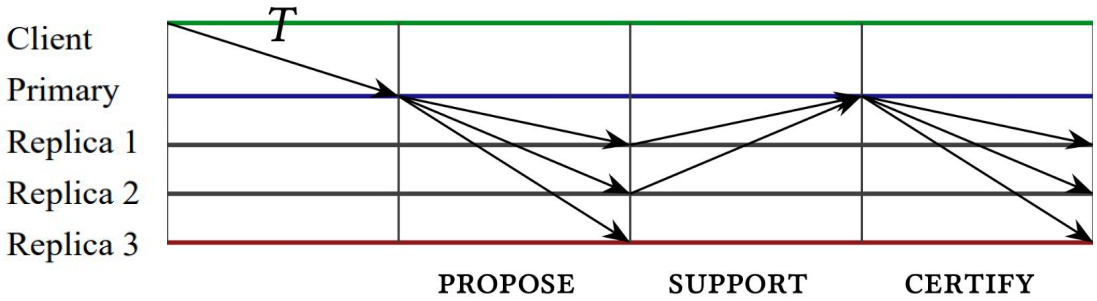


MAC

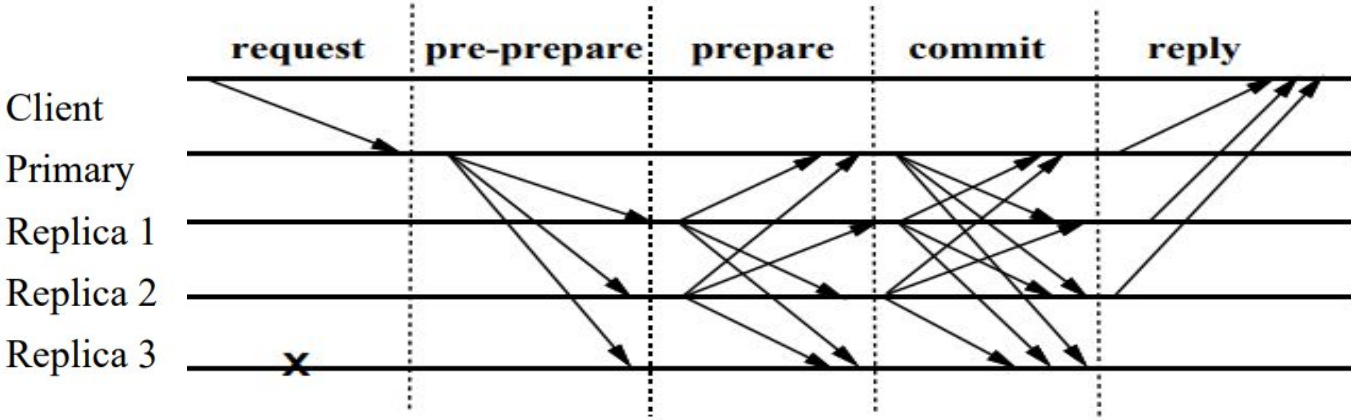


POE

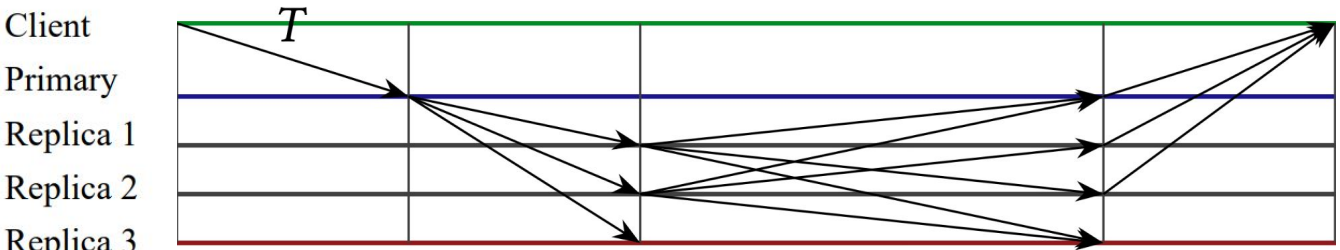
TSS



PBFT

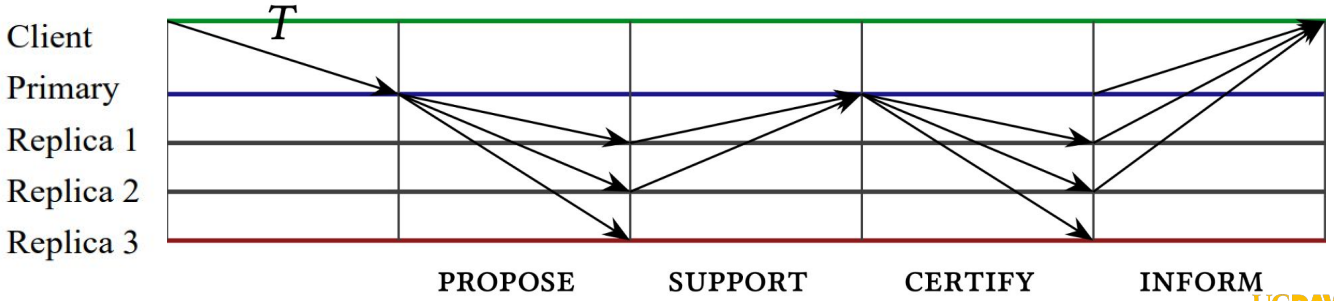


MAC

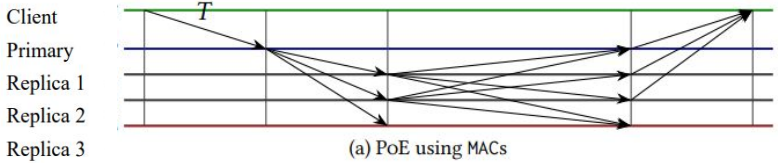


POE

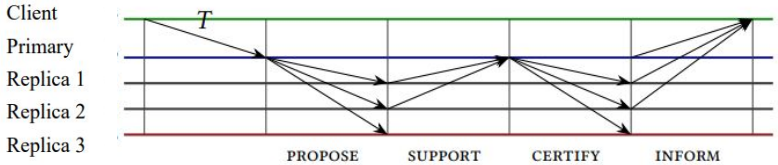
TSS



POE

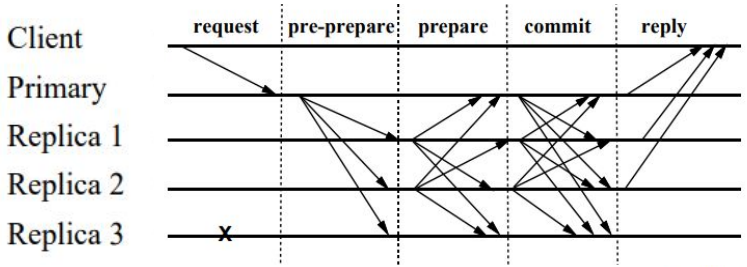


(a) PoE using MACs

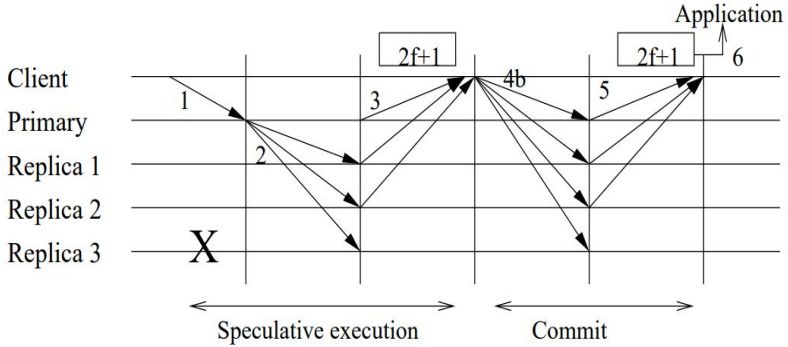


(b) PoE using TSs.

PBFT



Zyzzyva



(b) Faulty replica

Zyzzyva: View Change

View Change Protocol

- Guarantees of electing a **new primary**.
- It requires a **correct replica**
  - To observe the **primary as faulty**
  - Possess **evidence** that **f+1 replicas have committed** to a View-Change.
- Traditional methods (Byzantine) needs to **maintain the history** of the requests.
- Additional properties of View-Change Protocol in Zyzzyva
  - **Liveness**
  - **Safety**

Liveness: Missing Phase

- **Strengthens** commitment of correct replica by adding **“I hate new Primary” phase**.
- Liveness brings “Prepare” and “Commit” phase into one single phase (following PBFT).
- Every **correct** replica has to abandon a view for view-change.

- Procedure
  - A correct replica uses **voting system (no-confidence votes)** across other correct replica to identify faulty primary.
  - If atleast **f+1** votes, then View-Change is started to change the primary.
  - They use the rule **I-Hate-New-Primary** rule to elect for a faulty primary.

Safety: Uncommitted Request

- Weakens when request appears in the history included in the message.
- In case of **2f + 1** responses received by client
  - **Correct replica** has at least f+1 commit certificates
  - VIEW-CHANGE will contain all the commit certificates received from a LOCAL-COMMIT.

- In case of **3f + 1** responses received by client
  - **No commit** certificate in this case
  - All ORDER-REQ messages are saved in all the correct replicas.
  - **New Primary** would include all the ORDER-REQ messages into its history.

- There is no case where 2 completed requests can have the **same sequence number** by correct replica. This is a benign case of preserving safety.

Section 3: Correctness

1. **Safety**:
   a. **Agreement protocol is safe within single view**
      i. No Two requests complete with the same sequence number n
      ii. h_n is a prefix of $h_{n'}$ for n < n' and completed request r and r'
   b. **Agreement and view protocol together ensure safety across views**

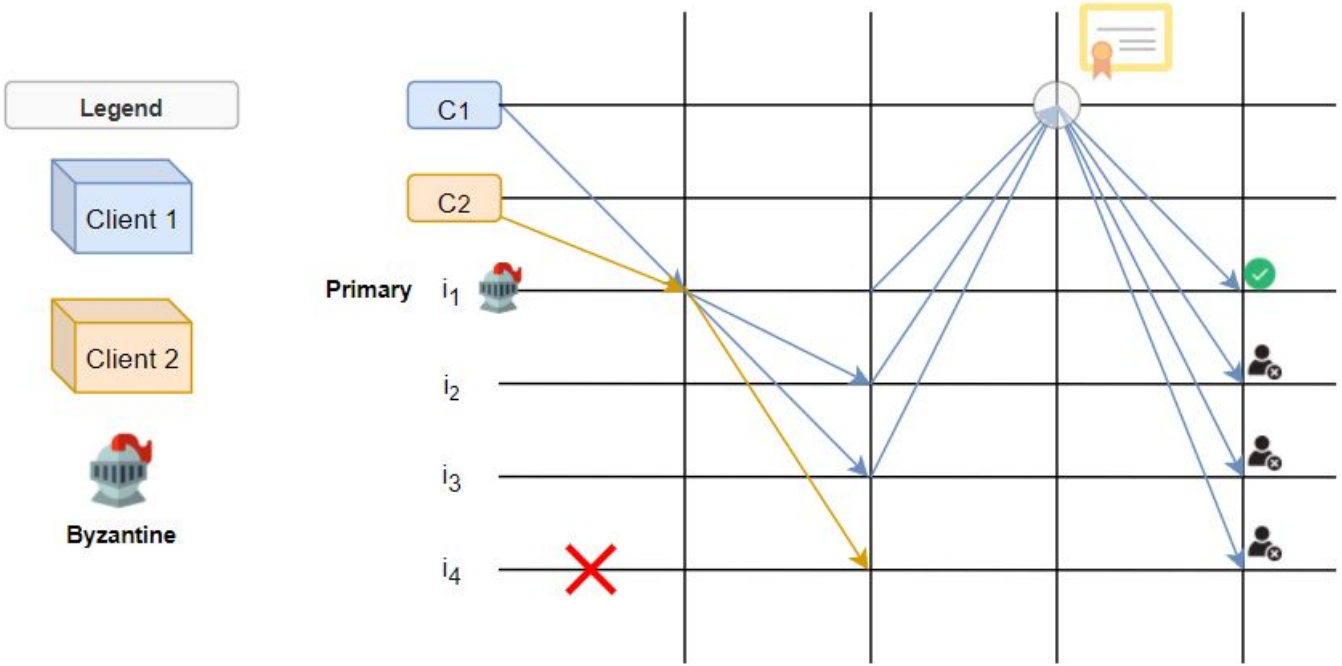
2. **Liveness**: **Zyzzyva guarantees liveness only during periods of synchrony**
   a. If **primary** is **correct** when a correct client issues the request, then the **request completes**
   b. If a request from a correct client **does not complete** during the **current view** then the **view change does not complete** during the **current view** then a view change occurs

Zyzzyva

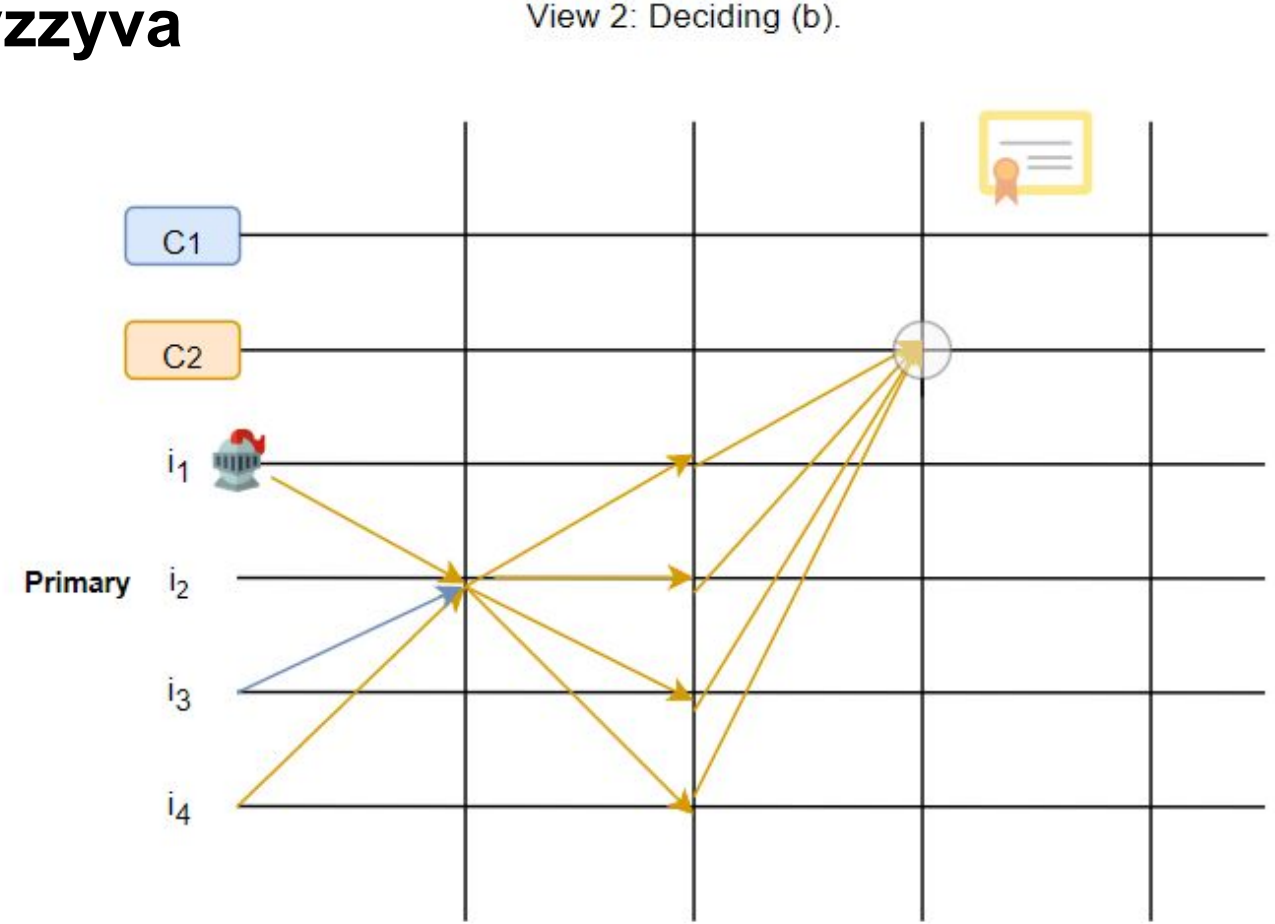
Breaking Safety

Scenario 1

View 1: Creating a commit-certificate for (a).

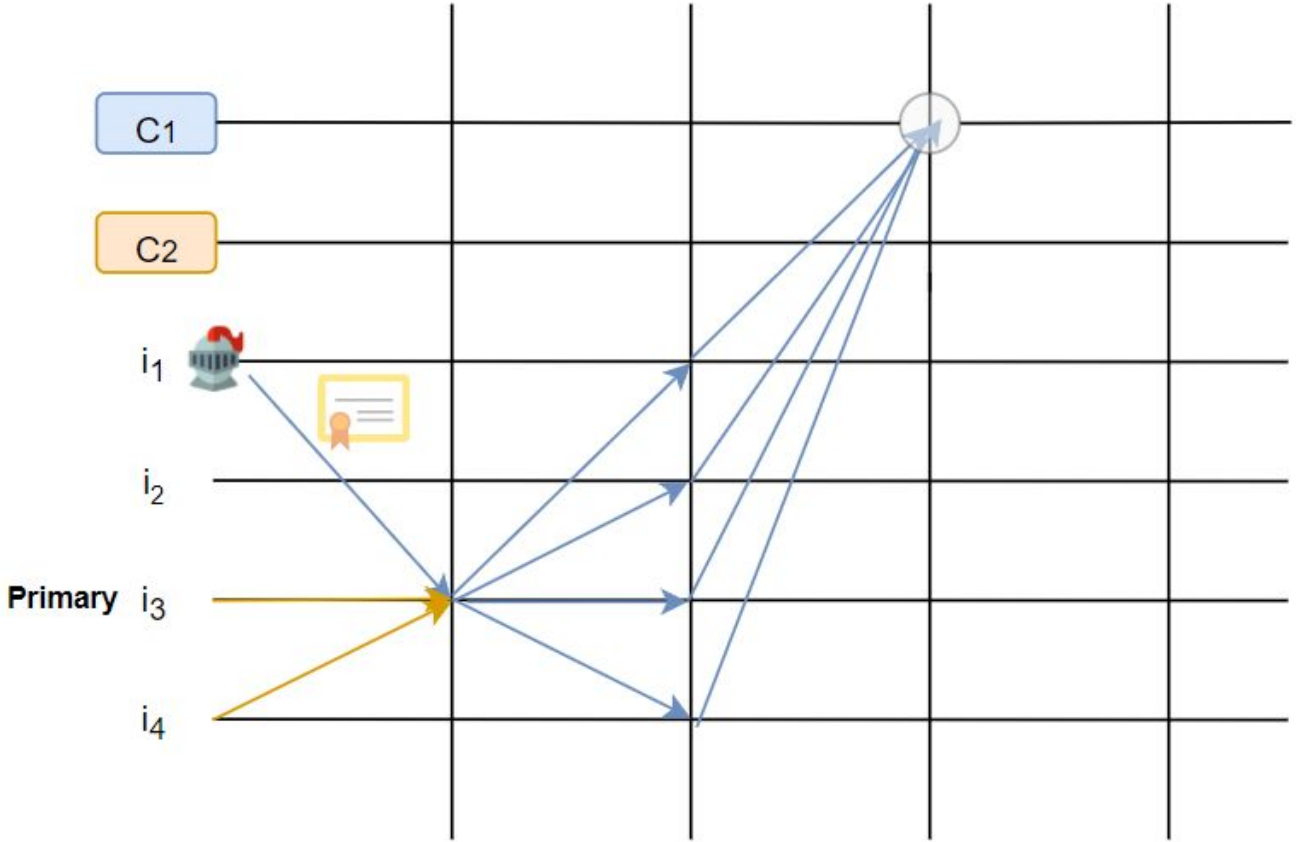


Zyzzyva



Zyzzyva

View 3: Choosing the wrong commit-certificate

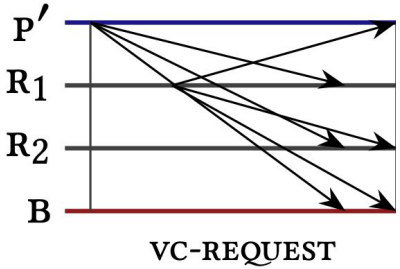


Proof of Execution: View Change

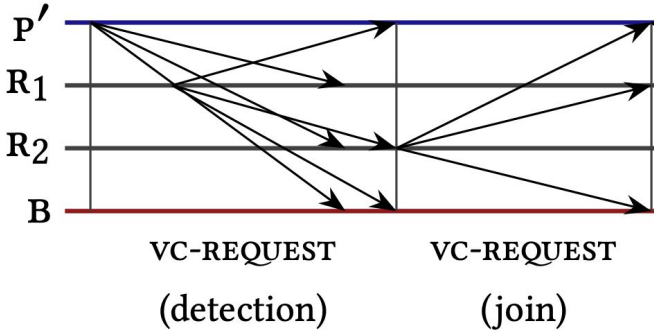
Proof of Execution - View Change

- Replicas elect new primary
- Needs to be detected by all non faulty replicas
- All replicas establish which transactions were included in view v
- A new primary proposes a new view.
  - If valid, the replicas switch over
  - If not, view change is initiated again

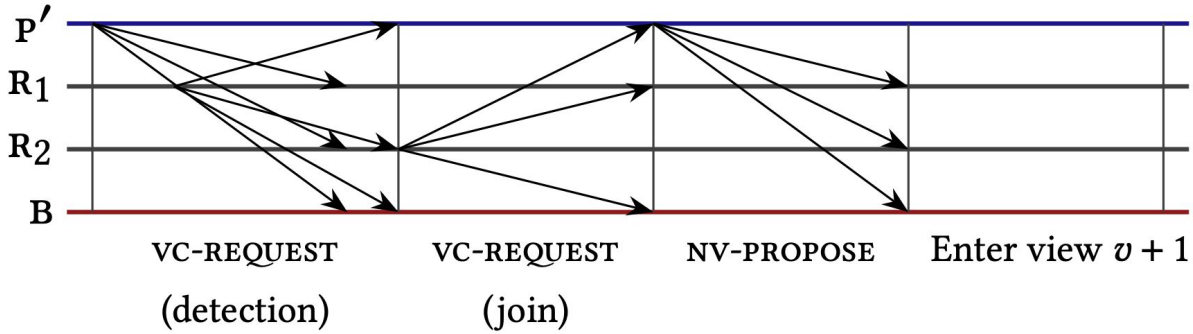
POE



POE



POE



Discussion

Zyzzyva Uses Speculation

- Reduce cost of BFT
- Simplify design of BFT

Zyzzyva Advantages over QU (Query Update)

1. **Fewer** replicas
2. Improved throughput via **batching**
3. **Simpler** State Machine Replication Semantics
4. Ability to support **high** contention **workloads**

Zyzzyva Disadvantages over QU

1. Will this actually **resolve** the problem?
2. What if there is an **extra** work?
3. Simpler does **not** mean it works **all** the **time**.

Q/U sidesteps this lower bound by providing a slightly weaker service than state machine replication and optimizes cases without concurrent access to any state

Zyzzyva vs PBFT

Zyzzyva reduces cryptographic overheads and increases peak throughout by a factor of two to an order of magnitude for demanding workloads.

Discussion

Proof Of Execution

- Reduce cost of BFT
- Simplify design of BFT

POE Advantages

1. **Fewer Replicas**
2. **Out of order** processing
3. **Speculative Execution**
4. No **Response Aggregation**

POE vs PBFT

POE reduces the amount of communication between replicas and increases throughput across multiple factors

Zyzzyva Evaluation

Evaluation

The protocol was evaluated on the following:

1. **Throughput**
2. **Latency**
3. **Fault Scalability**
4. **Performance During Failures**

Protocols to be compared

Query/Update (Q/U)

Hybrid-Quorum replication (HQ Replication)

Practical Byzantine Fault Tolerance (PBFT)

Zyzzyva

Throughput

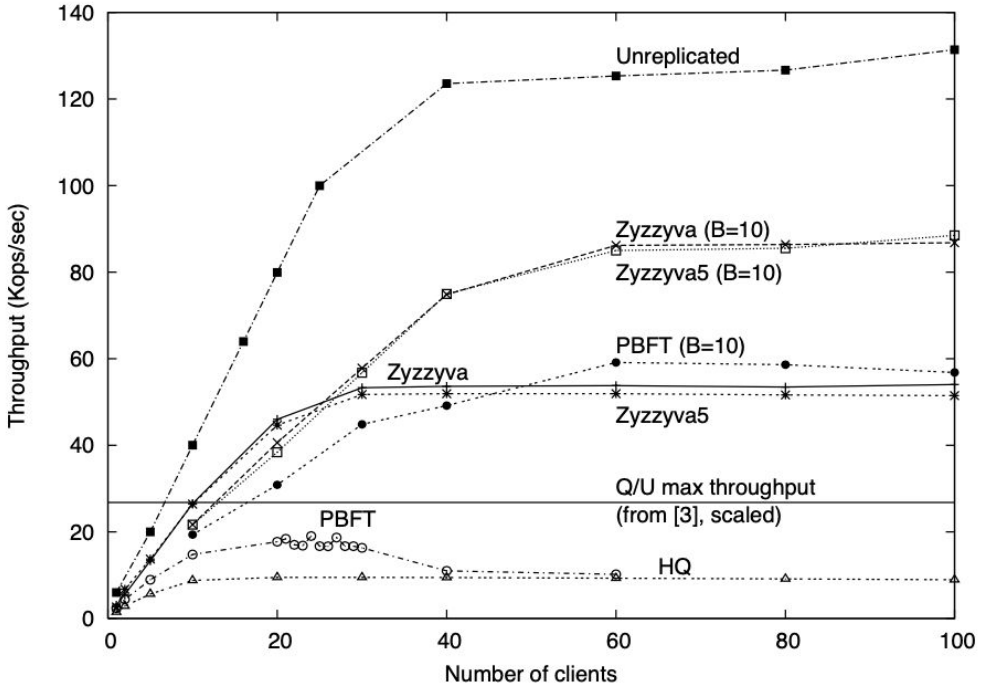


Figure 3: Realized throughput for the 0/0 benchmark as the number of client varies for systems configured to tolerate $f = 1$ faults.

Latency

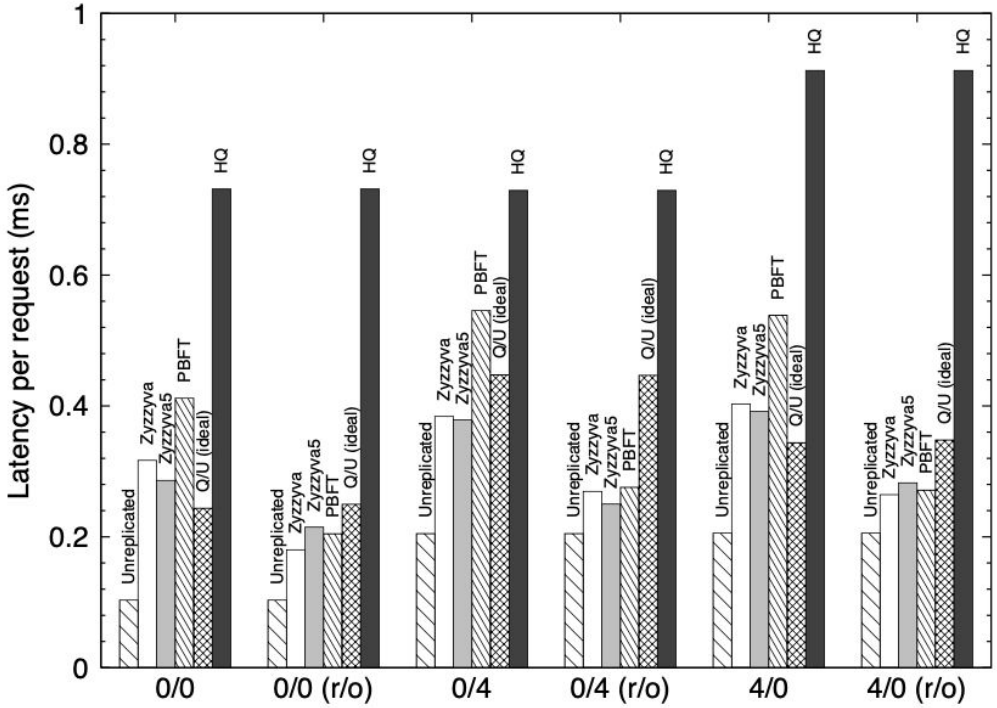


Figure 4: Latency for 0/0, 0/4, and 4/0 benchmarks for systems configured to tolerate $f = 1$ faults.

Fault Scalability

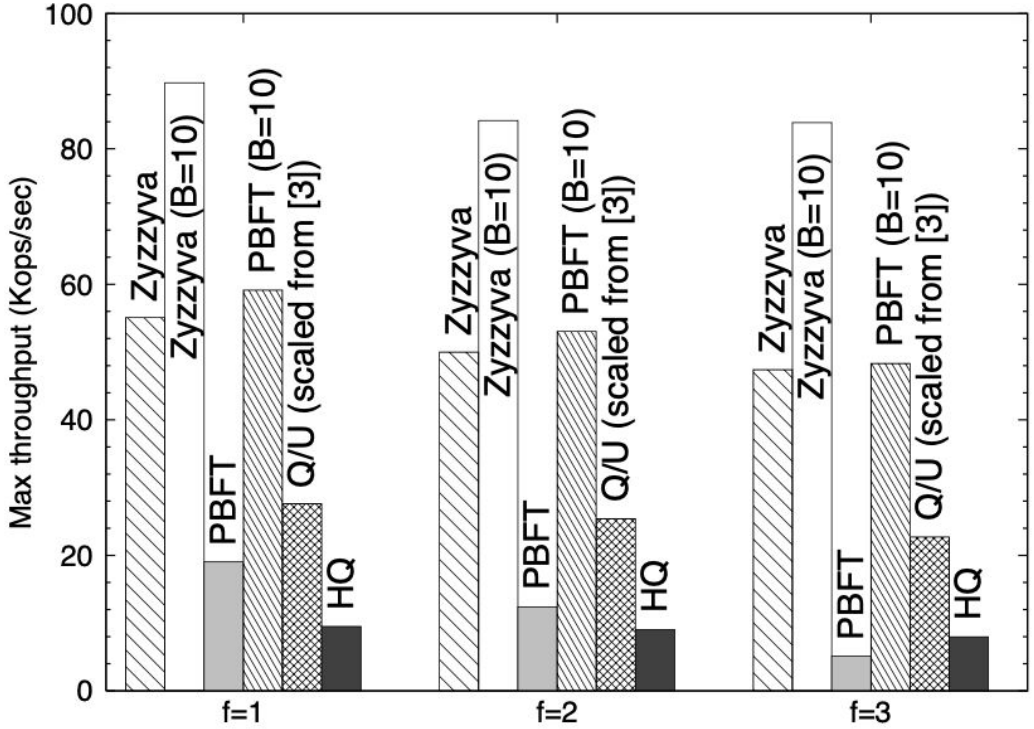


Figure 6: Fault scalability: Peak throughputs

Fault Scalability

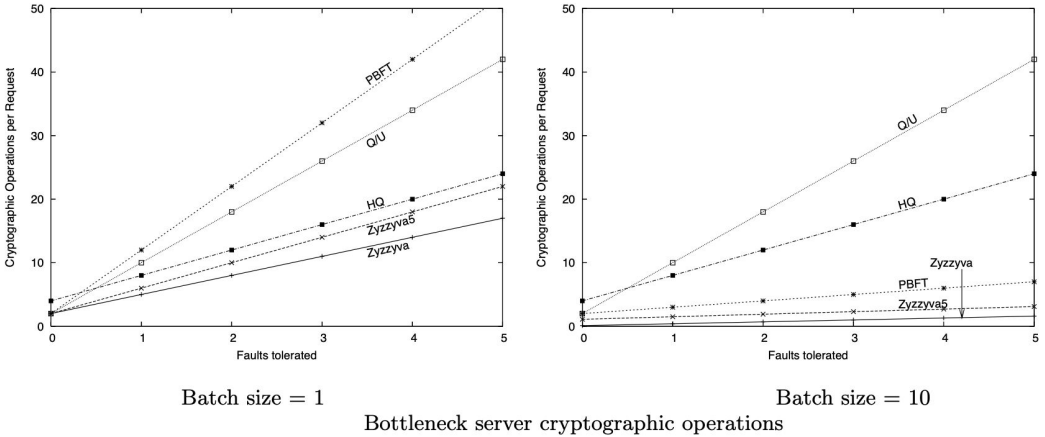


Figure 7: Fault scalability using analytical model

Performance During Failure

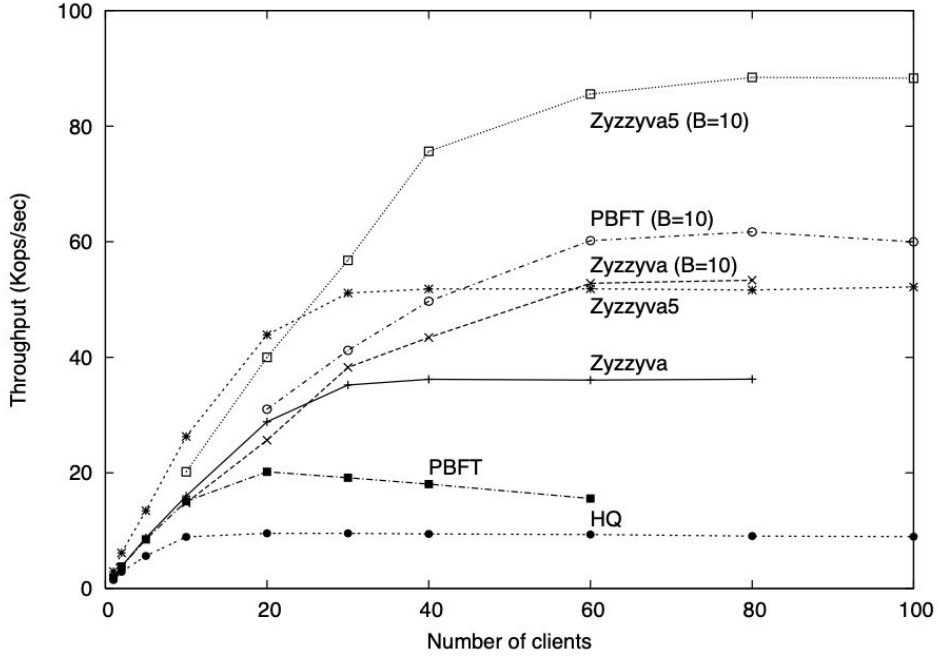


Figure 8: Realized throughput for the 0/0 benchmark as the number of client varies when f non-primary replicas fail to respond to requests.

Zyzzyva Implementation Optimizations

Implementation Optimizations

The protocol has 7 optimizations that were applied:

1. **Replacing Signatures with MACS**
2. **Separating Agreement from Execution**
3. **Request Batching**
4. **Caching Out of Order Requests**
5. **Read-Only Optimization**
6. **Single Execution Response**
7. **Preferred Quorums**

Proof of Execution: Evaluation

Evaluation

POE was evaluated against the following criteria

1. **Performance under failures**
2. **Benefit to batching client requests**
3. **Performance under zero payload**
4. **Scalability**

Protocols to be compared

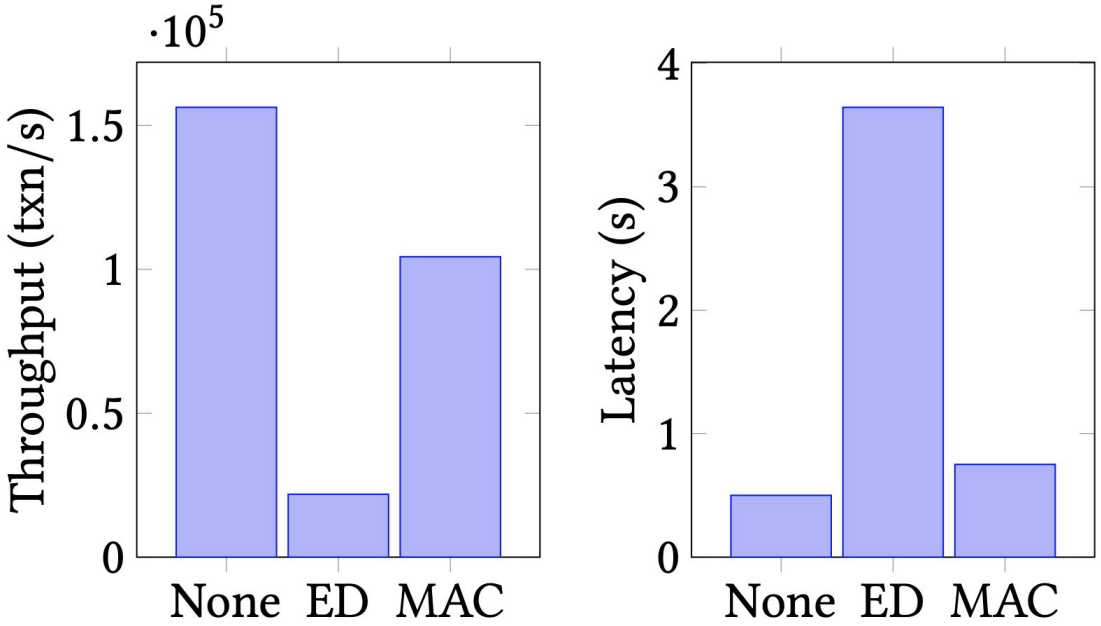
Practical Byzantine Fault Tolerance (PBFT)

Zyzzyva

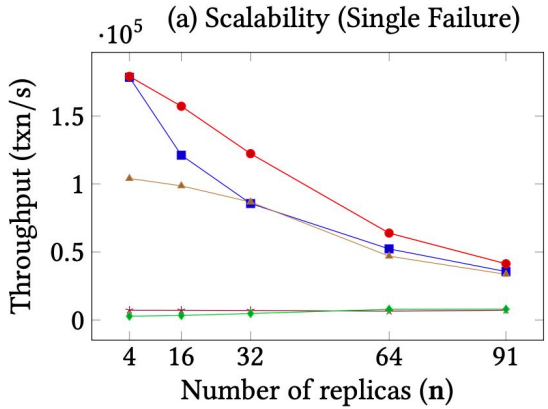
SBFT

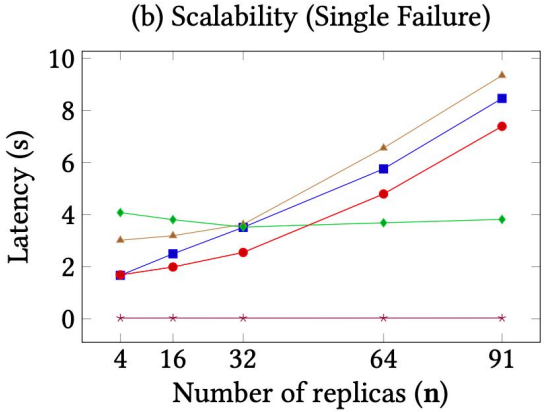
HotStuff

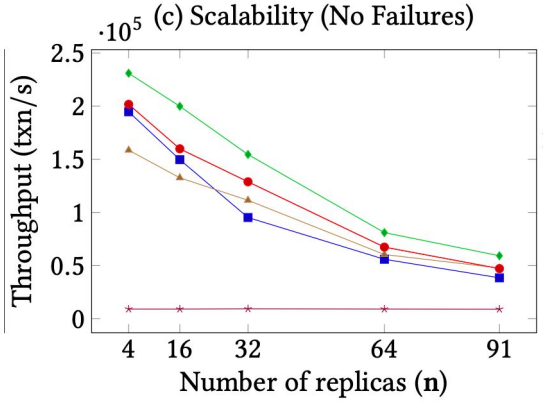
Evaluation - Performance with different signature schemes

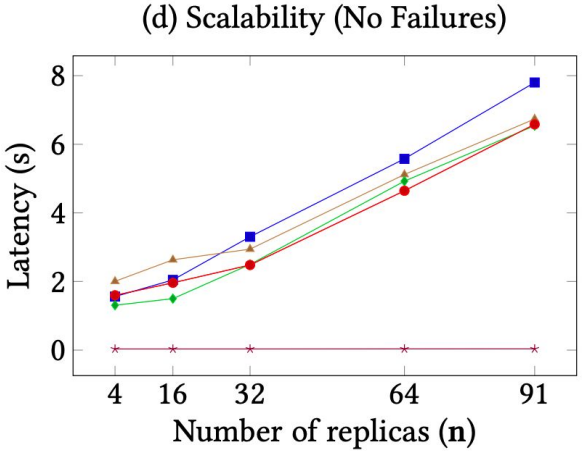


Evaluation - Scalability

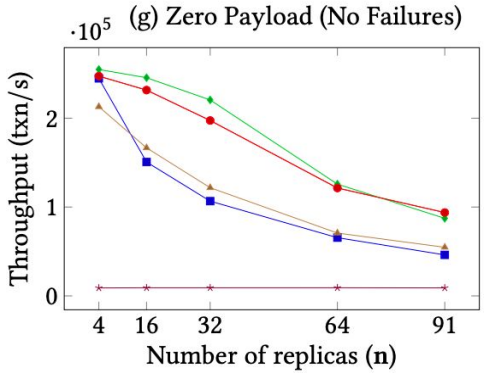


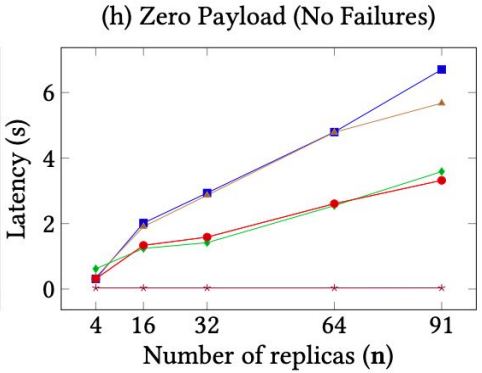


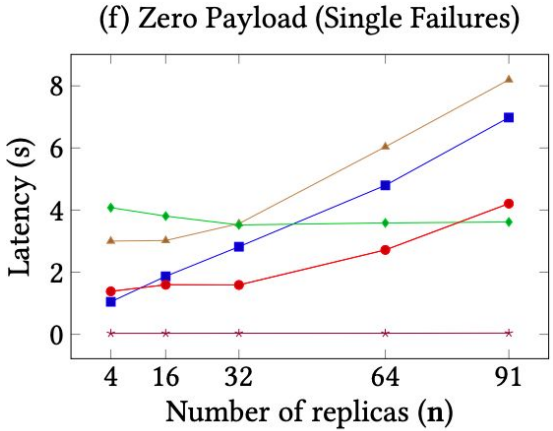


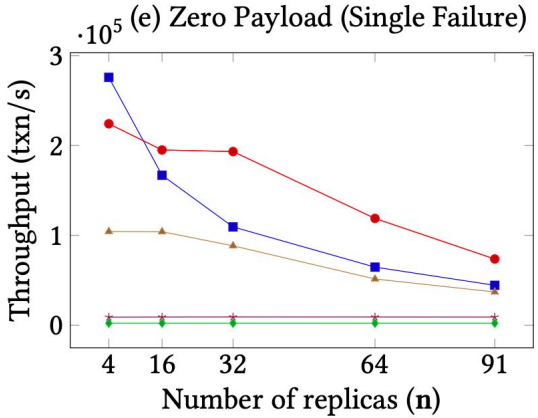


Evaluation - Zero Payload

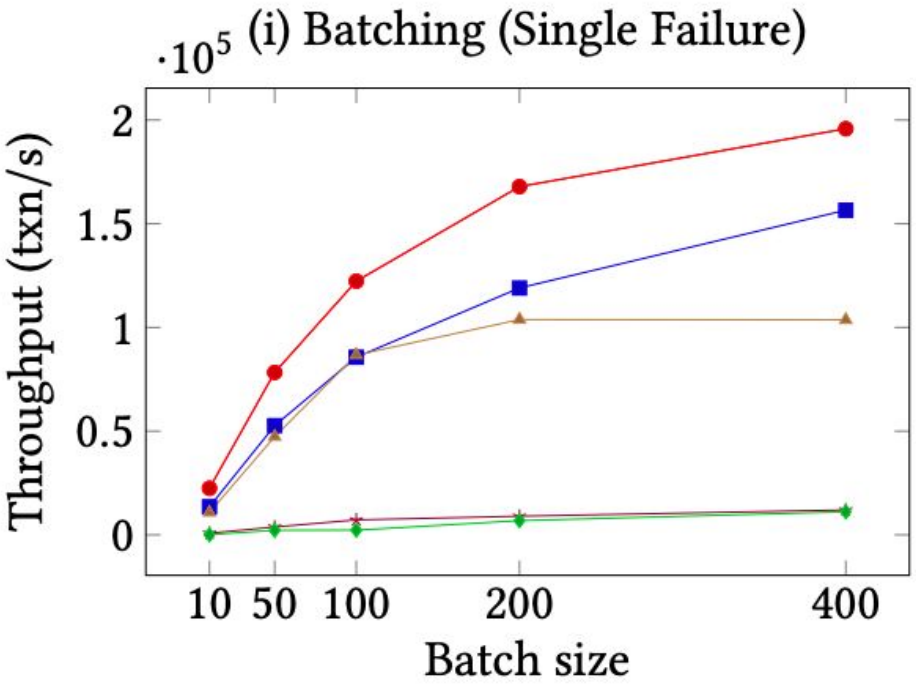


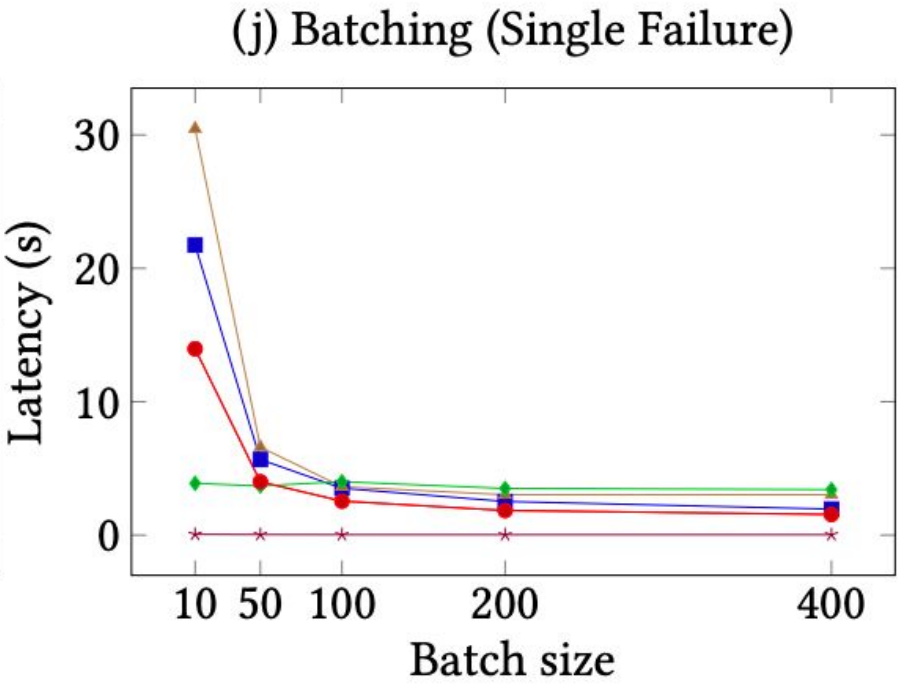




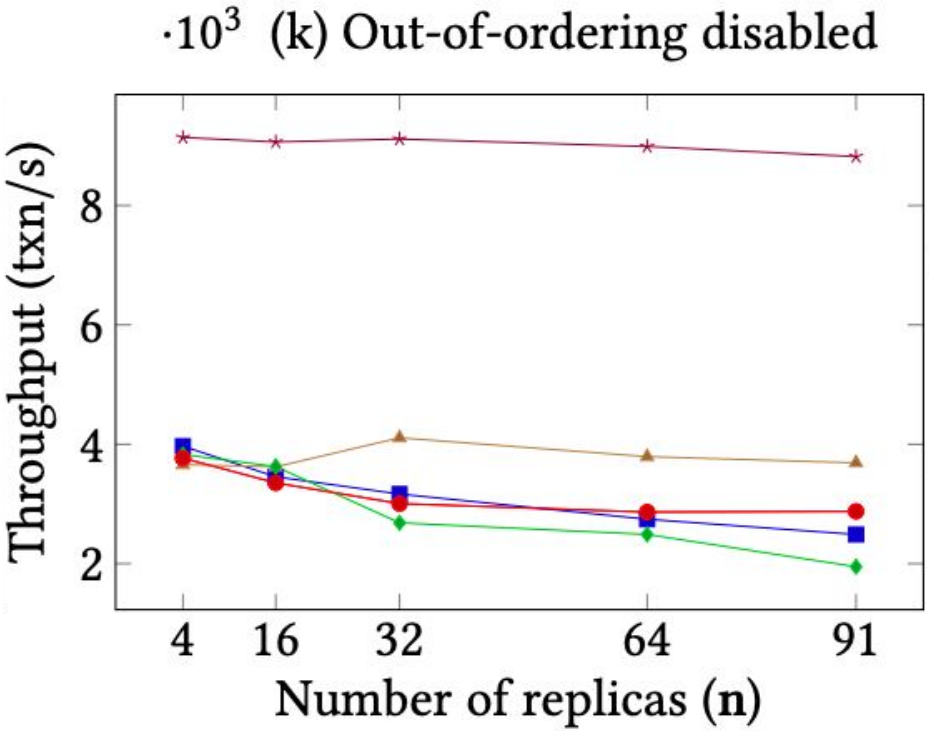


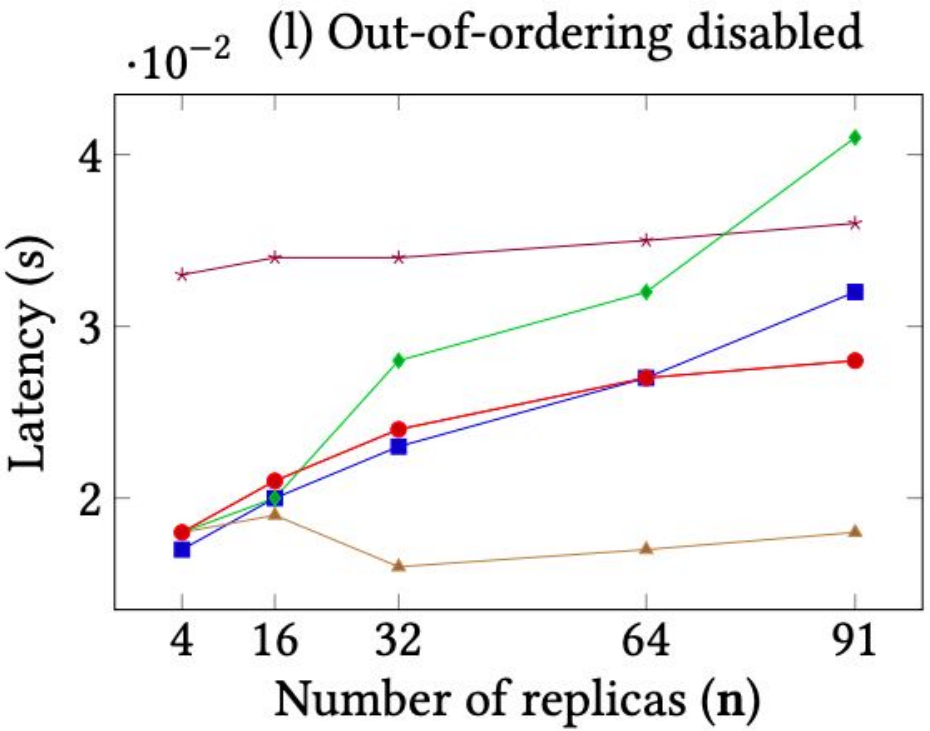
Evaluation - Batching





Evaluation - Out of Order Execution





Evaluation - Failures