Zyzzyva: Speculative Byzantine Fault Tolerance

Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong

Presented by Madelyn Gatchel

EECS 591 Fall 2021

Byzantine Fault Tolerance (BFT) Replication

System Model

- Asynchronous system
- Unreliable channels

Crypto

- Public/private key pairs
- Signatures
- Collision-resistant hashes

Service

- Byzantine clients
- Up to f Byzantine servers
- $n > 3f$ total servers

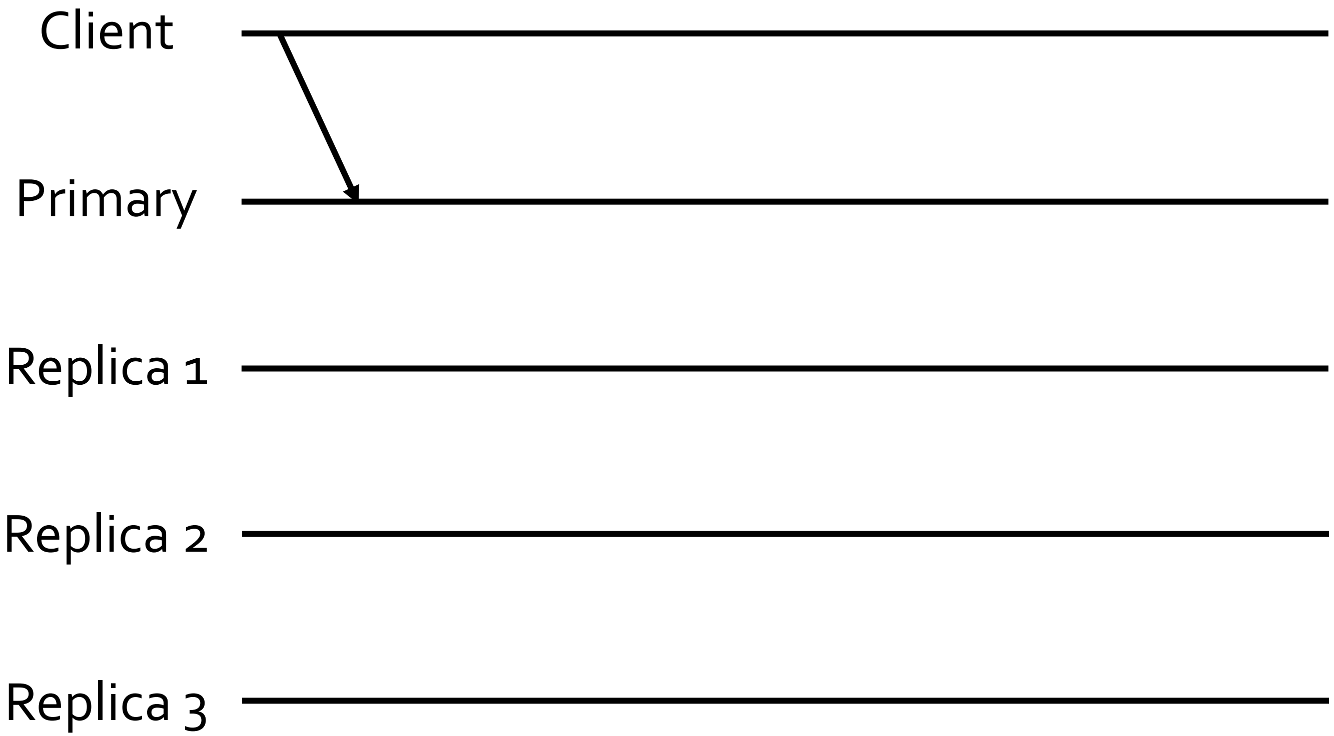
System Goals

- Always safe
- Live during periods of synchrony

[Adapted from EECS 591 Lecture 12 Slide 20]

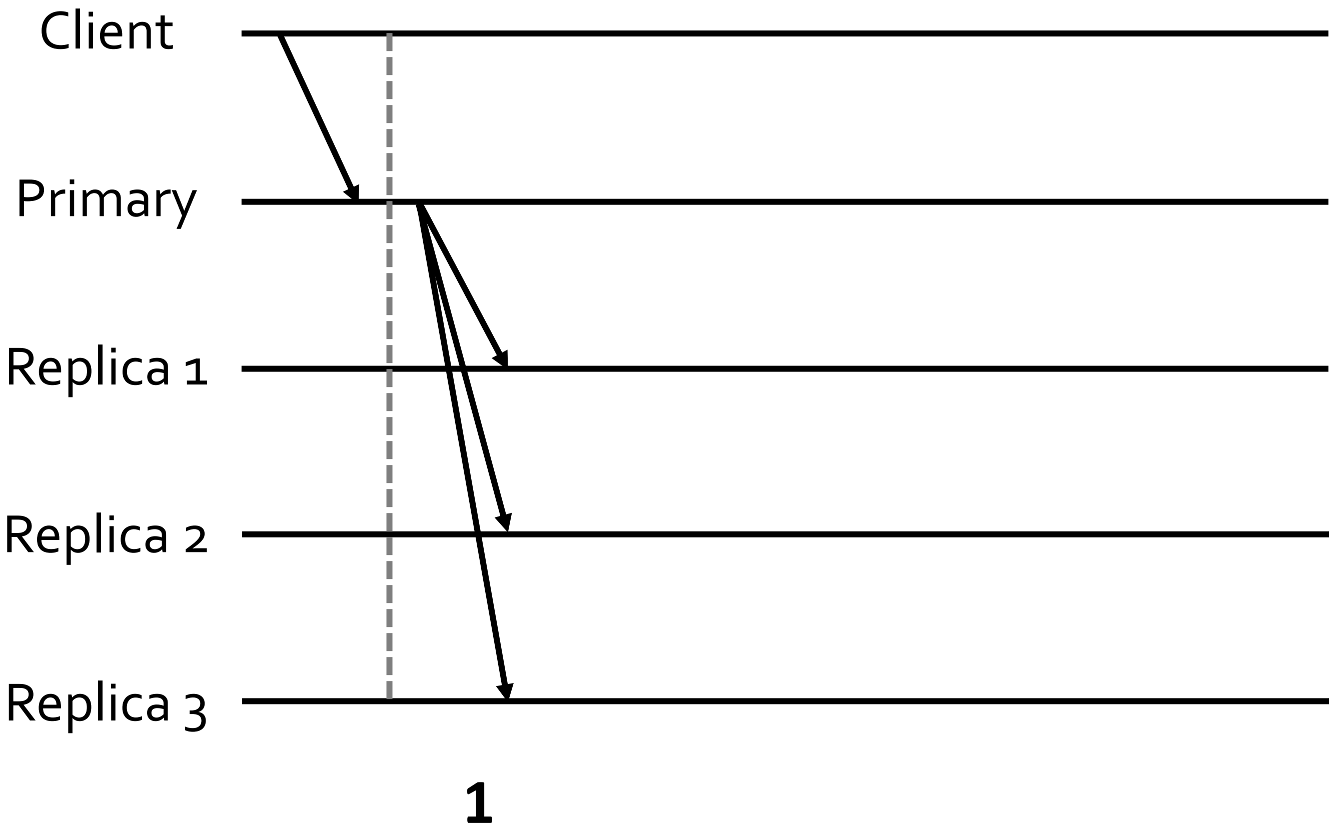
PBFT Review

PBFT Review



- Client sends message to primary

PBFT Review

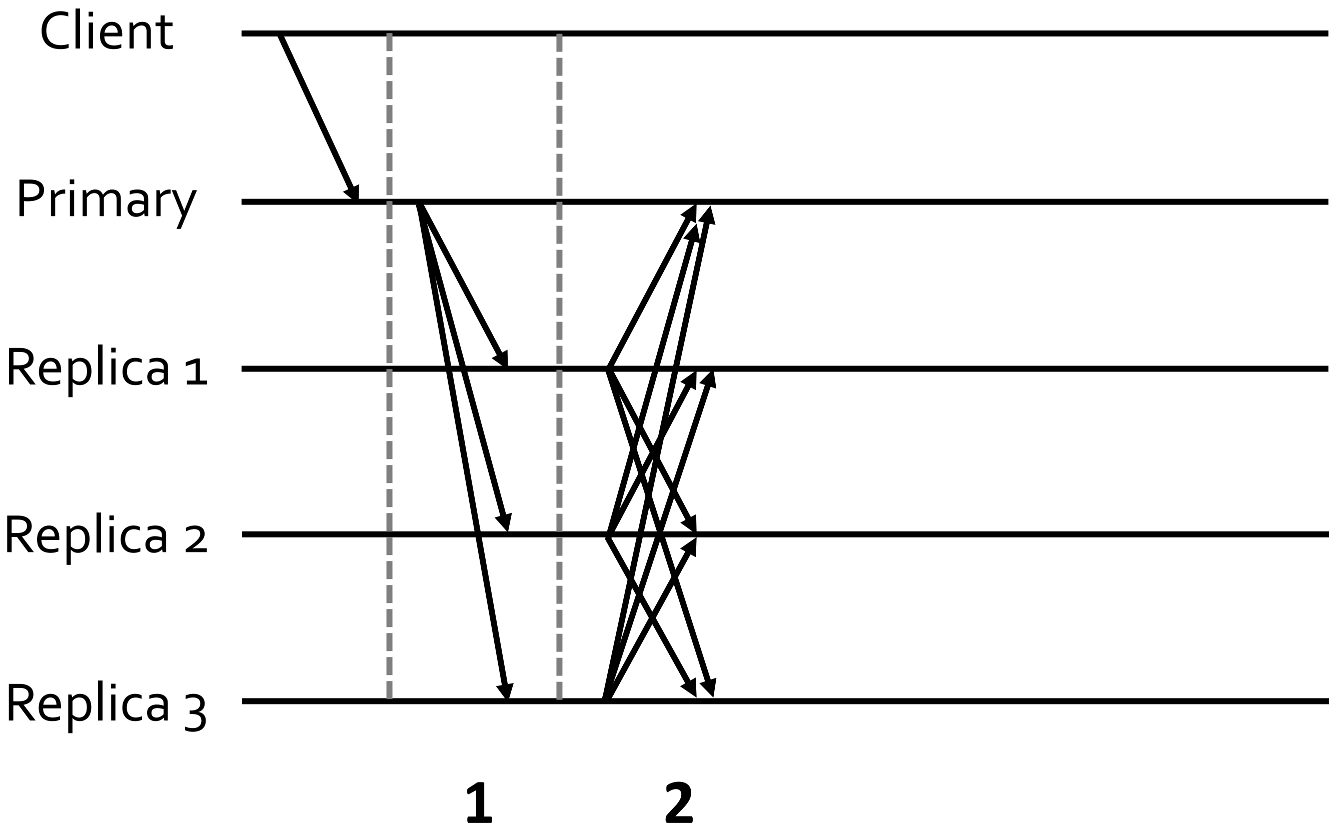


- Client sends message to primary

Three-phase commit:

1. Pre-prepare

PBFT Review

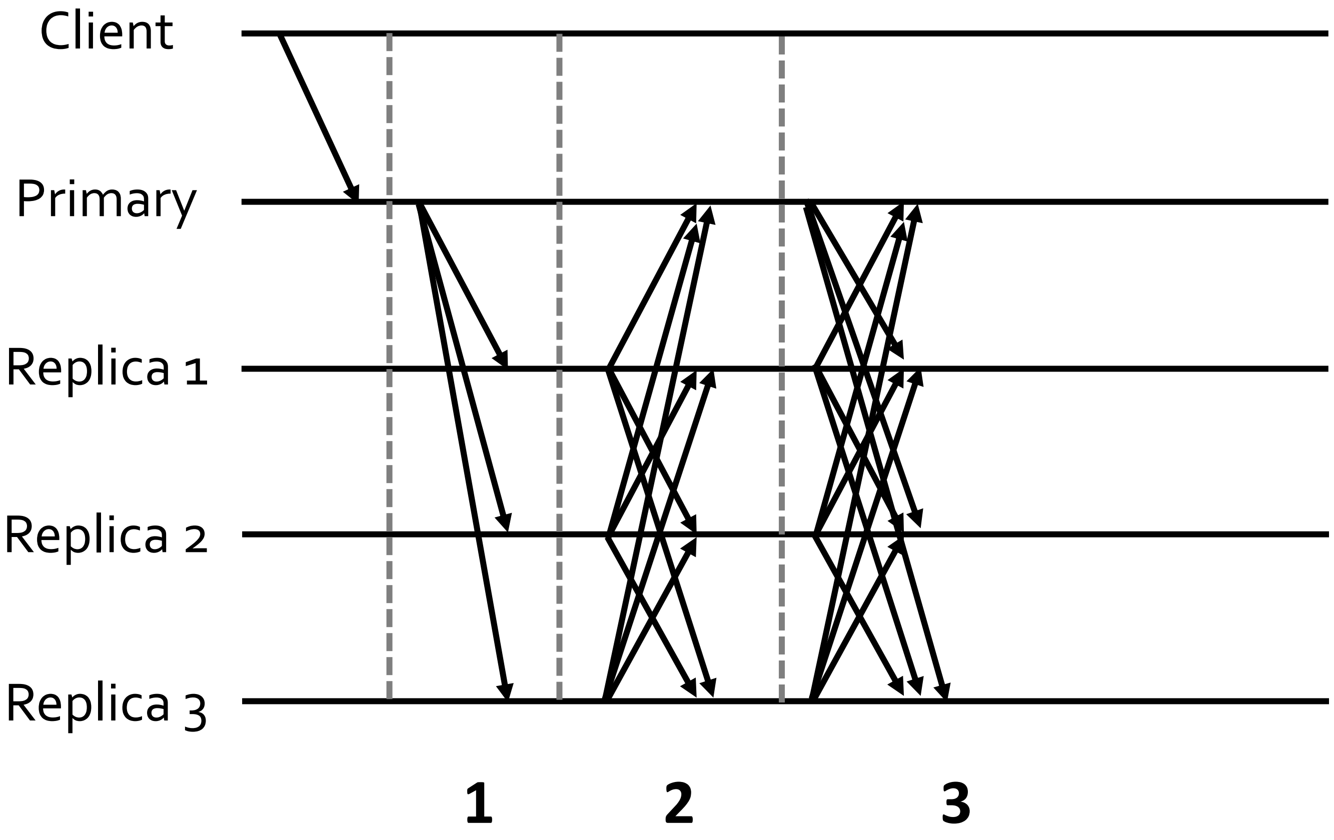


- Client sends message to primary

Three-phase commit:

1. Pre-prepare
2. Prepare

PBFT Review

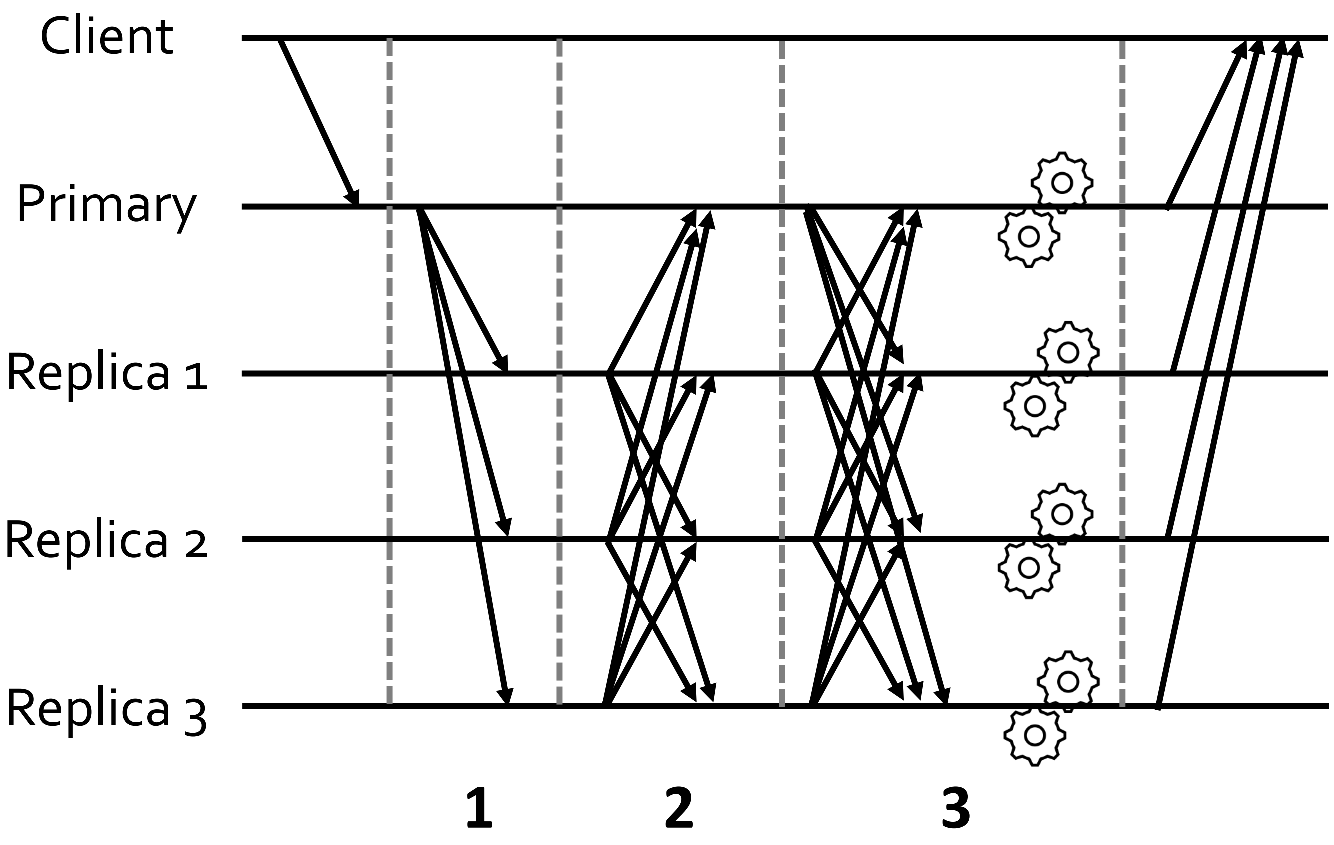


- Client sends message to primary

<u>Three-phase commit</u>:

1. Pre-prepare
2. Prepare
3. Commit

PBFT Review



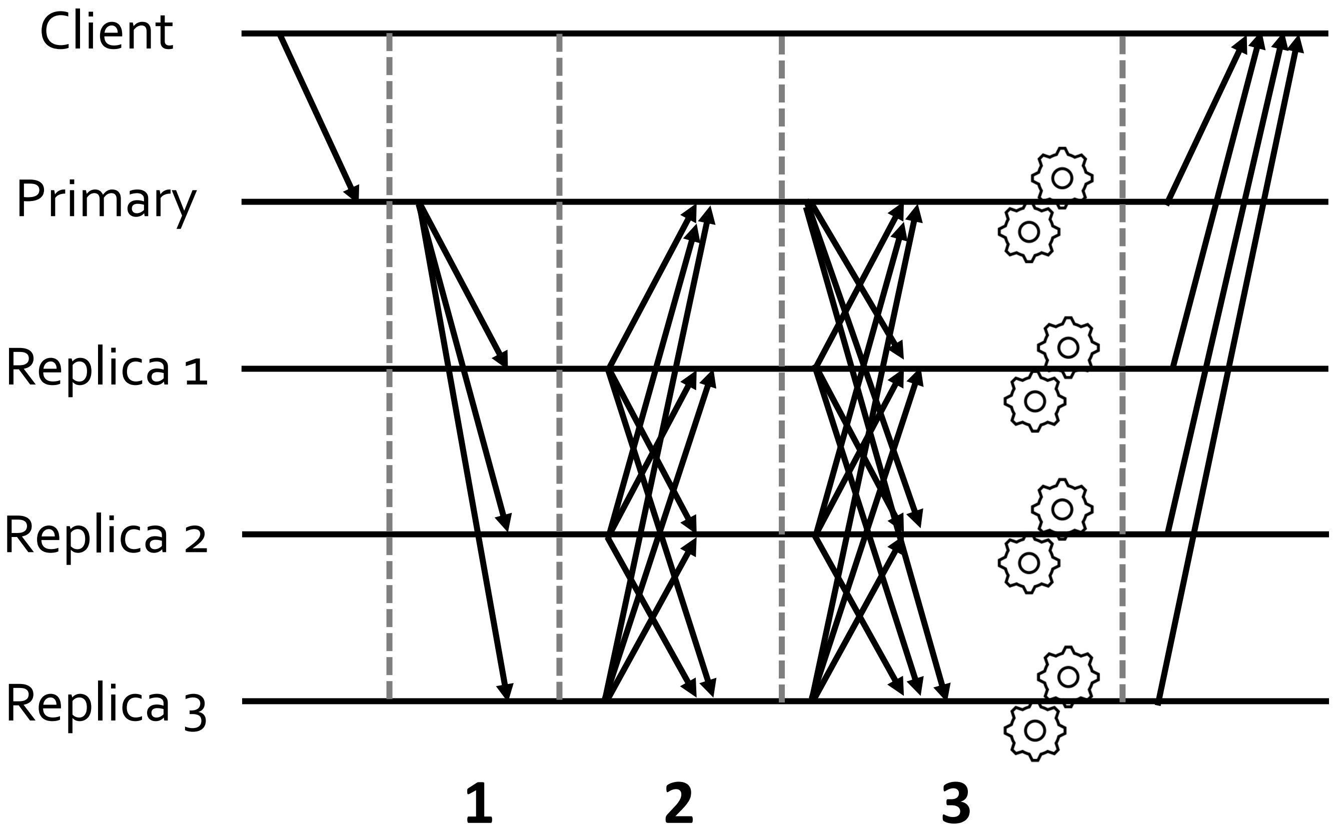
- Client sends message to primary

Three-phase commit:

1. Pre-prepare
2. Prepare
3. Commit

- Replicas execute and send reply to client

PBFT Review



The three-phase commit protocol is ***expensive.***

Introducing…Zyzzyva

- **Novel contribution:** replicas speculatively execute requests without 3-phase commit
- Correct replicas may be inconsistent
- Replicas may send different responses to clients
- Clients use history and replies to detect inconsistencies
- Clients wait until history and speculative reply are stable to complete request

Zyzzyva: tropical weevil and last word in dictionary

https://www.usatoday.com/story/news/nation-now/2017/06/27/zyzzyva-newest-last-word-oxford-english-dictionary-explained/431203001/

Why Zyzzyva?

- State-of-the-art BFT protocols
  - Practical Byzantine Fault Tolerance (PBFT) [Castro and Liskov, 1999]
  - Query/Update (Q/U) [Abd-El-Malek et al., 2005]
  - Hybrid-Quorum replication (HQ) [Cowling et al., 2006]
- HQ replication paper => best technique depends on workload
- How does Zyzzyva solve this issue?

BFT State-of-the-Art Comparison

**Gray/bold = best

		PBFT	Q/U	HQ	Zyzzyva
Cost	Total replicas Reps w/ app state	**3f+1** **2f+1**	5f+1 5f+1	**3f+1** 3f+1	**3f+1** **2f+1**
Throughput	MAC ops at bottleneck server	2+(8f+1)/b	2+8f	4+4f	**2+3f/b**
Latency	Critical path NW 1-way latencies	4	**2**	4	**3**

[Adapted from Table 1]

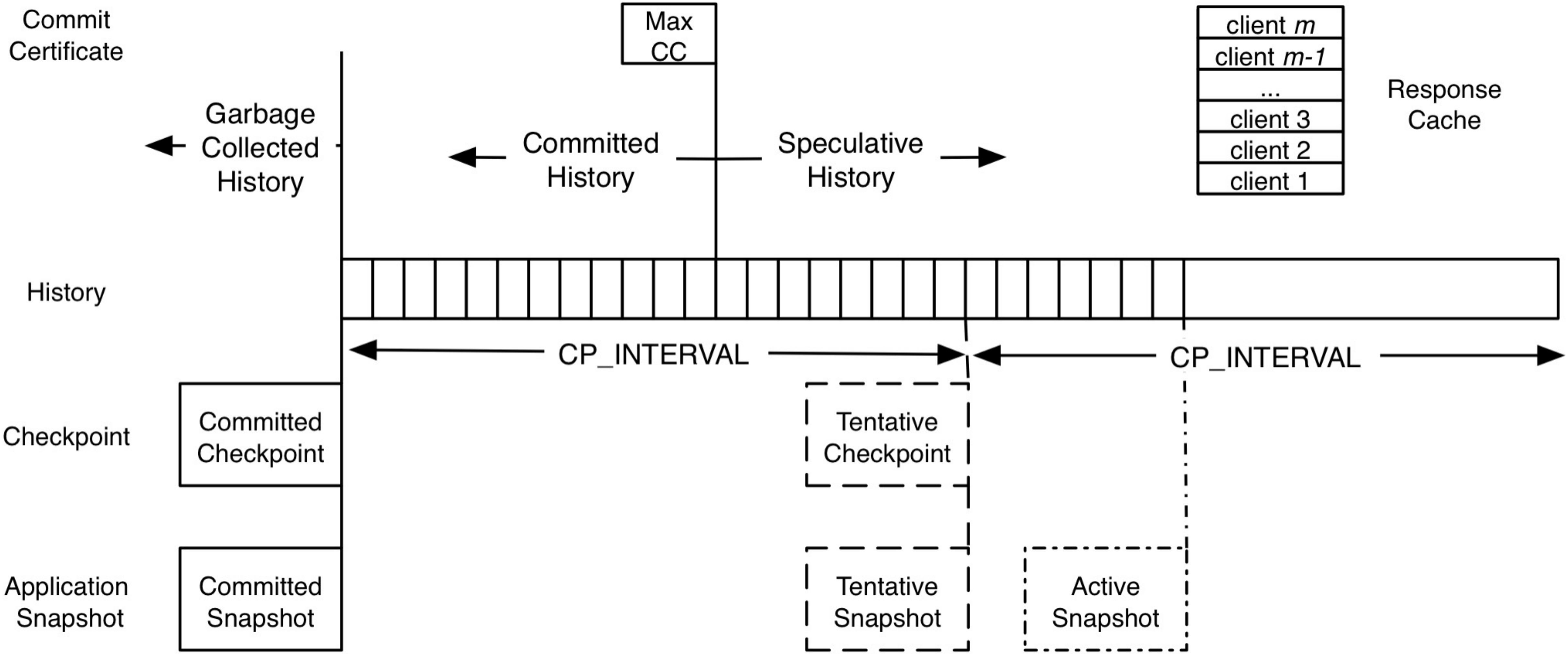
Zyzzyva Overview

- One primary, 3f replicas
- Execution proceeds as a sequence of views
- Design challenges
  - Conditions for client request completion
  - Defining subprotocols to ensure correctness
- Subprotocols:

- Agreement	Orders requests for replica execution
- Checkpoint	Limits state replicas must store and reduces cost of view changes
- View Change	Coordinates new primary election if current is faulty or system is running slowly

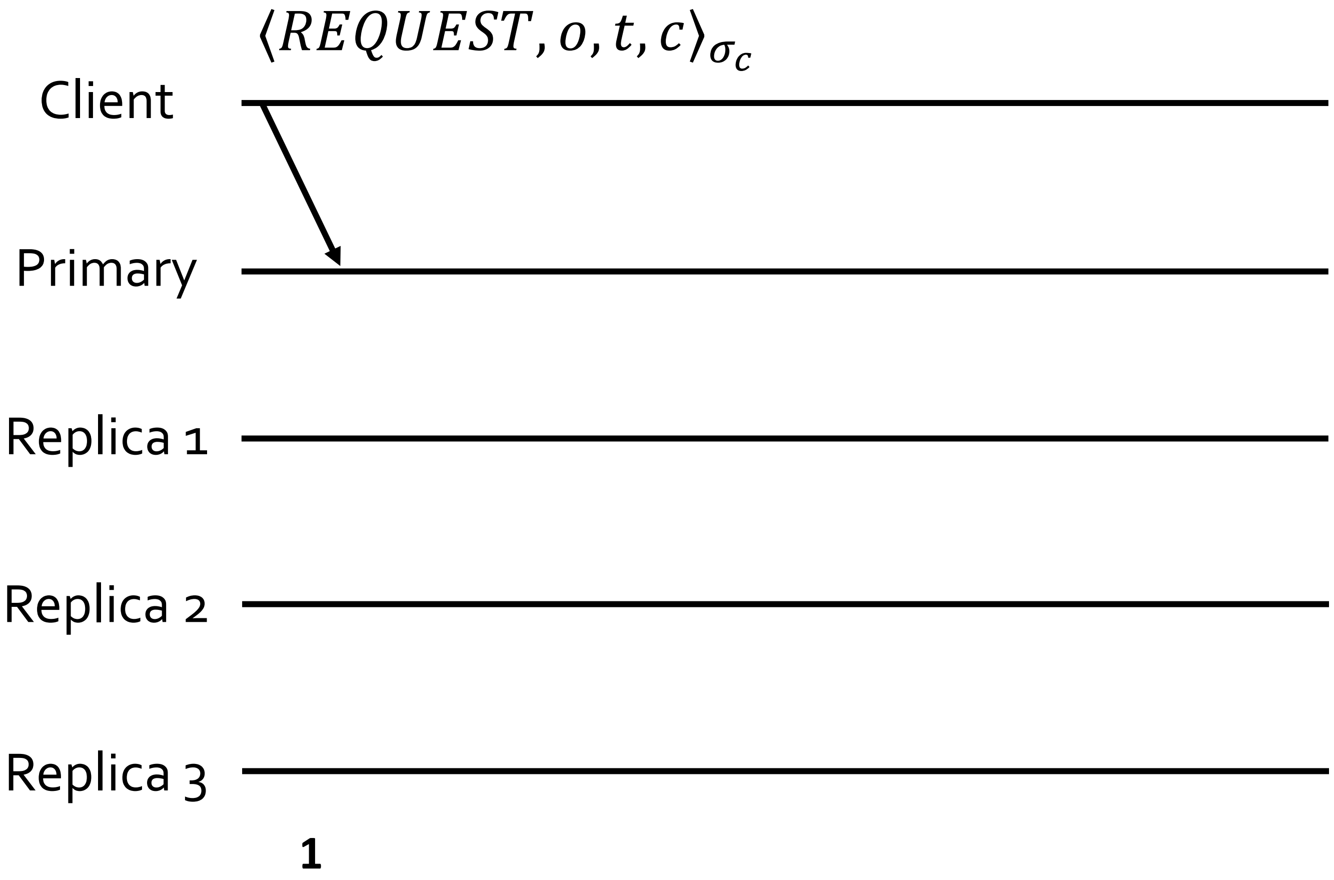
Node State & Checkpoint Subprotocol

Node State & Checkpoint Subprotocol



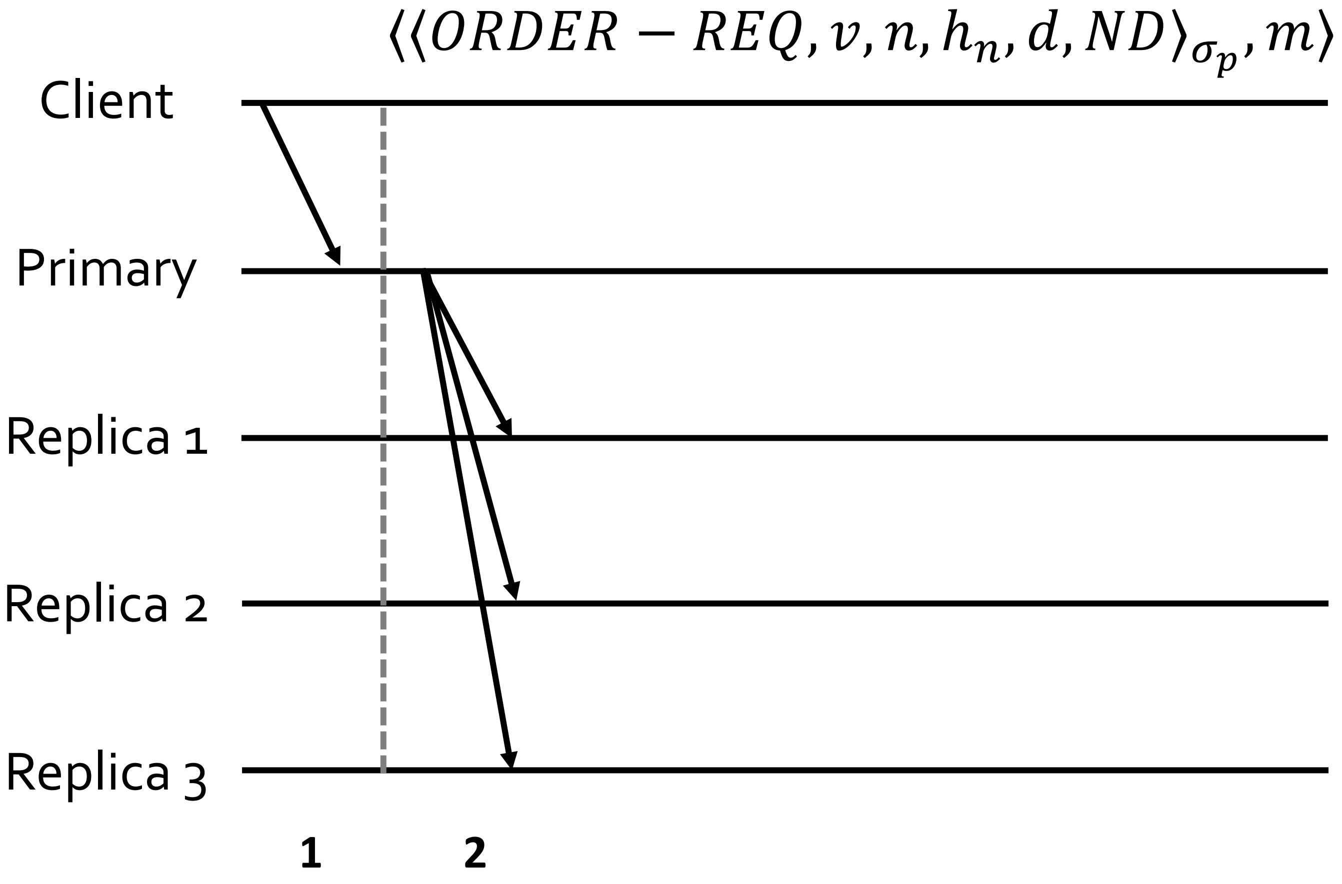
Agreement Subprotocol

Agreement Subprotocol



1. Client sends request to the primary.

Agreement Subprotocol



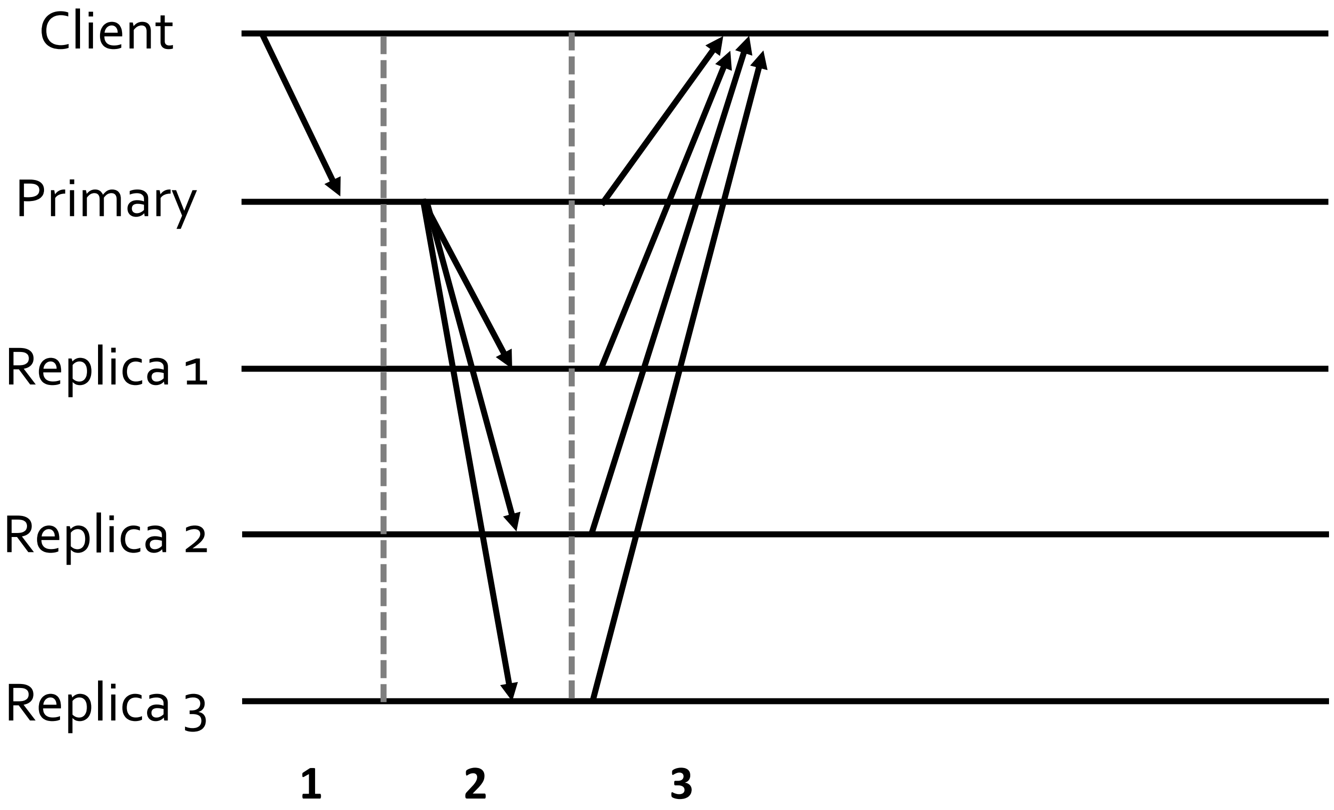
2.

- Primary receives request
- Assigns sequence number
- Forwards ordered request to replicas

Agreement Subprotocol

$OR = \langle ORDER-REQ, v, n, h_n, d, ND \rangle_{\sigma_p}$

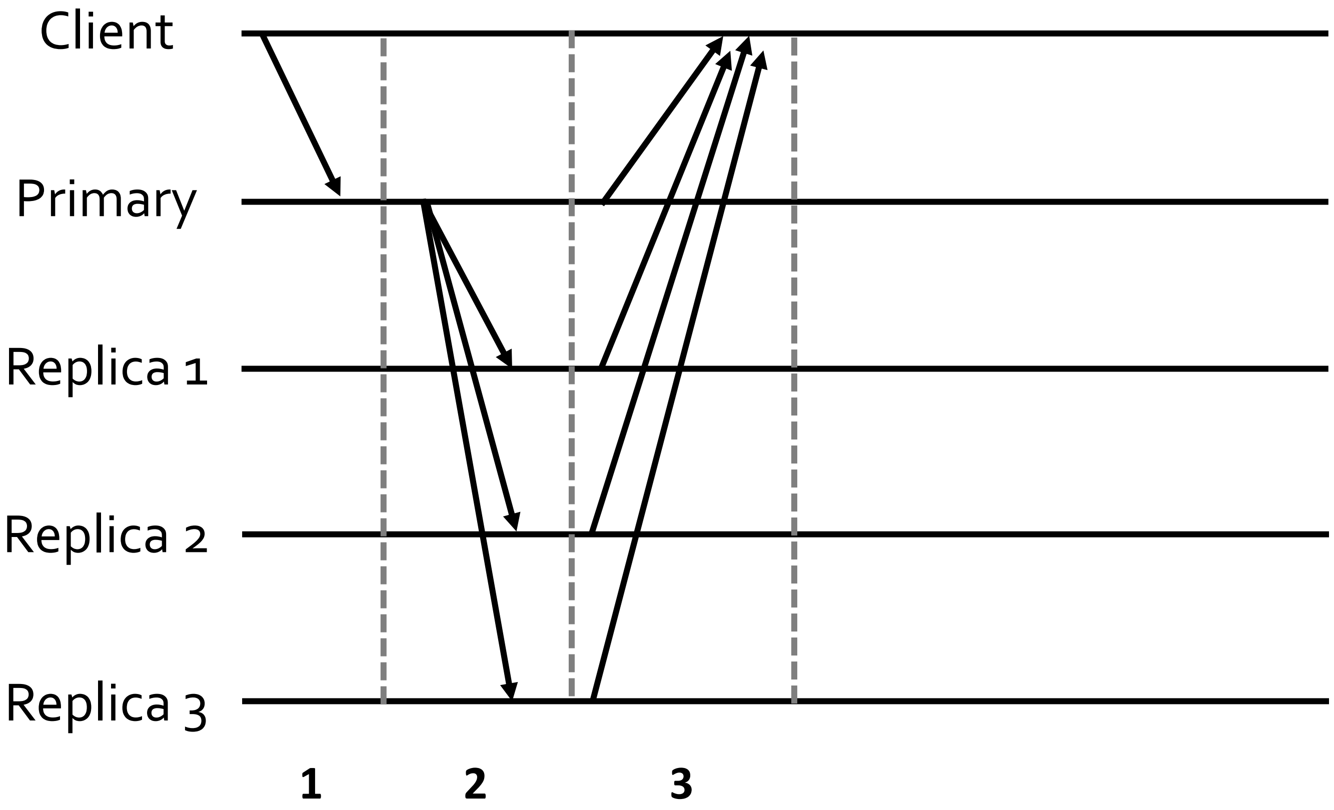
$\langle \langle SPEC-RESPONSE, v, n, h_n, H(r), c, t \rangle_{\sigma_i}, i, r, OR \rangle$



3.

- Replica receives ordered request
- Speculatively executes request
- Responds to the client

Agreement Subprotocol



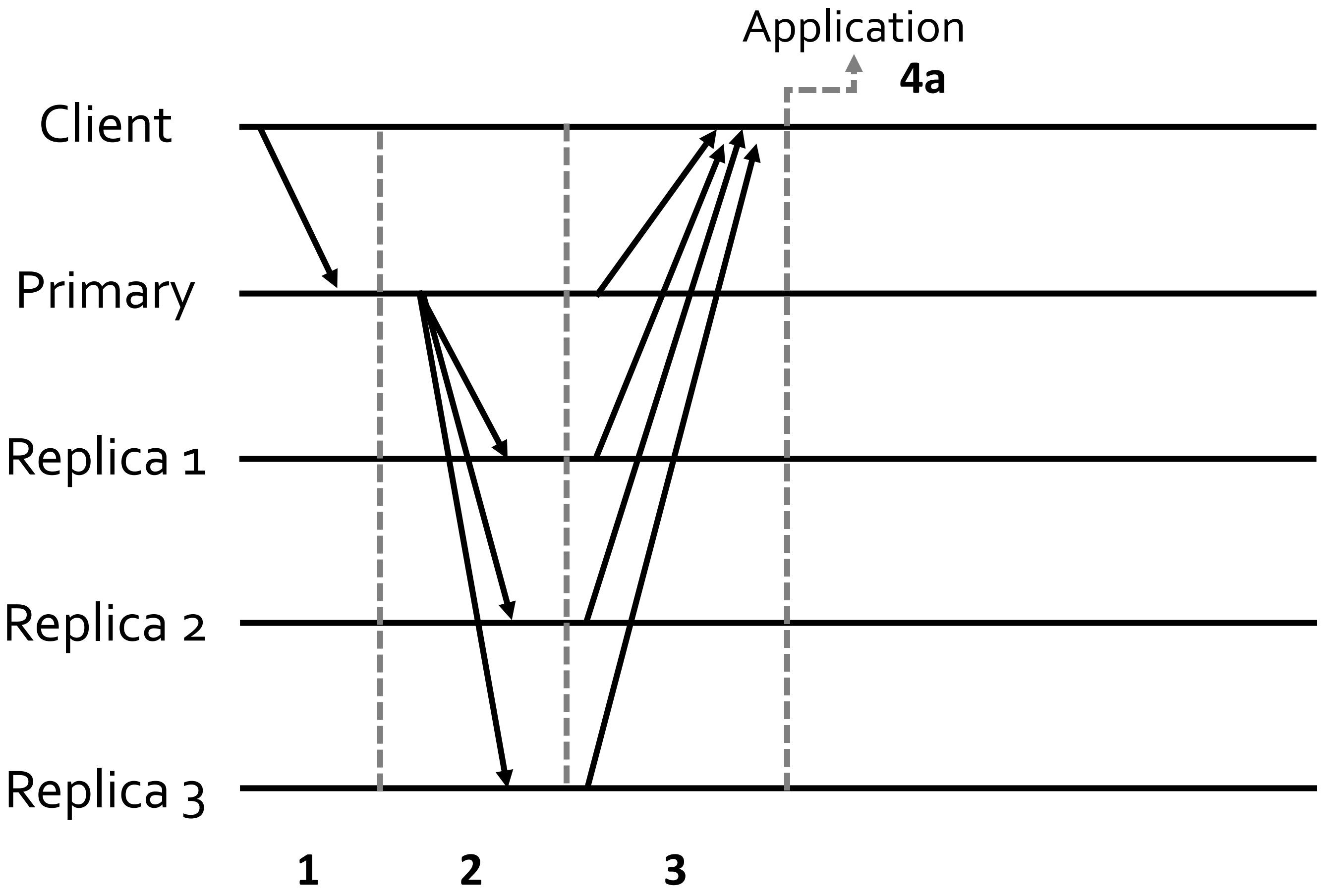
4. Client gathers speculative responses

Client Completion Summary

If client receives…

Exactly
$3f + 1$ speculative
response messages

Agreement Subprotocol

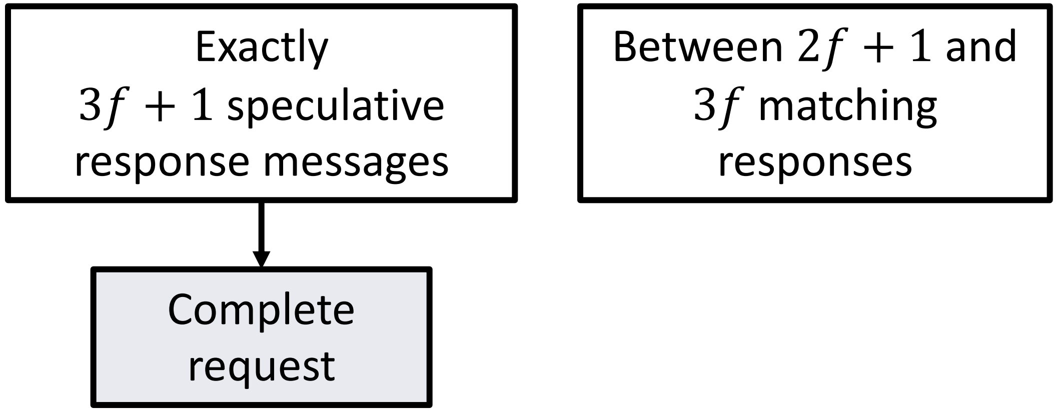


4a. If client receives **exactly $3f+1$ matching responses:**

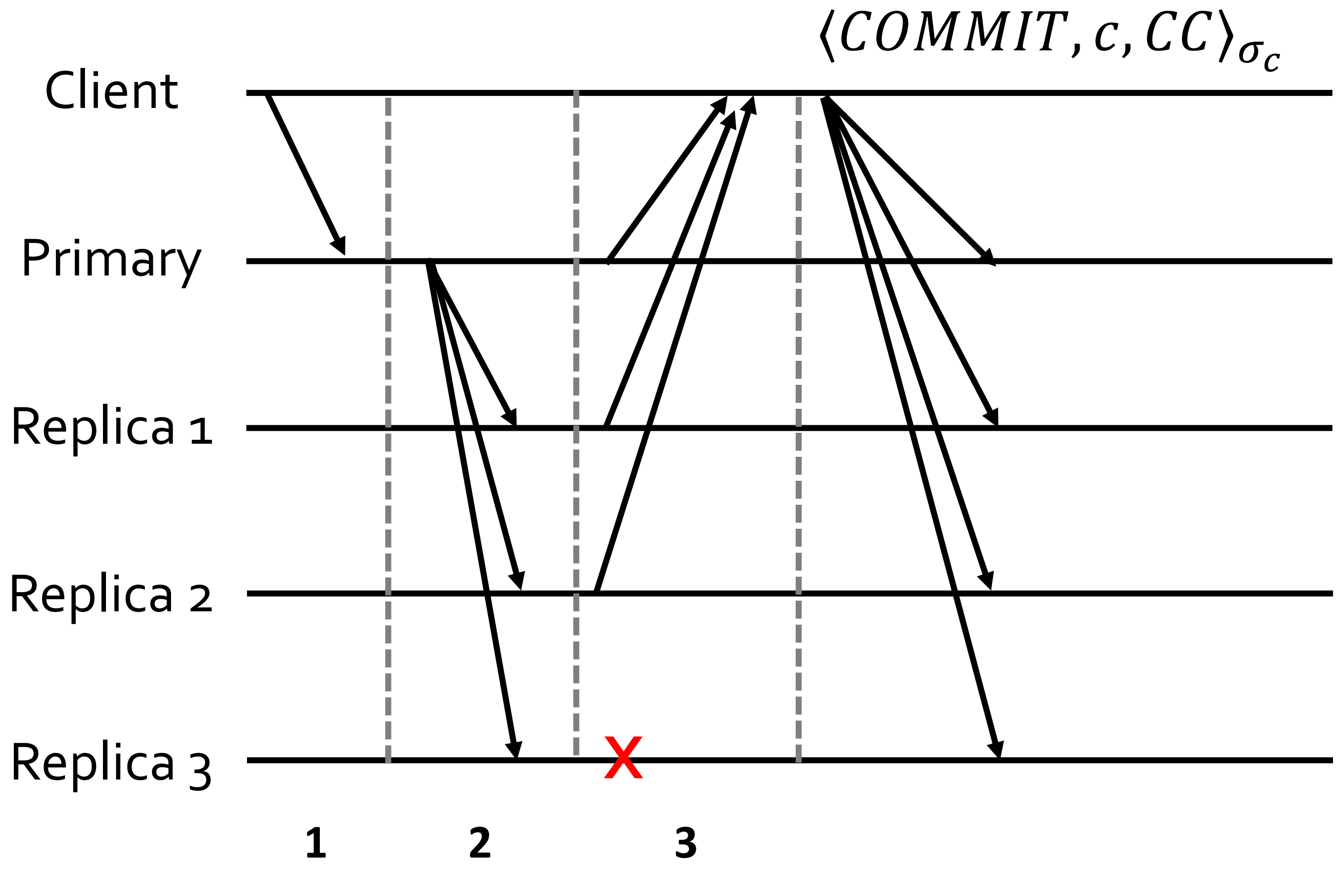
- Client completes the request.

Client Completion Summary

If client receives…



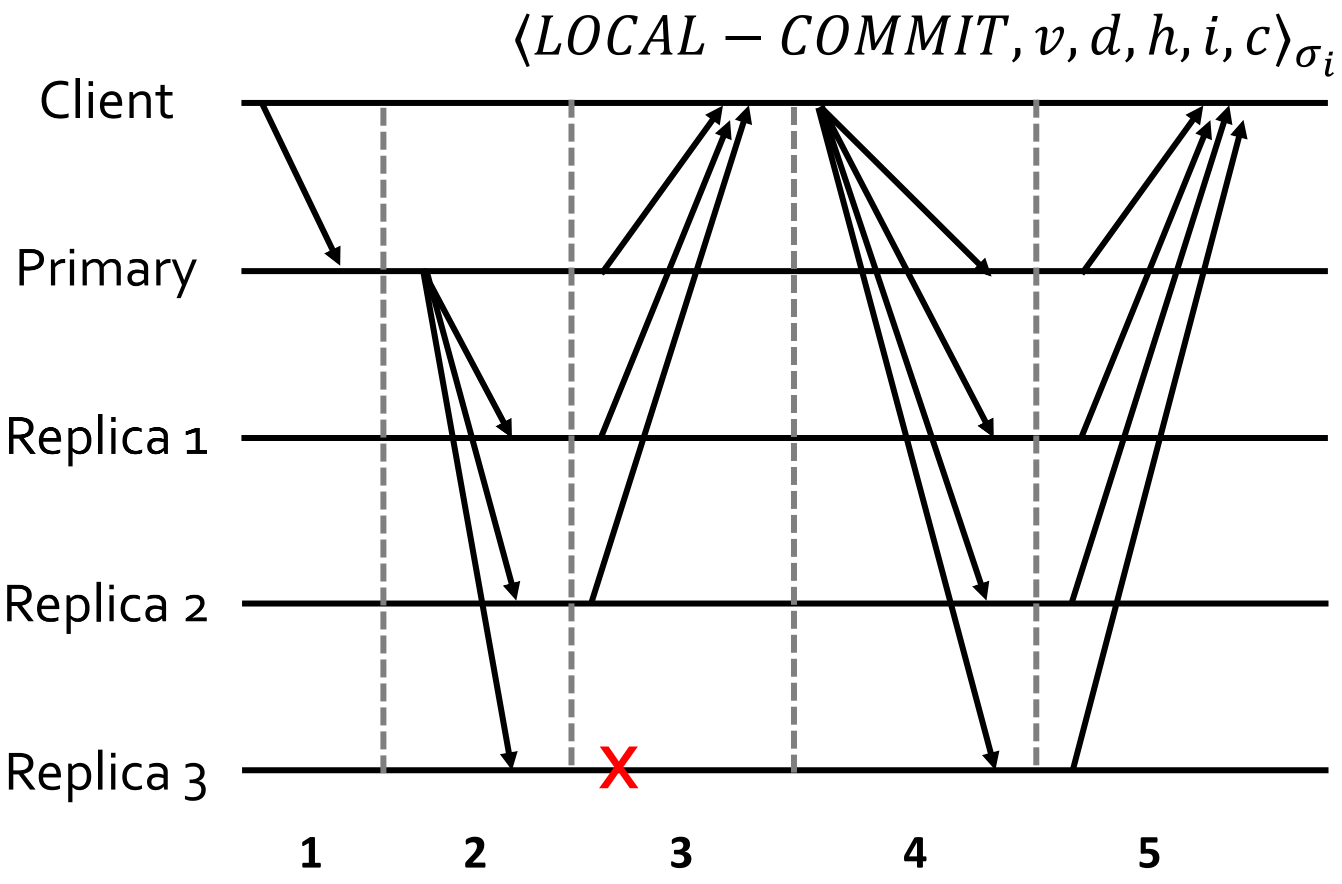
Agreement Subprotocol



4b. If client receives **between $2f + 1$ and $3f$ matching responses:**

- Client assembles a C-certificate
- Transmits it to the replicas

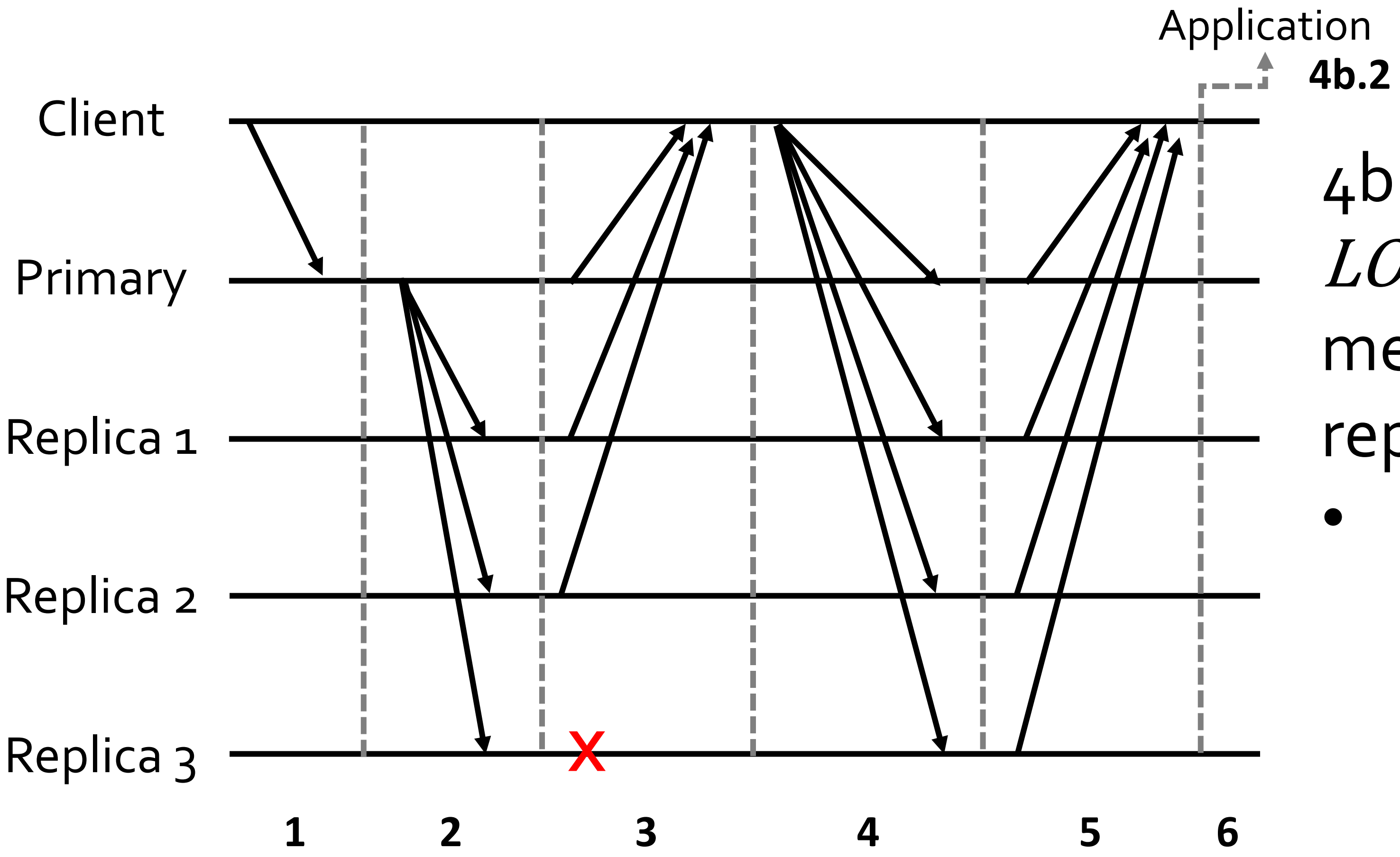
Agreement Subprotocol



4b.1.

- Replica receives a *COMMIT* message from a client containing a C-certificate
- Replica acknowledges with a *LOCAL-COMMIT* message.

Agreement Subprotocol

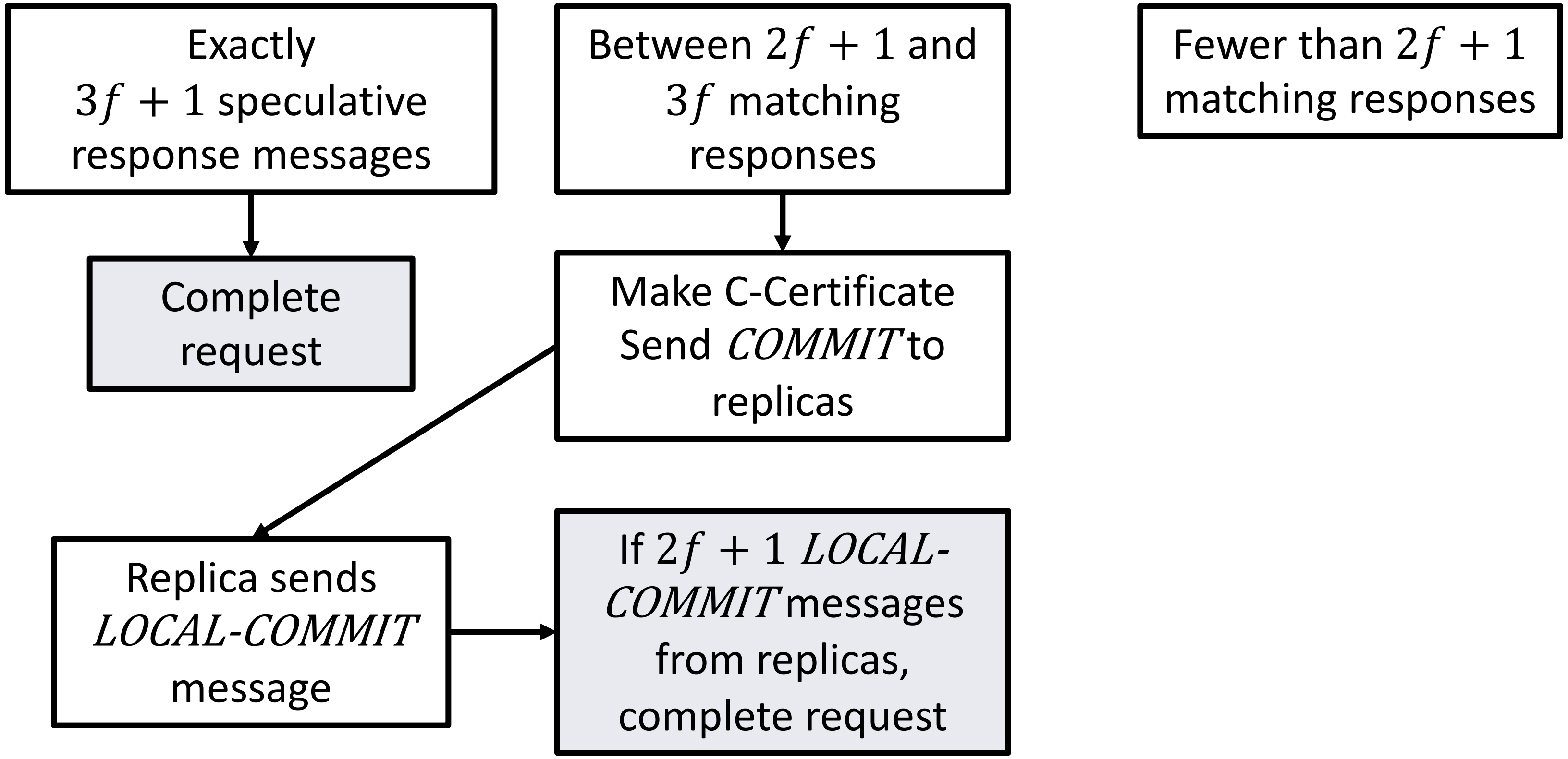


4b.2. If client receives a *LOCAL-COMMIT* message from $2f + 1$ replicas:

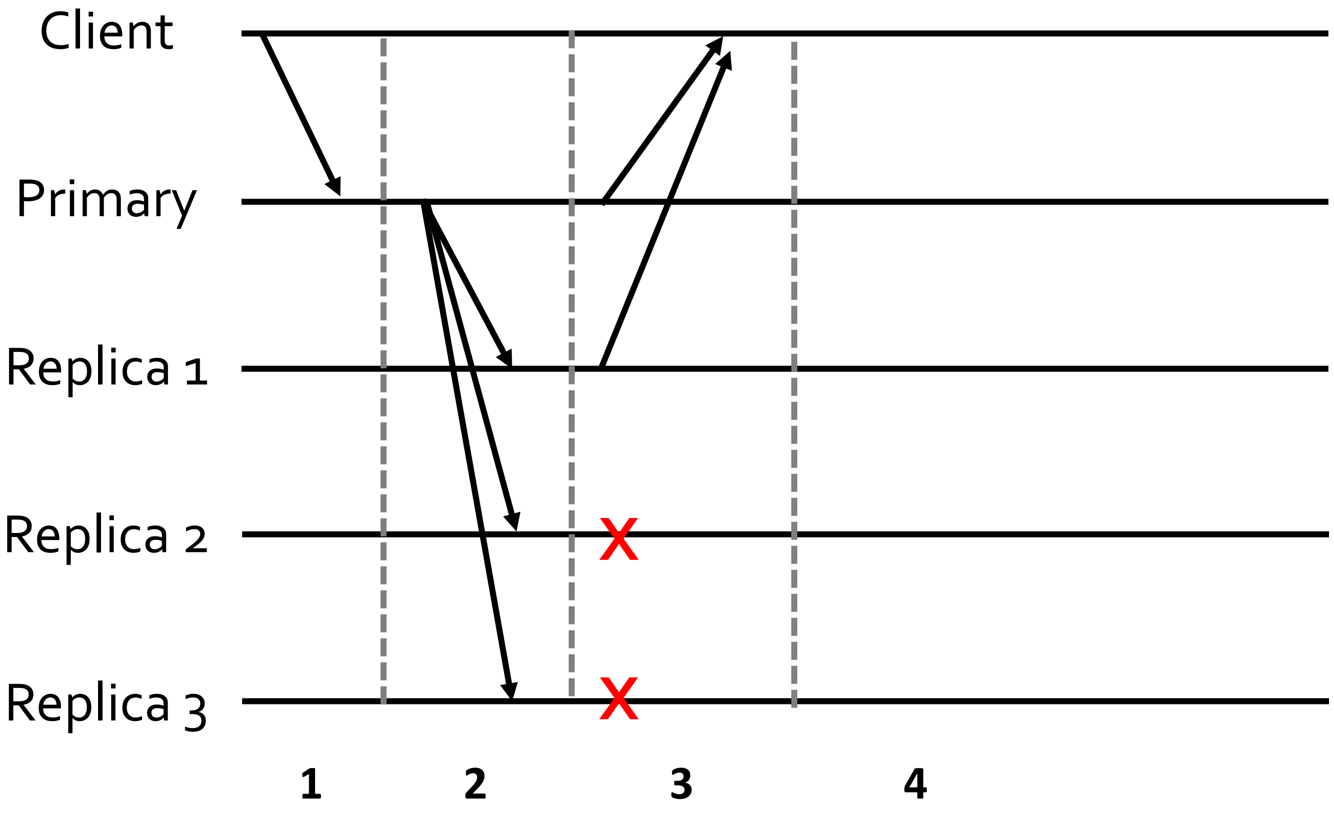
- Client completes the request.

Client Completion Summary

If client receives…

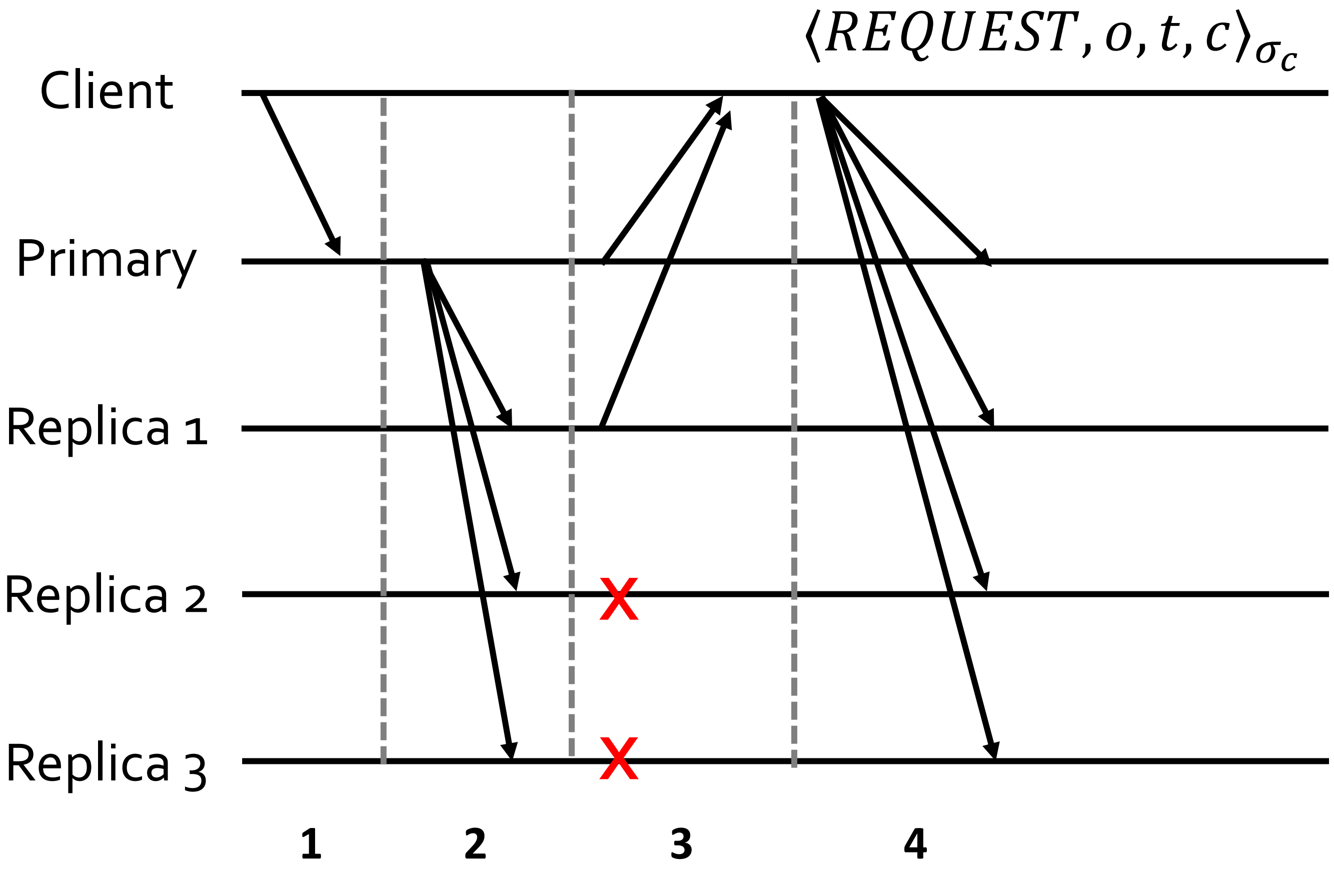


Agreement Subprotocol



4c. If client receives **fewer than $2f + 1$ matching responses:**

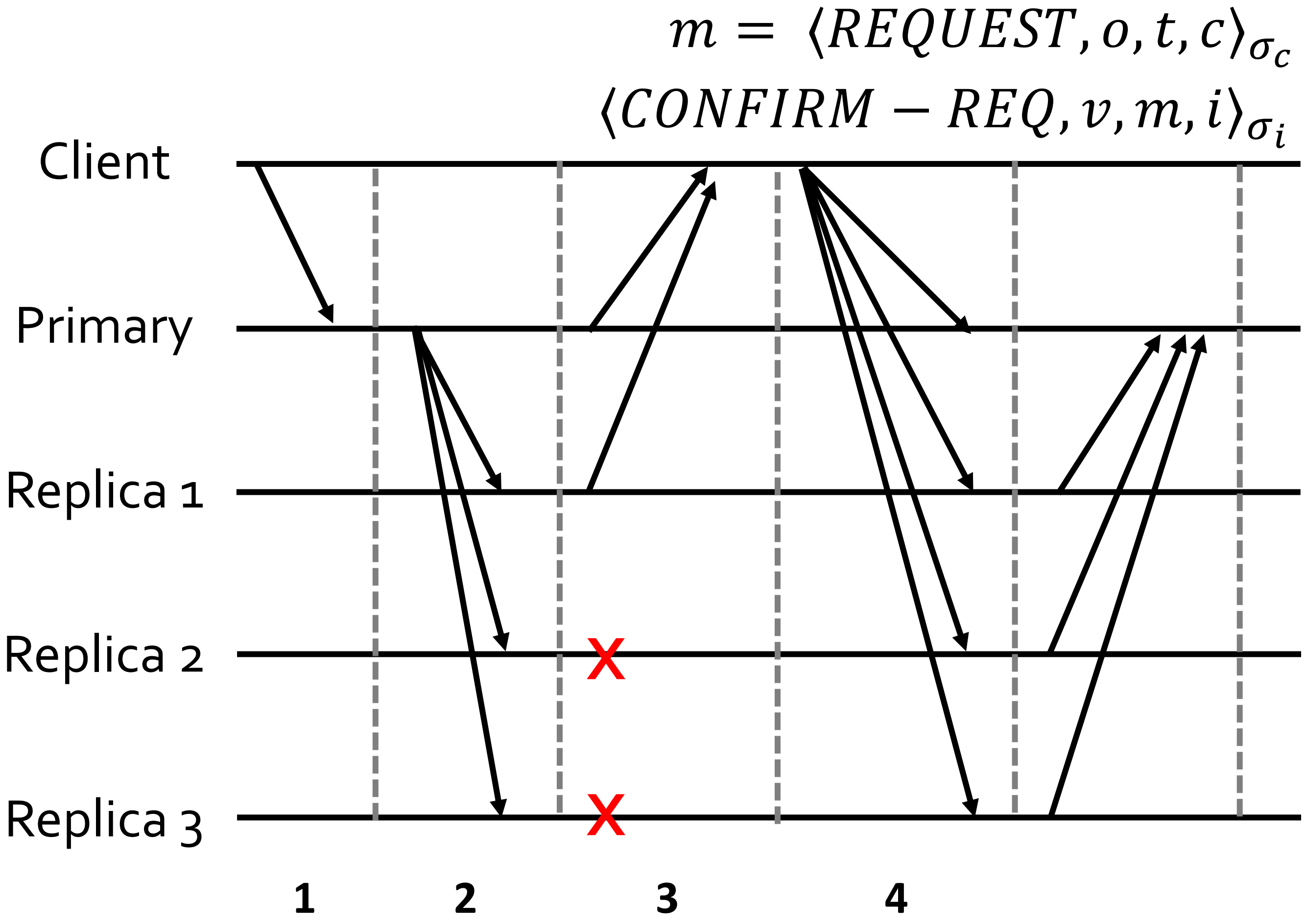
Agreement Subprotocol



4c. If client receives **fewer than $2f + 1$ matching responses:**

- Client resends its request to all replicas

Agreement Subprotocol

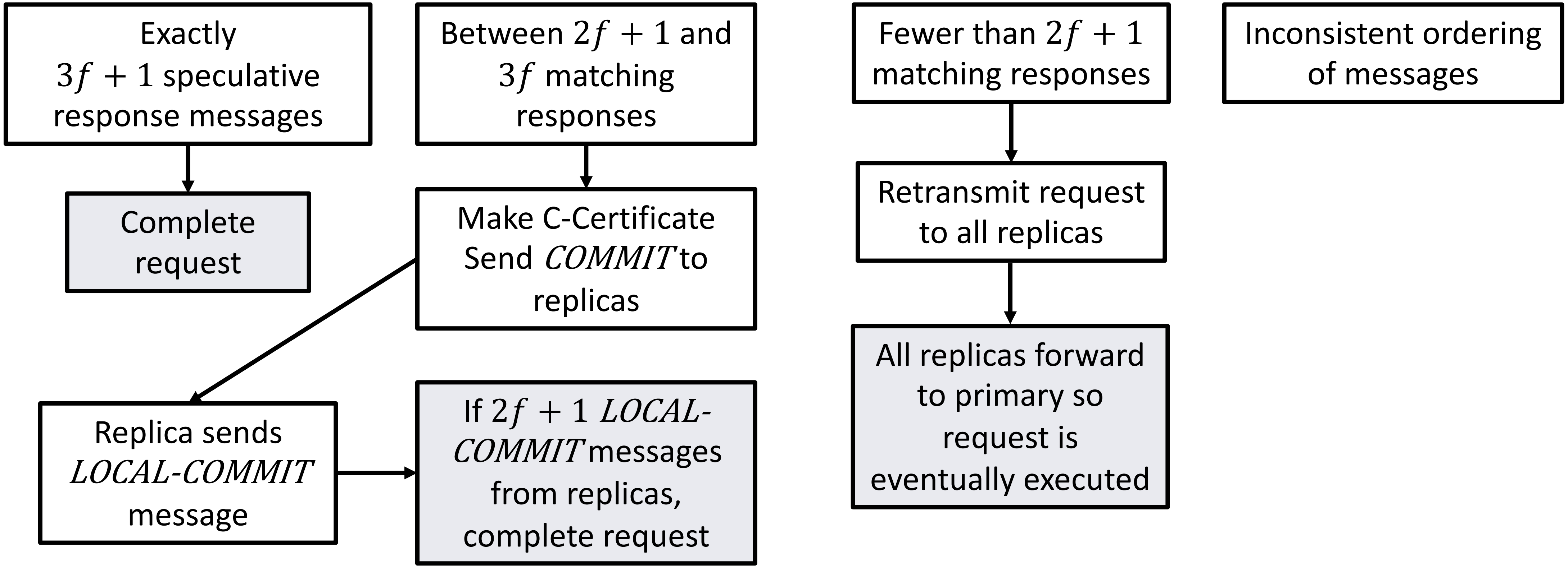


4c. If client receives **fewer than $2f + 1$ matching responses:**

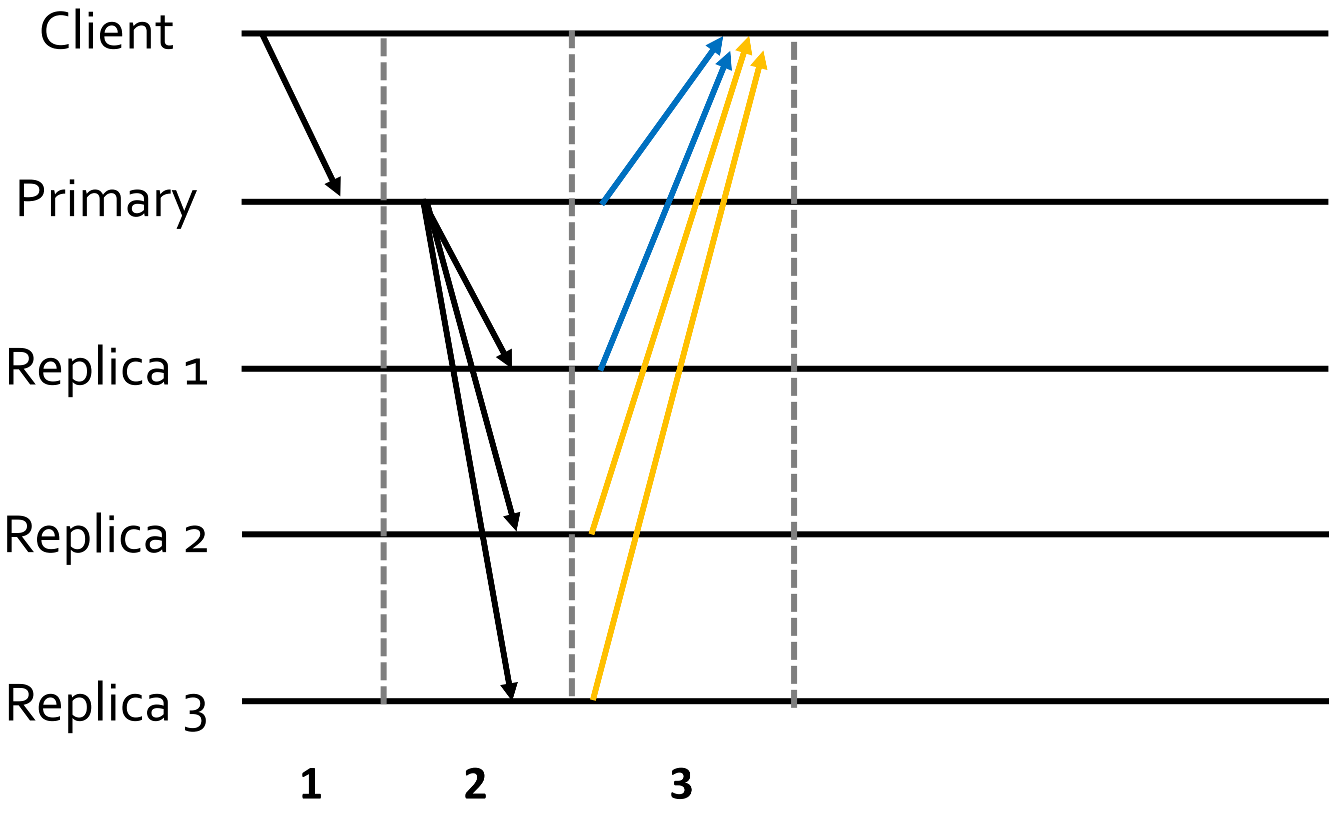
- Client resends its request to all replicas
- Replicas forward the request to the primary

Client Completion Summary

If client receives…

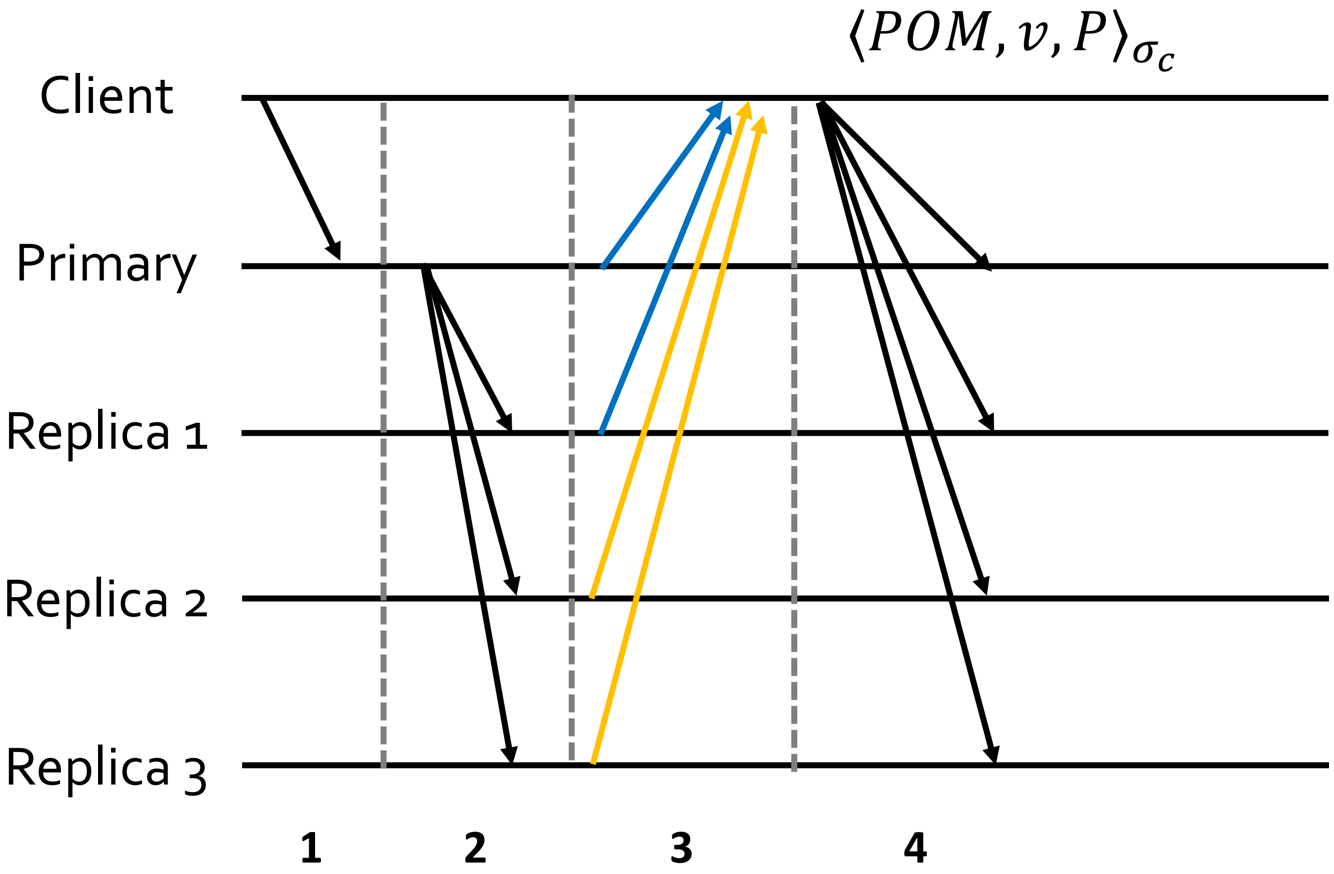


Agreement Subprotocol



4d. If client receives responses indicating **inconsistent ordering** by the primary:

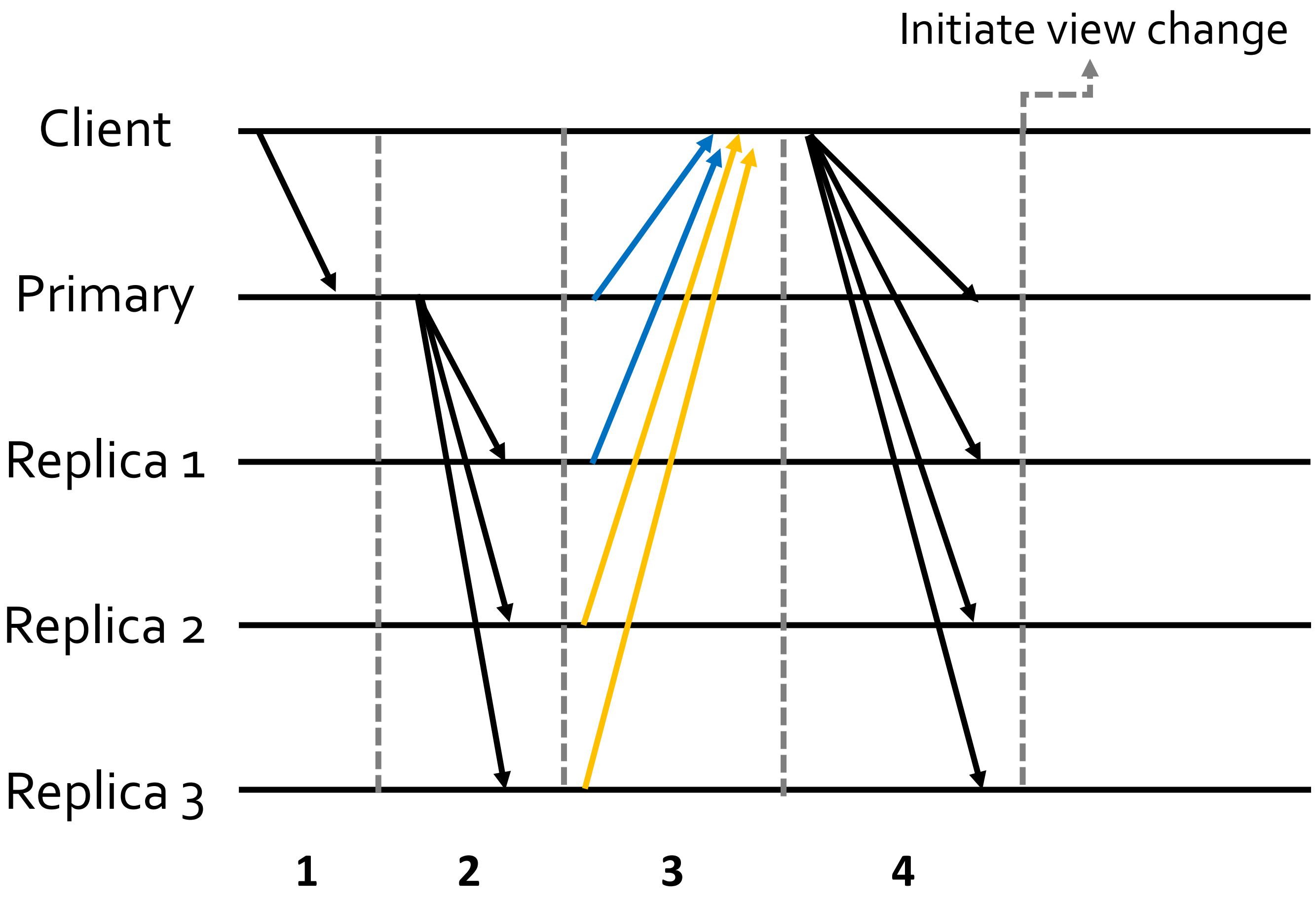
Agreement Subprotocol



4d. If client receives responses indicating **inconsistent ordering** by the primary:

- Client sends a proof of misbehavior to the replicas

Agreement Subprotocol

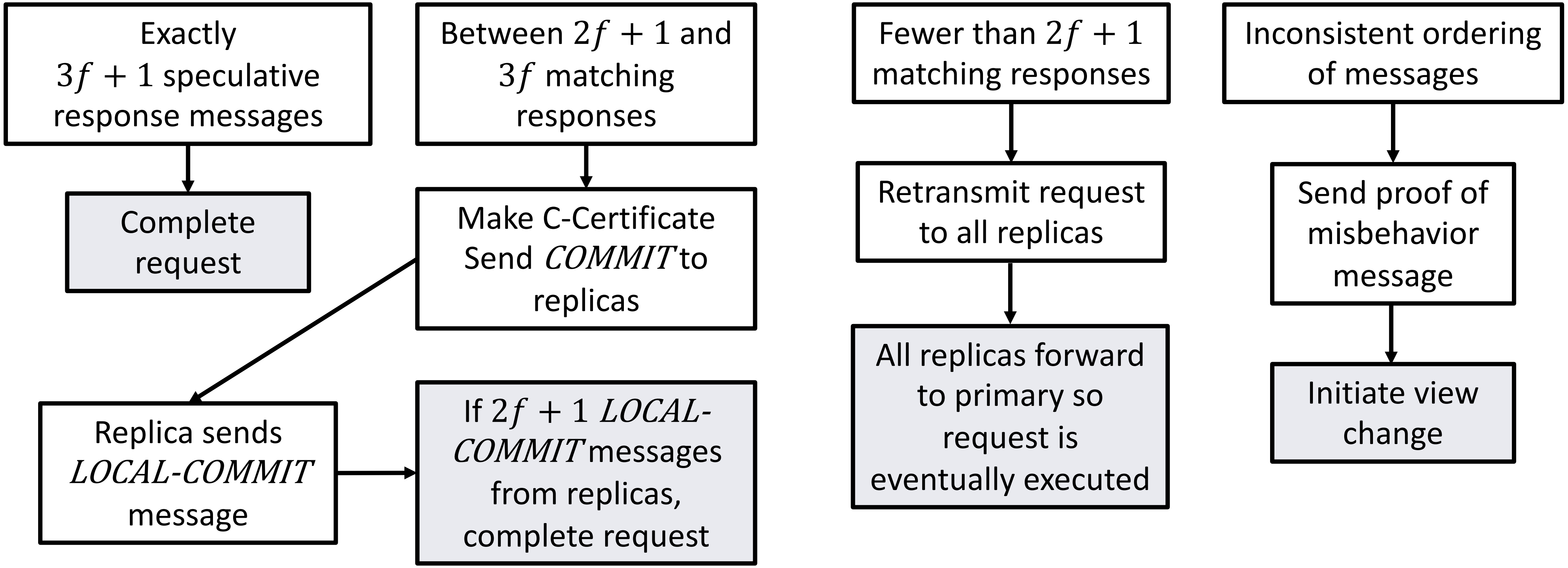


4d. If client receives responses indicating **inconsistent ordering** by the primary:

- Client sends a proof of misbehavior to the replicas
- Replicas initiate a view change to oust the faulty primary.

Client Completion Summary

If client receives…

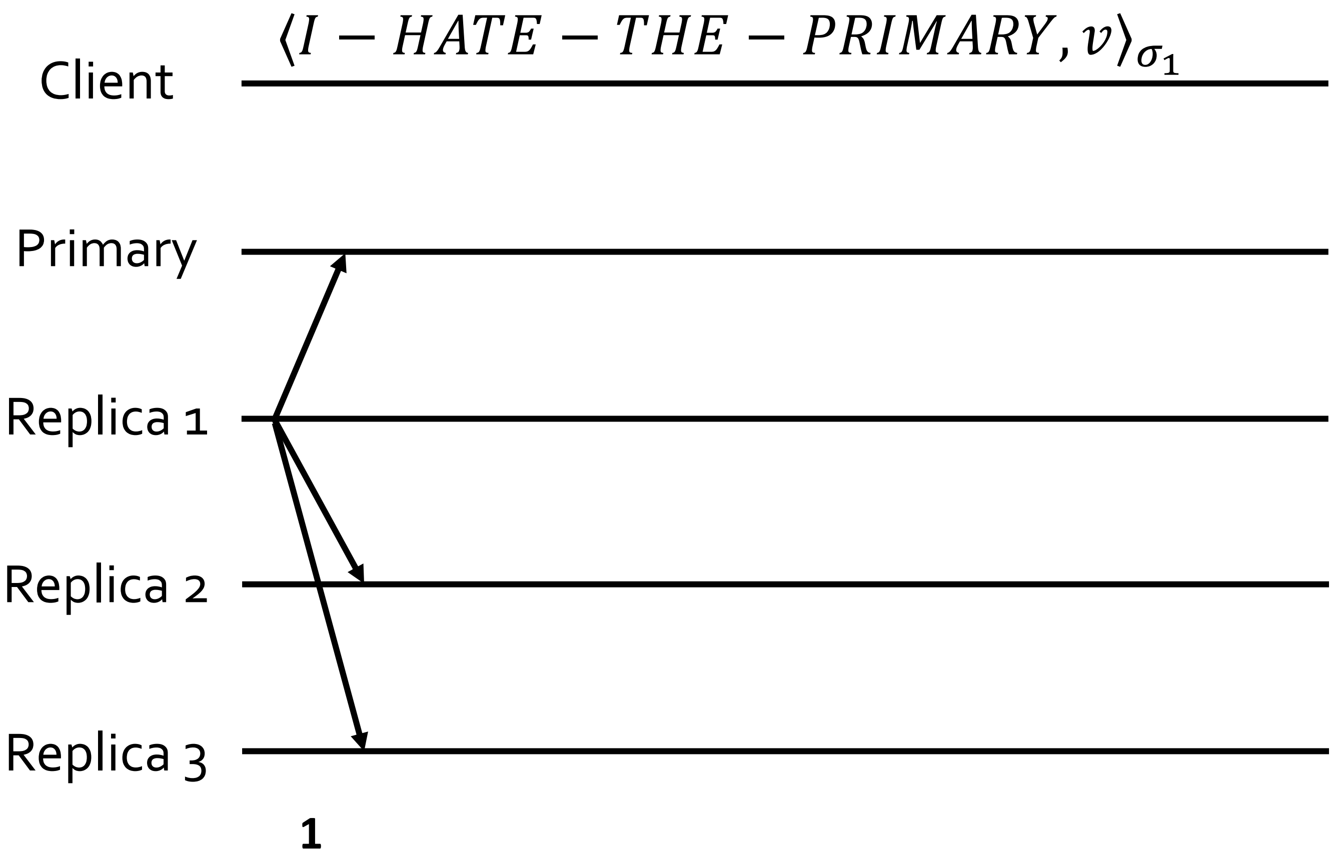


View Change Subprotocol

View Change Subprotocol

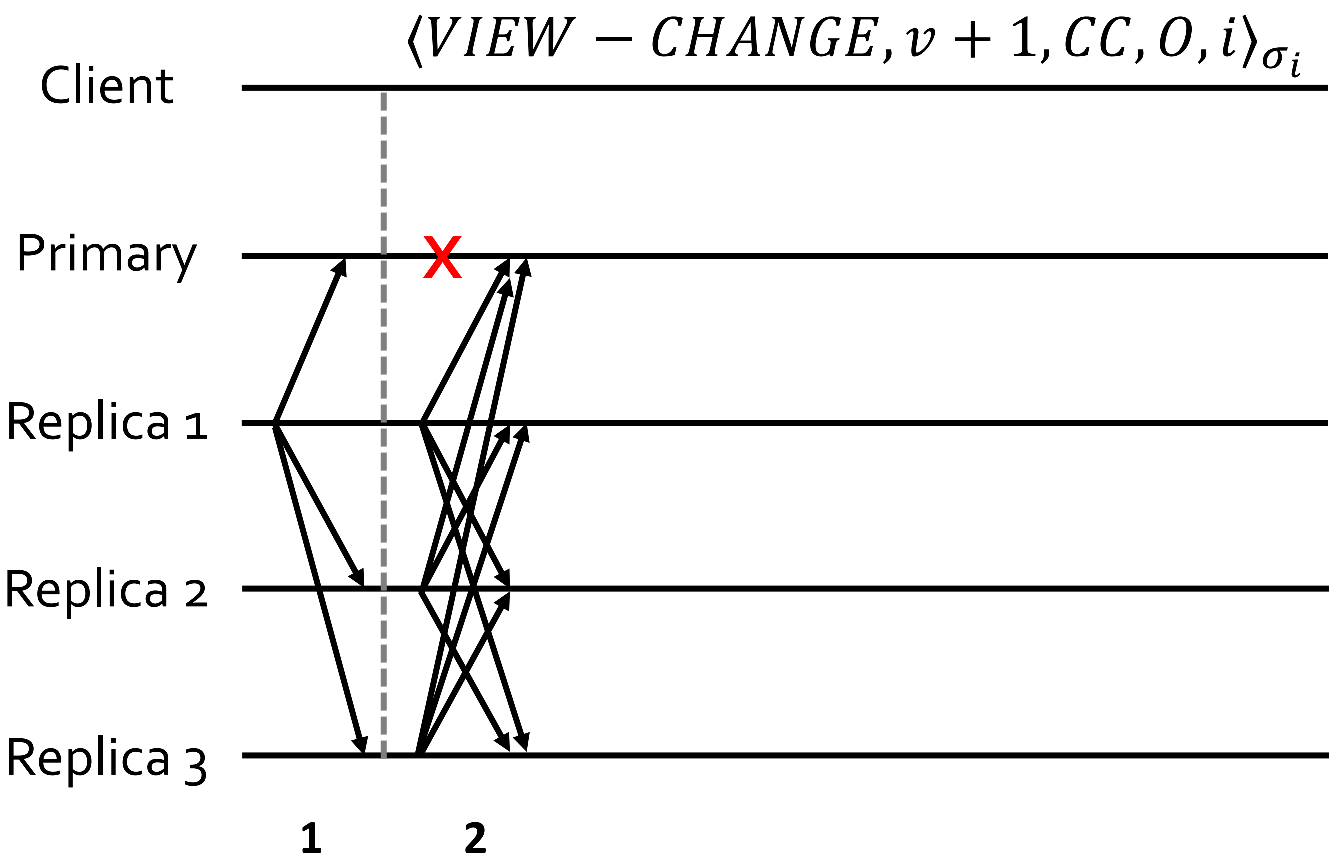
- To ensure liveness
  - Extra “I hate the primary” phase added
  - A correct replica will not abandon a view unless every other correct replica does as well
- To guarantee safety
  - Weakens condition under which a request appears in the new view’s history

View Change Subprotocol



VC1. Replica initiates the view change by sending an accusation against the primary to all replicas.

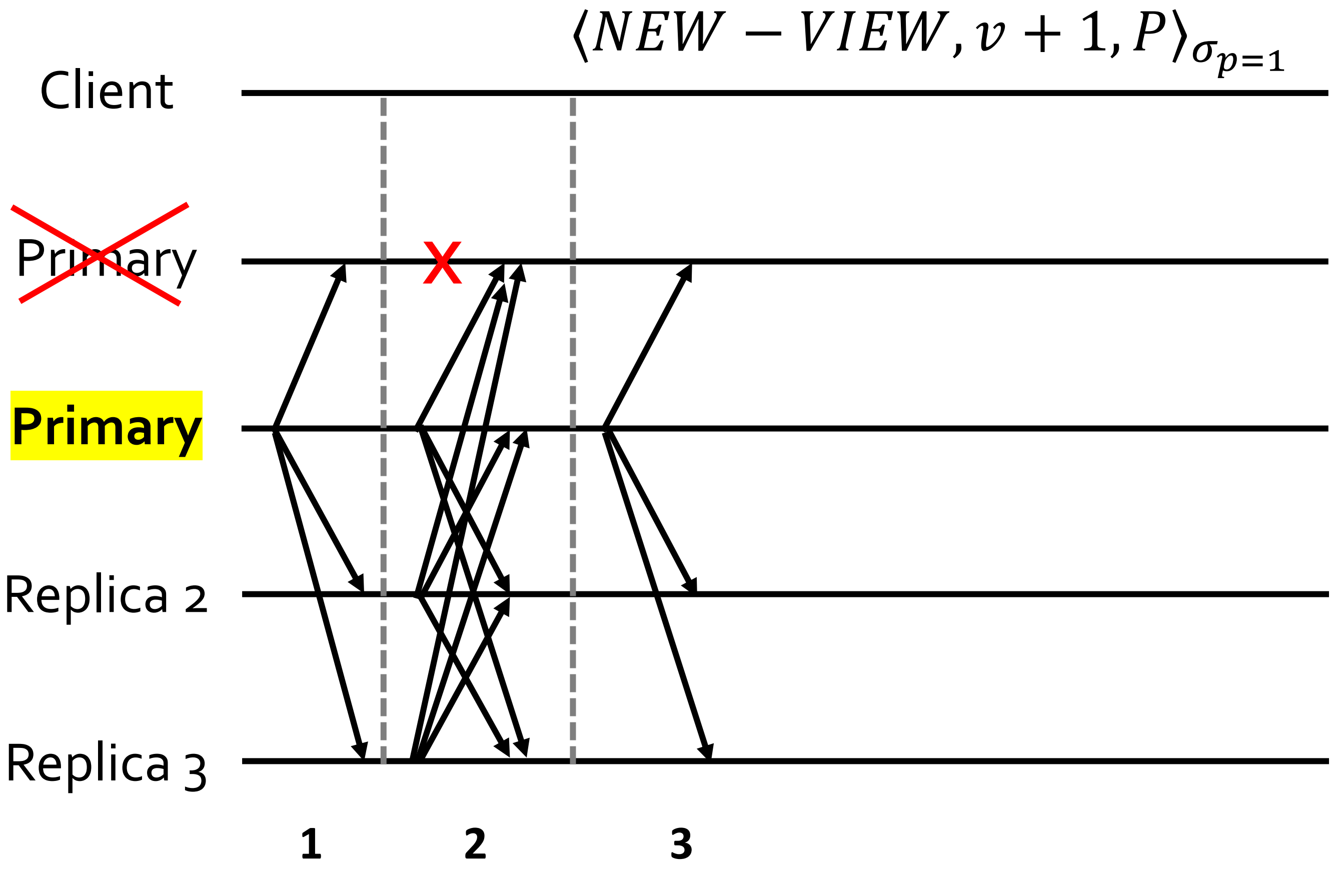
View Change Subprotocol



VC2.

- Replica receives $f+1$ accusations that the primary is faulty
- Then it commits to the view change.

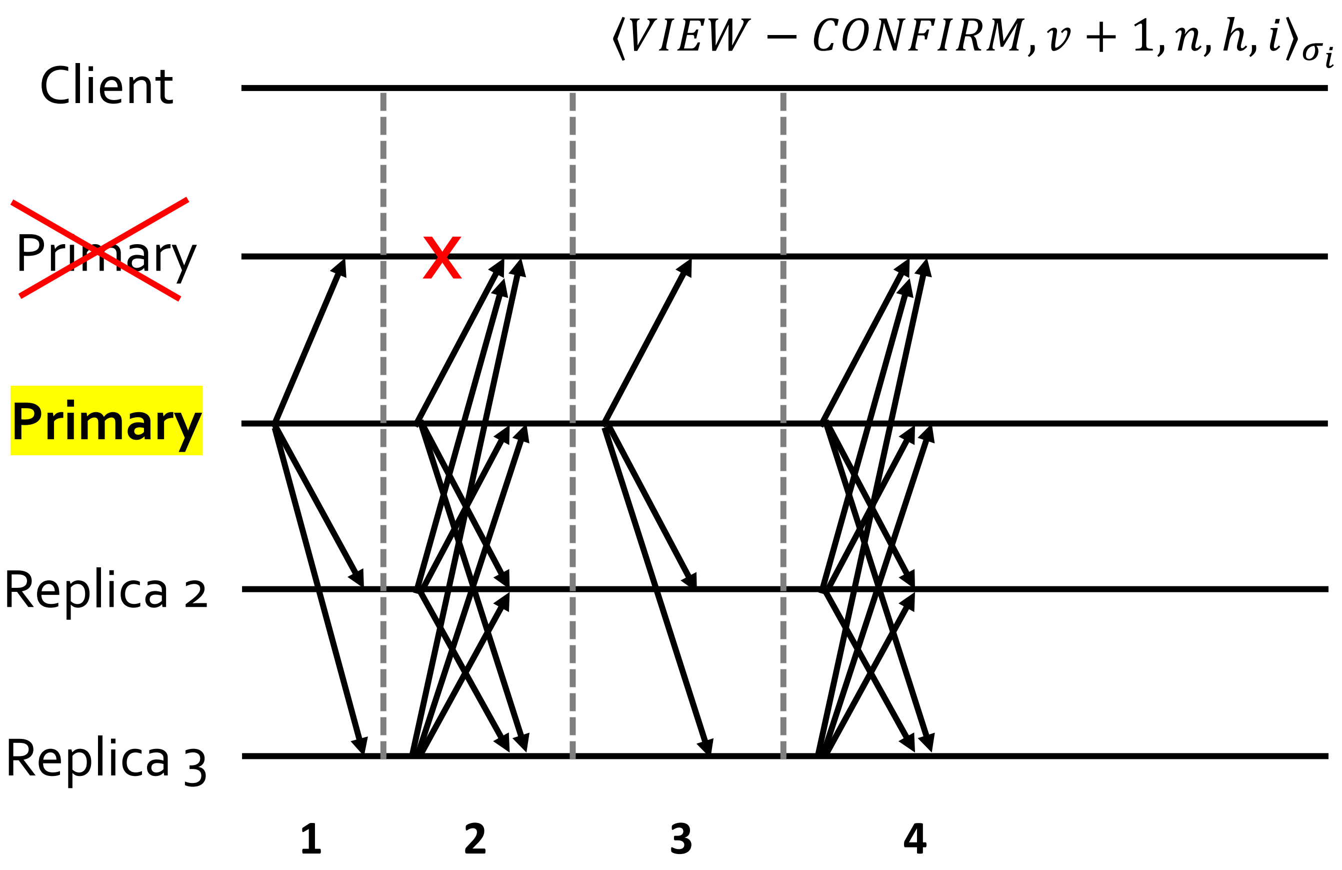
View Change Subprotocol



VC3.

- Replica receives $2f+1$ view change messages
- New primary sends new view message

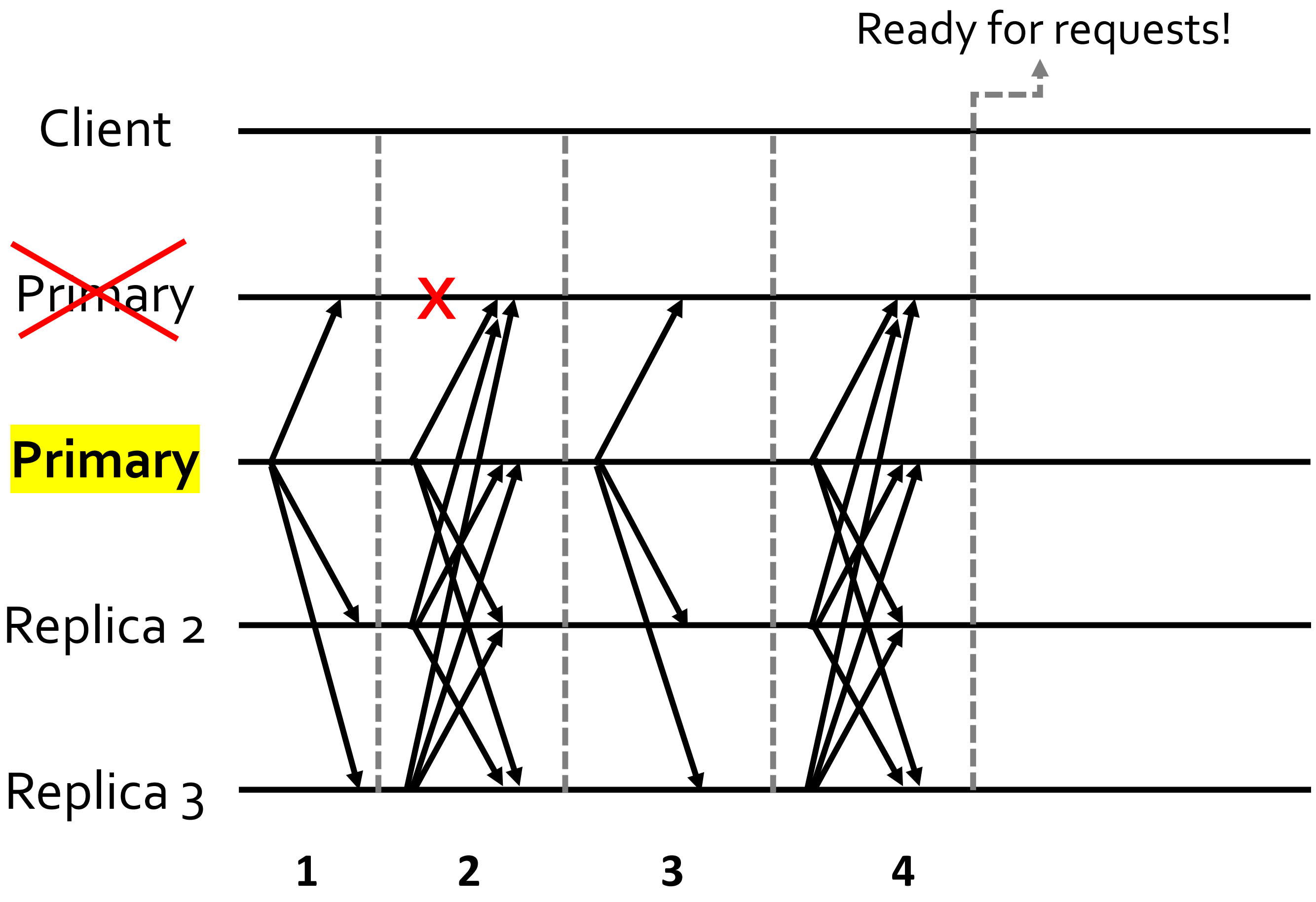
View Change Subprotocol



VC4.

- Replica receives a valid new view message
- Then it sends a view confirmation message to all other replicas.

View Change Subprotocol



VC5:

- Replica receives $2f + 1$ matching *VIEW-CONFIRM* messages
- Begins accepting requests in the new view

Implementation Optimizations

- Replacing signatures with MACs
- Separating agreement from execution
- Request batching
- Caching out of order requests
- Read-only optimization
- Single execution response
- Preferred Quorums

Midterm question!

Evaluation

Evaluation: Throughput

$*f = 1$ tolerated faults

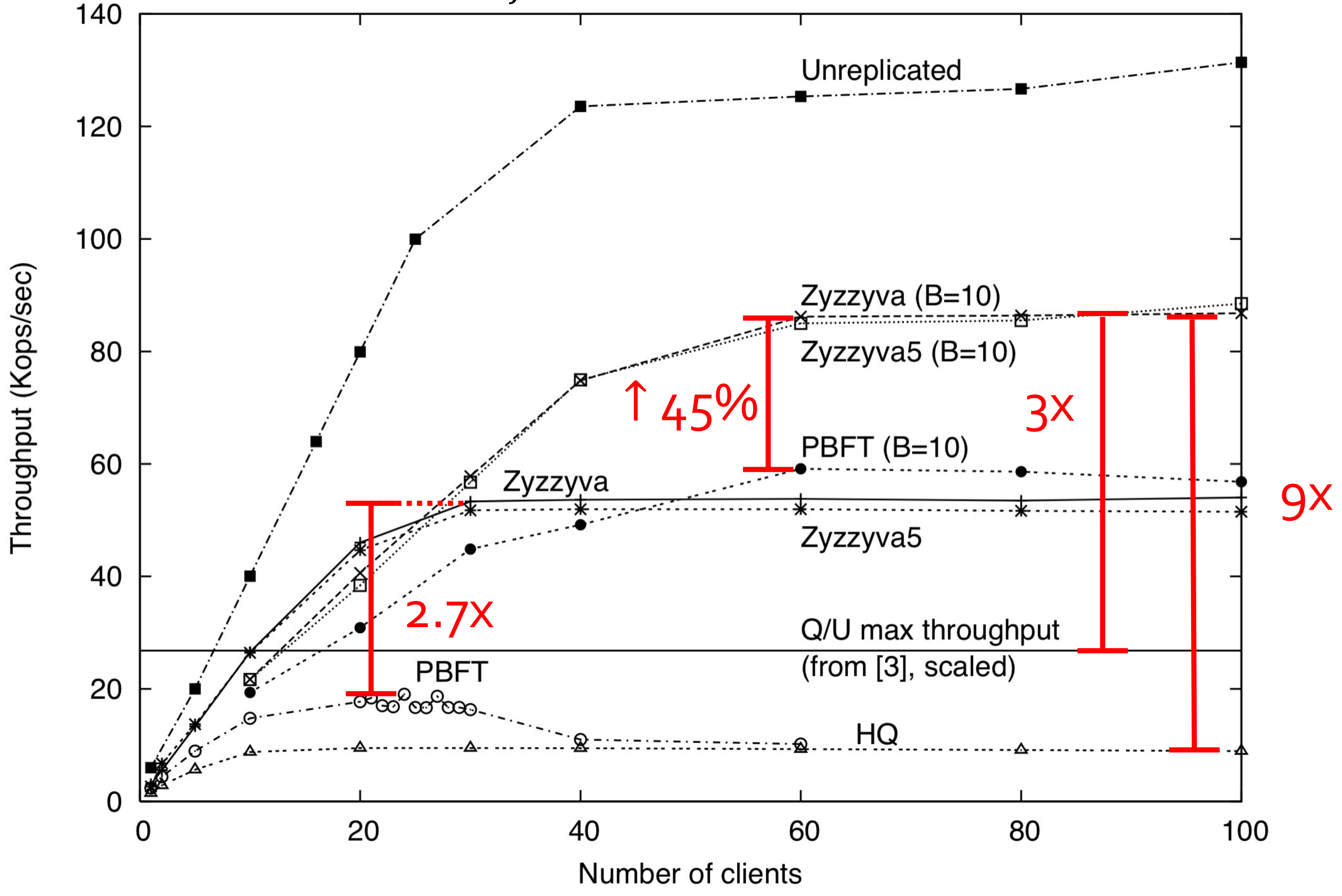


Figure 3

Evaluation: Latency

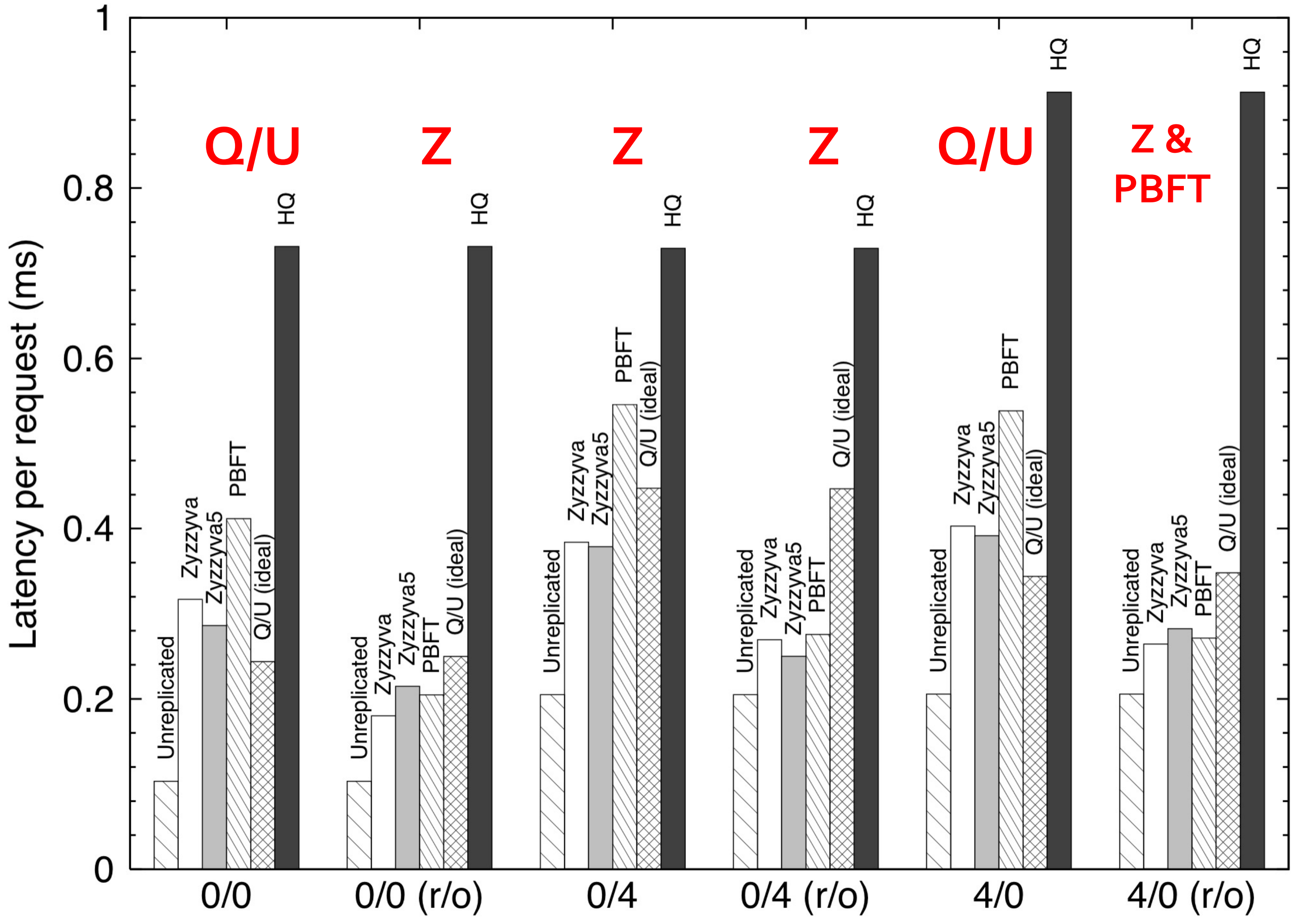


Figure 4

Evaluation: Fault Scalability

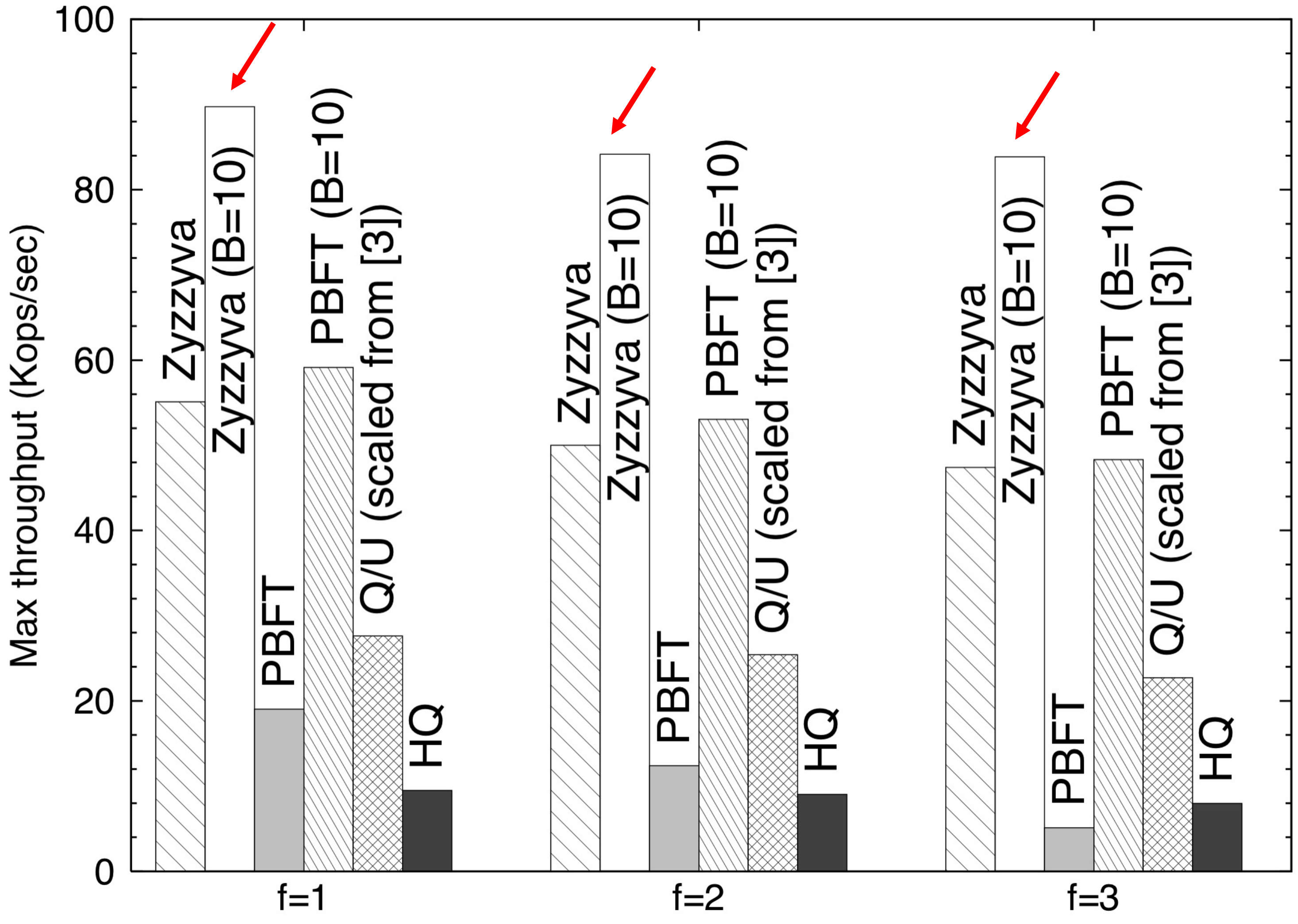


Figure 6

Robust to increasing *f*

Conclusion

“By systematically exploiting speculation, Zyzzyva exhibits significant performance improvements over existing BFT services. … approach[ing] the theoretical lower bounds for any BFT protocol.”

References

[1] M. Abd-El-Malek, G. Ganger, G. Goodson, M. Reiter, and J. Wylie. Fault scalable byzantine fault-tolerant services. In *Proc. SOSP*, Oct. 2005.

[2] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proc. OSDI*, February 1999.

[3] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shira. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In *Proc. OSDI*, Nov. 2006

[4] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: Speculative Byzantine Fault Tolerance. In *Proc. SOSP*, October 2007.

Questions?