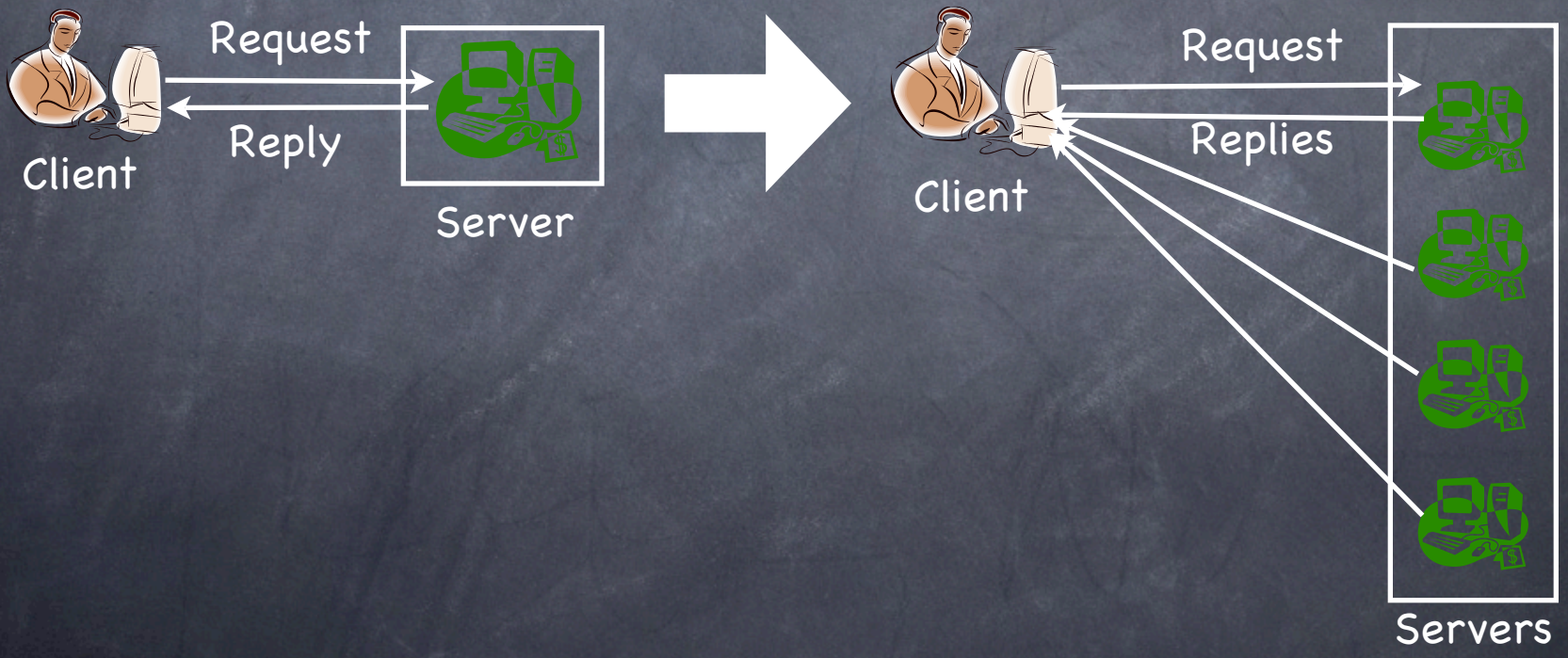
Zyzzyva
Speculative Byzantine Fault Tolerance

Ramakrishna Kotla

L. Alvisi, M. Dahlin, A. Clement, E. Wong
University of Texas at Austin

The Goal

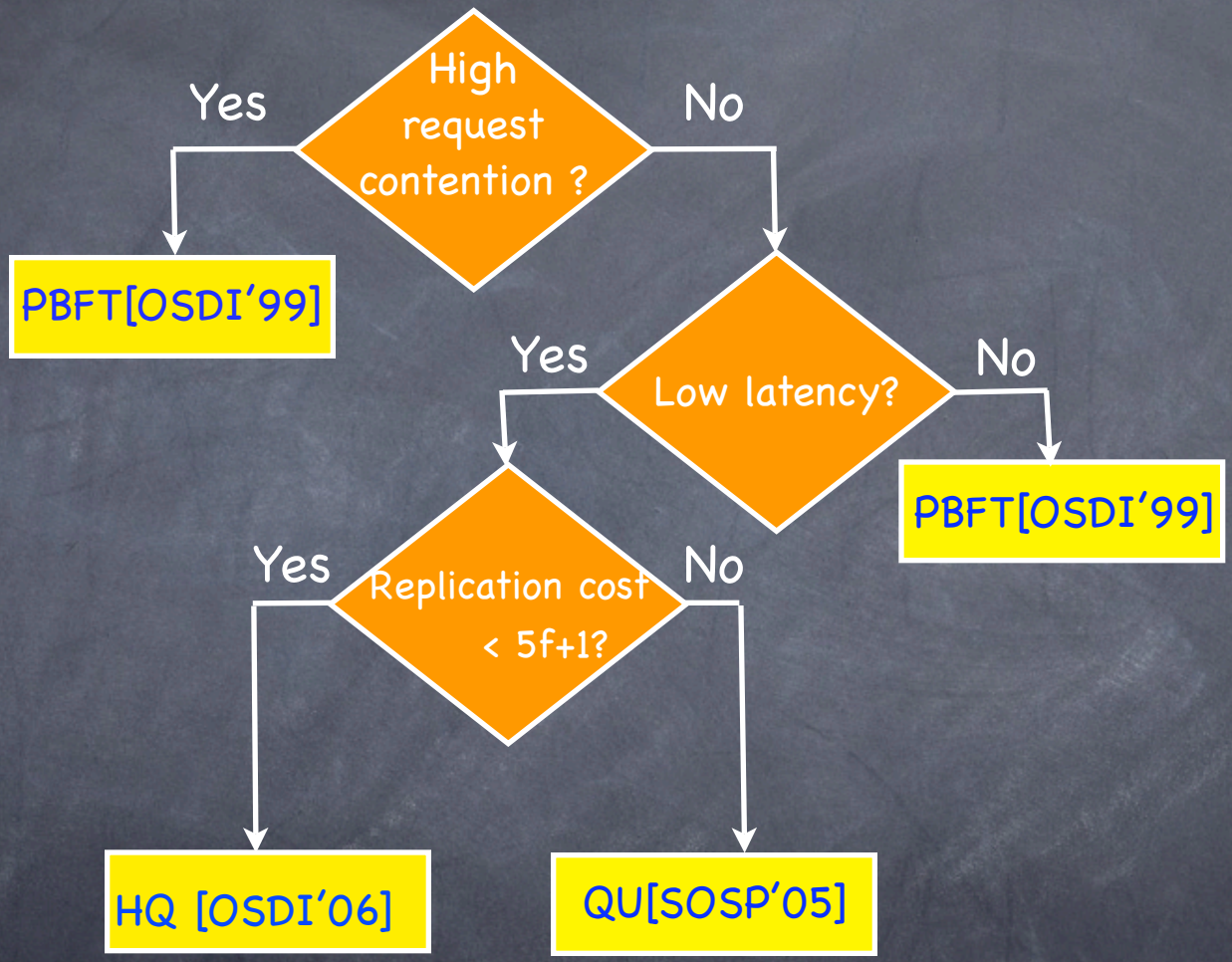
Transform high-performance service into high-performance and reliable service



BFT state machine replication

- BFT state-of-the-art
  - Practical Byzantine Fault Tolerance [OSDI'99, OSDI'00]
  - Generalized abstraction [SOSP'01]
  - Reduced replication cost [SOSP'03]
  - High Throughput [DSN'04]
  - Applications: Farsite[OSDI'02], Oceanstore[FAST'03]
  - Quorum based approaches: Q/U[SOSP'05], HQ[OSDI'06]
- Promising approach to build reliable systems

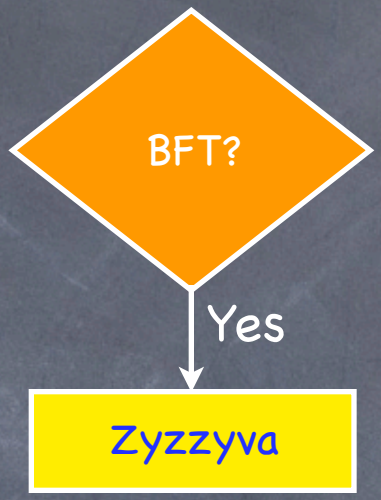
Why another BFT protocol?



HQ[OSDI'06]

- BFT state-of-the-art is too complex

Zyzzyva: Rethinks BFT state machine replication



- Outperform existing BFT approaches
- High performance: Comparable to unreplicated services
- Low overhead: Approaches lower bounds

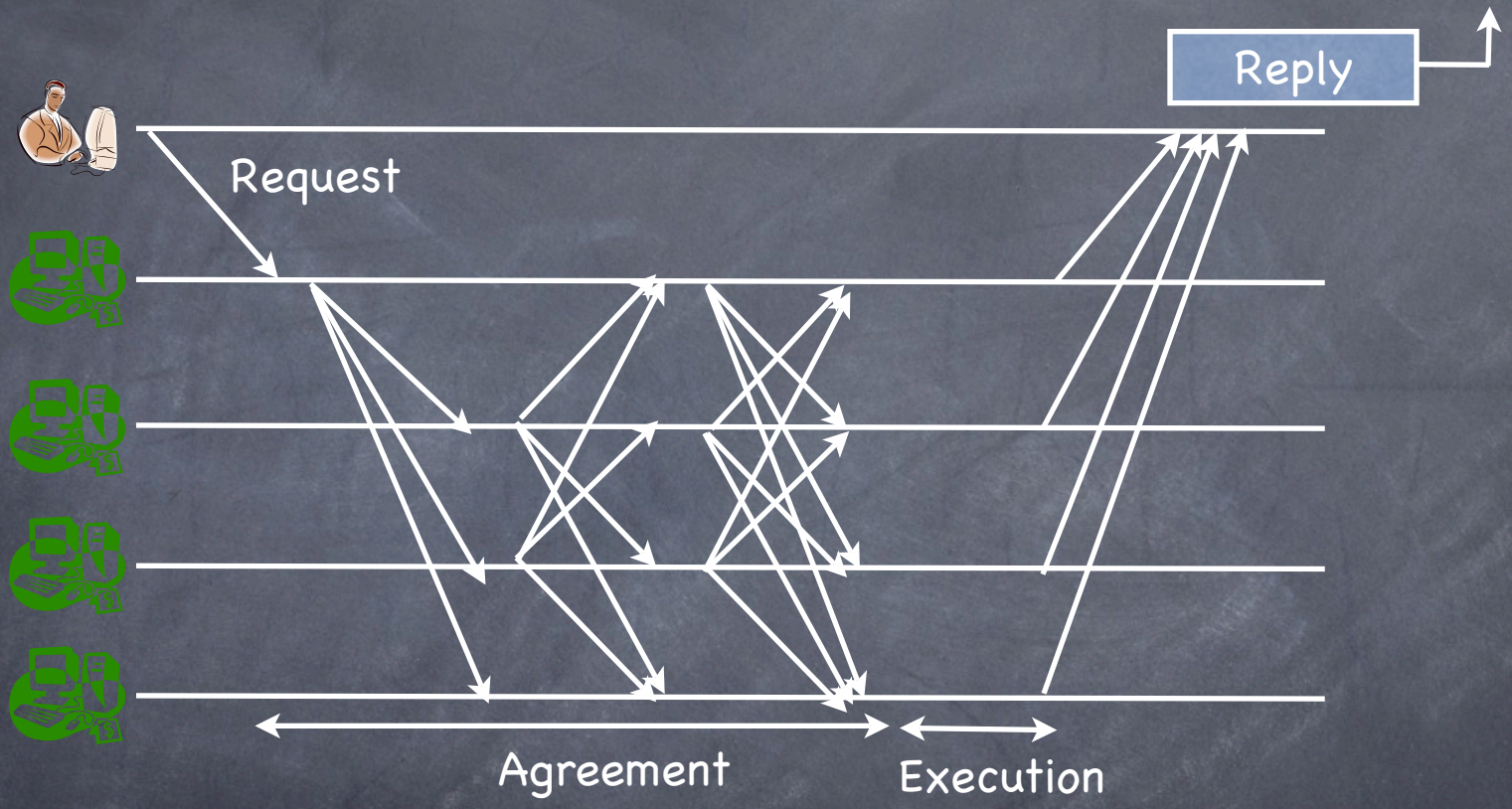
Zyzzyva: Outline

- **Rethink state machine replication**
- Speculation: Avoiding explicit replica agreement
- Speculative BFT: Double edged sword
- Implementation and Optimizations
- Evaluation

State Machine Replication

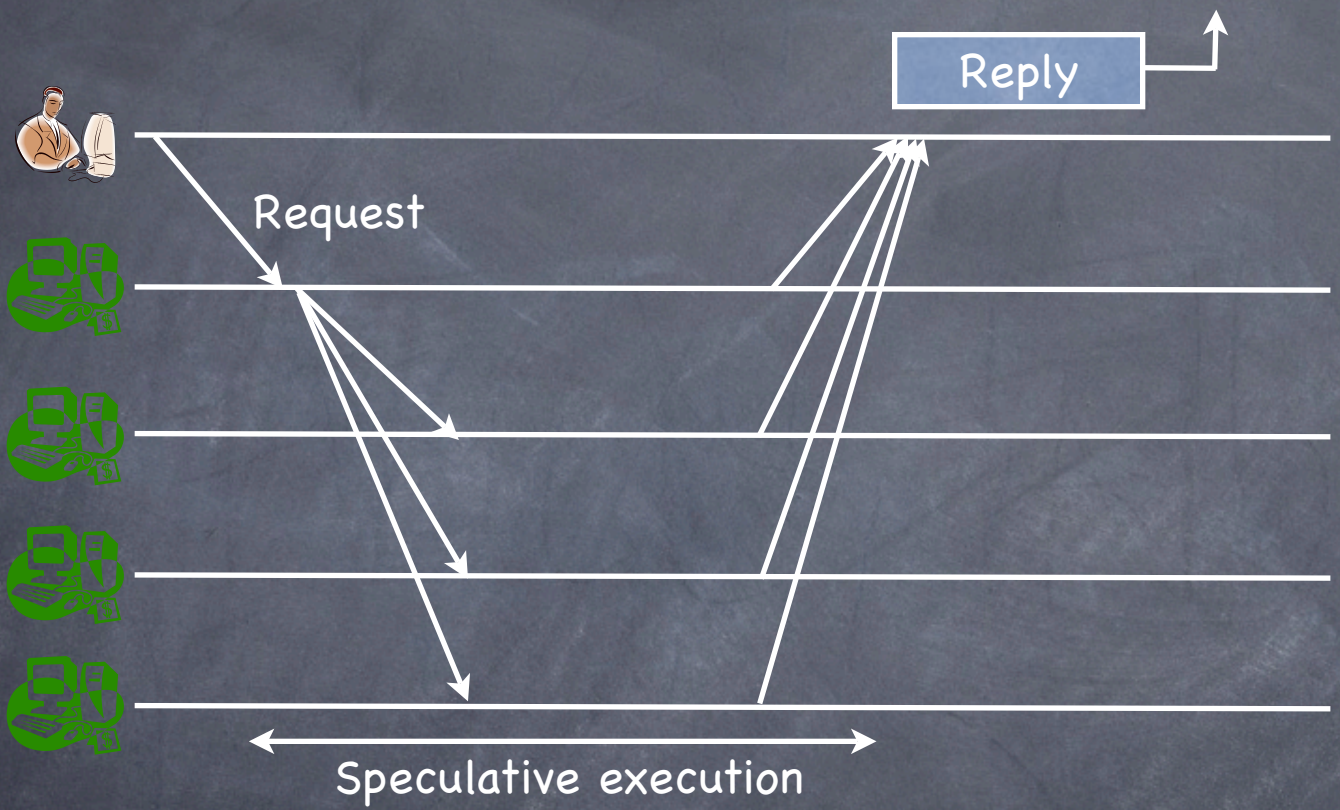
- Service is replicated to tolerate failures
- Requirement: Applications observe centralized service
- How: Replicas execute requests in the same order
  - Agreement: Replicas agree on the request order
  - Execution: Replicas execute requests in agreed order

Traditional BFT state machine replication



- Replicas agree on the request order before executing
  - Cost: Agreement protocol overhead

Zyzzyva: Speculative BFT Replication



- Replicas execute requests without agreement
  - Cost: No explicit replica agreement

Avoid explicit replica agreement

- Idea: Leverage clients to avoid explicit agreement
- Intuition: Output commit at the client
  - Sufficient: Client knows that system is consistent
  - Not required: Replicas know that they are consistent
- How: Client commits output only if system is consistent
  - Applications observe centralized service

Zyzzyva: Outline

- Rethink state machine replication
- **Speculation: Avoiding explicit replica agreement**
- Speculative BFT: Double edged sword
- Implementation and Optimizations
- Evaluation

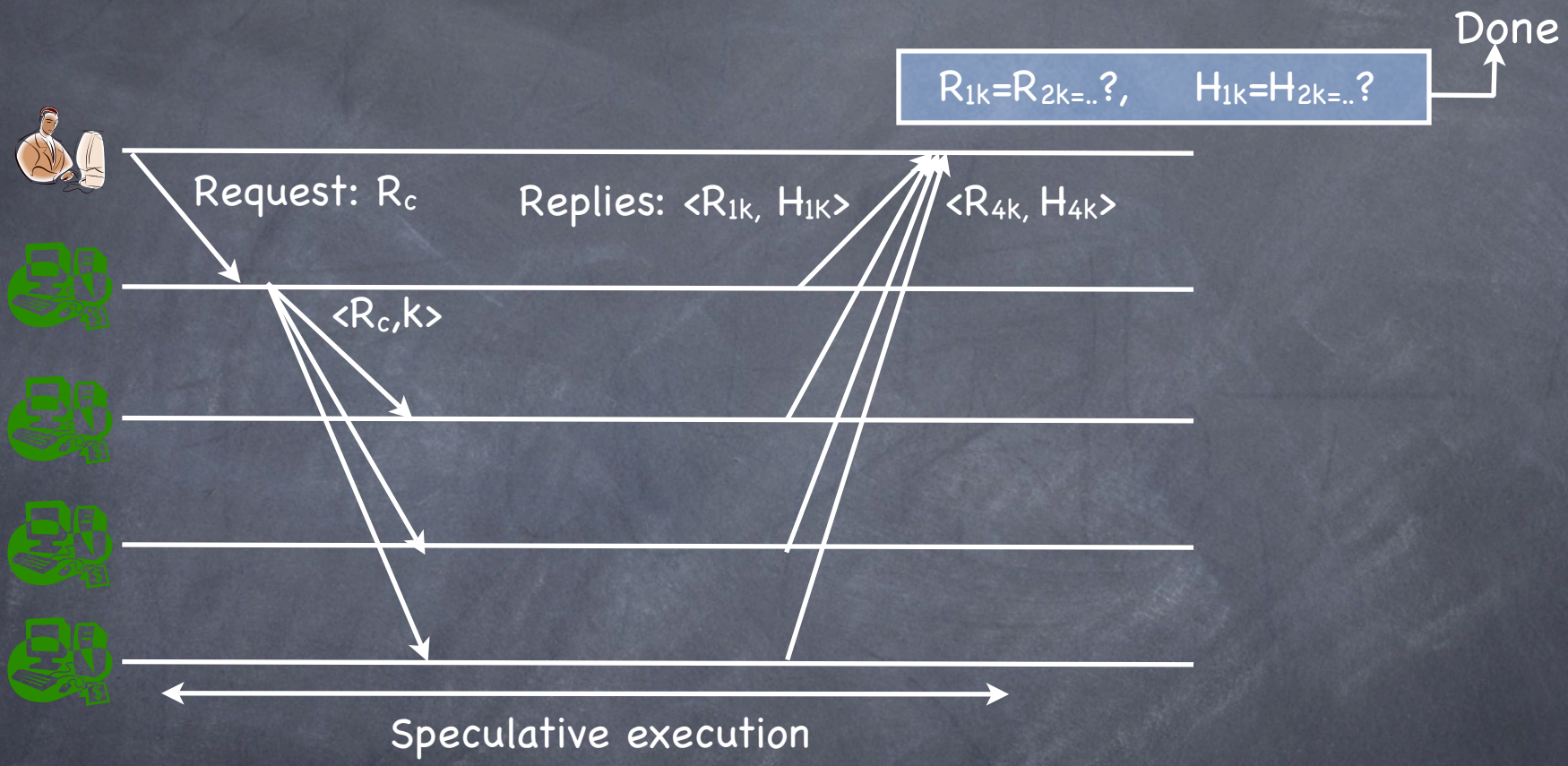
Speculative BFT: Leveraging client

- Idea: Leverage clients to avoid explicit agreement
- Intuition: Output commit at clients and not replicas
  - Replicas need not know if system is consistent
- How: Client can verify if reply is stable
  - Before committing a reply to the application
  - Stable reply: Replicas are in consistent state

Speculative BFT: Request history

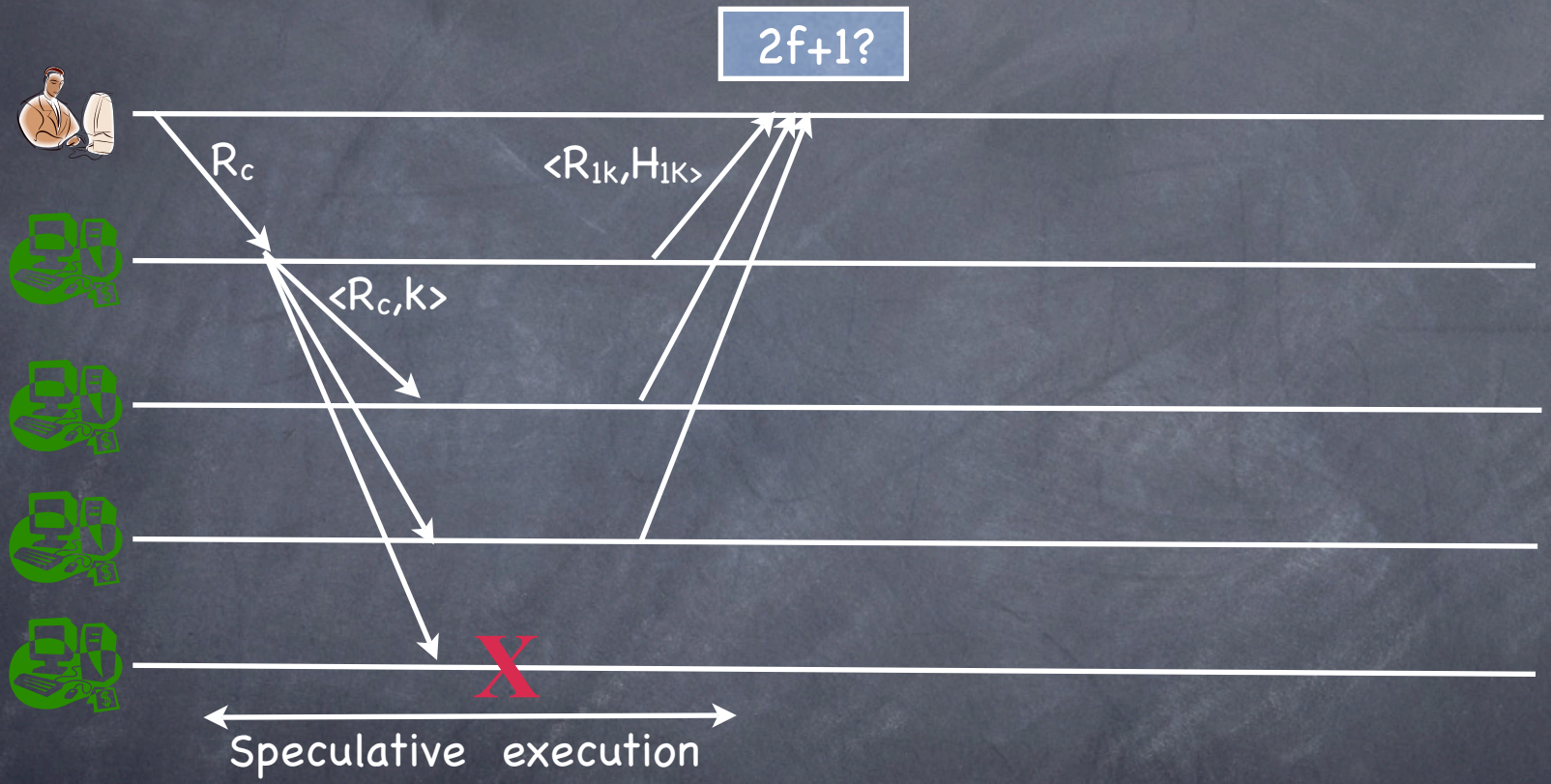
- Request history allows client to verify stable reply
- Replicas include request history in the replies
  - Request history: Ordered set of requests executed
  - Replies include application response and request history
  - $<R_{ik}, H_{ik}>$: Reply from a replica i after executing request k

Stable: Unanimous reply



- Client commits the output when all replies match
  - All correct replicas are in consistent state

Replies: Only majority match

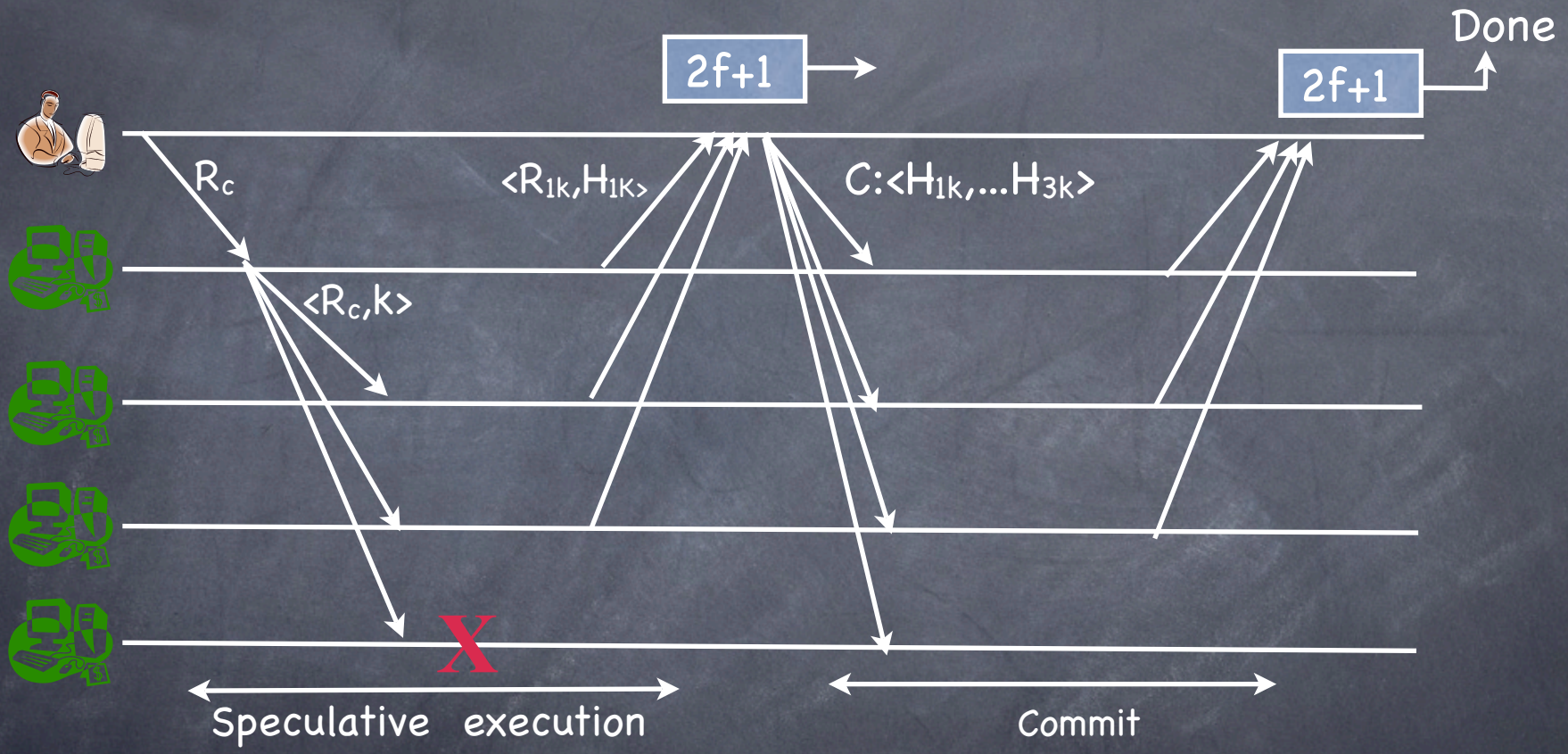


- Majority of correct replicas share the same history
  - Client receives at least 2f+1 matching replies

Stable reply with failures

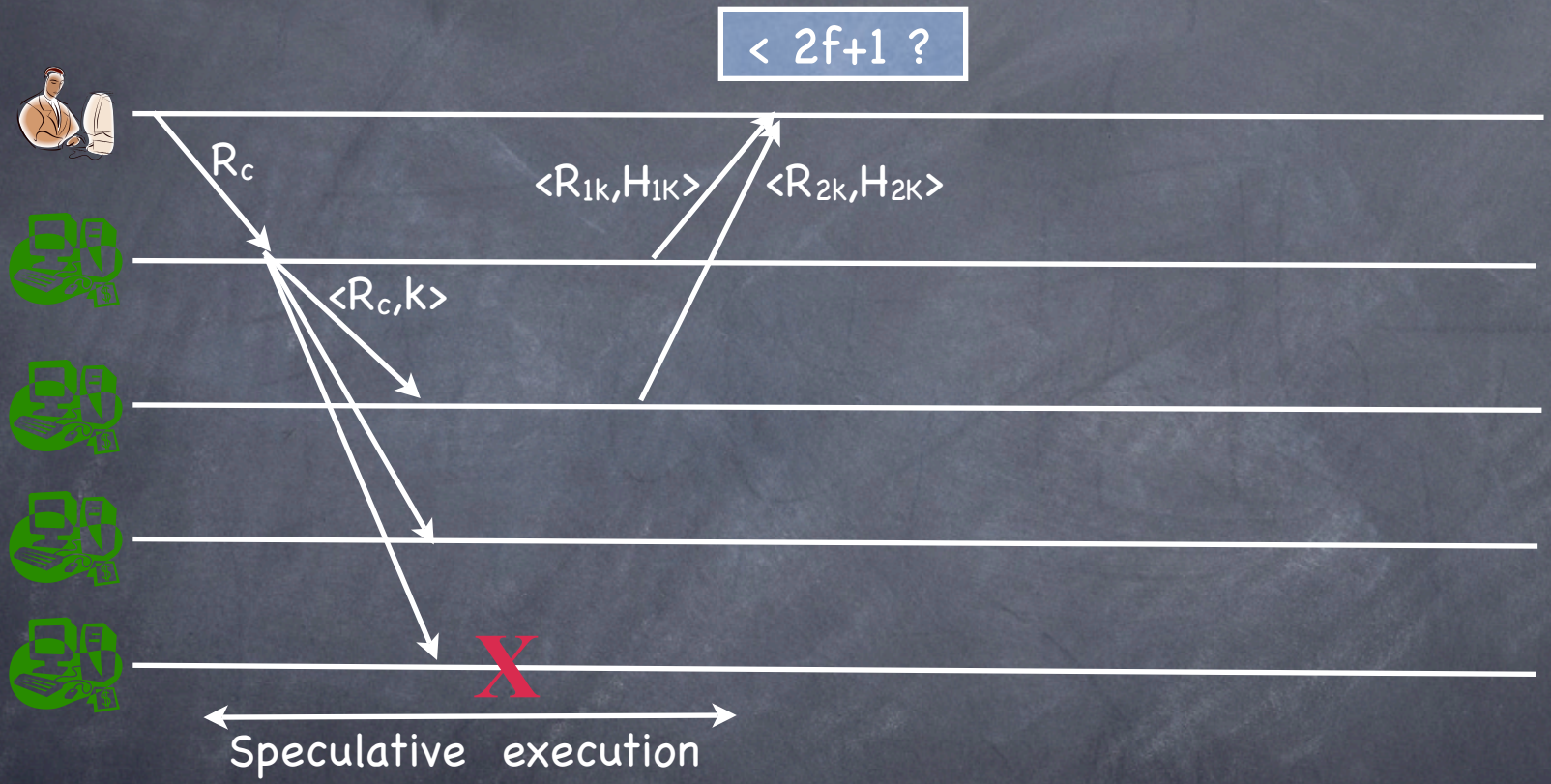
- Client can make progress with additional work
- Sufficient: Majority of correct replicas can prove
  - That they share request history to other replicas
  - Intuition: Eventually all correct replicas agree
- Commit phase: Client deposits commit certificate
  - Commit certificate consists of 2f+1 matching histories
  - Client commits after receiving 2f+1 matching acks

Stable reply: Majority



- Client deposits commit certificate
- Client commits when it receives 2f+1 matching acks

Failures: Primary or Network



- Client receives fewer than 2f+1 matching replies
- View change: Client retransmissions act as hint

Zyzzyva: Speculative BFT

- Same consistency guarantees as traditional BFT
  - Application observes centralized service
- Leverage clients to avoid explicit replica agreement
  - Significantly lower overhead

Zyzzyva: Outline

- Rethink state machine replication
- Speculation: Avoiding explicit replica agreement
- **Speculative BFT: Double edged sword?**
- Implementation and Optimizations
- Evaluation

Can a faulty client block?

- By not depositing the commit certificate
- Faulty clients cannot block other correct clients
- Liveness: Correct clients ensure system progress
  - Protocol uses cumulative request histories
  - Correct clients commit all previous requests as well
  - Faulty client can only affect its own progress

Can a faulty client compromise safety?

- By committing inconsistent history?
- Faulty clients cannot compromise safety
  - Faulty clients cannot deposit inconsistent histories
- Safety:
  - Faulty clients cannot forge request histories
  - No two valid commit certificates can have varying prefixes

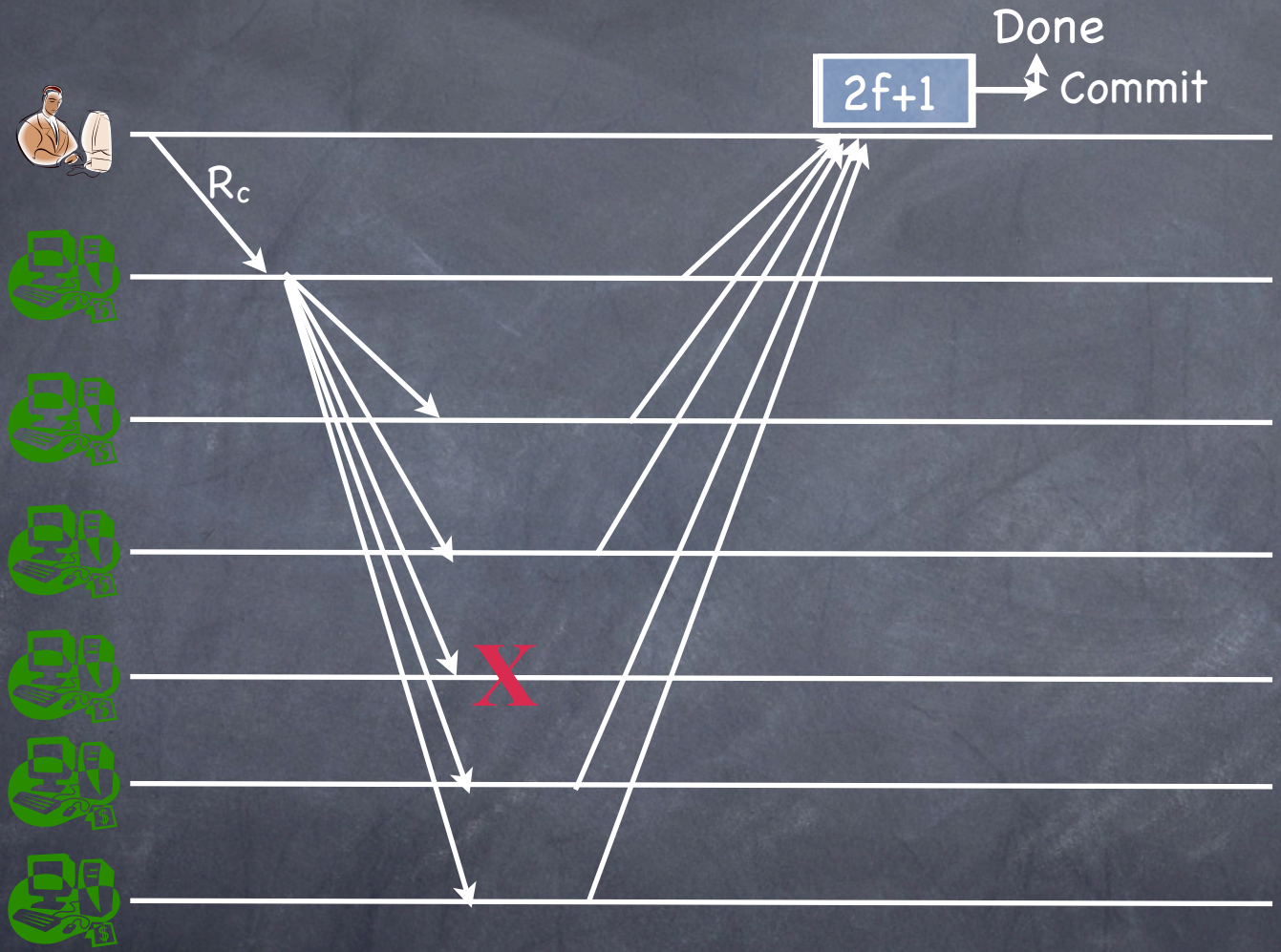
Zyzzyva: Outline

- Rethink state machine replication
- Speculation: Avoiding explicit replica agreement
- Speculative BFT: Double edged sword
- **Implementation and Optimizations**
- Evaluation

Implementation details

- Checkpoint protocol: Garbage collect histories
- View change protocol: Elect new primary
- Optimizations
  - Replace digital signatures with MACs
  - Application state is replicated at only 2f+1 replicas
  - Request batching

Optimization: Making faulty case faster



- Zyzzyva5: Speeds up using 5f+1 replicas
  - Completes in a single phase with f faulty replicas

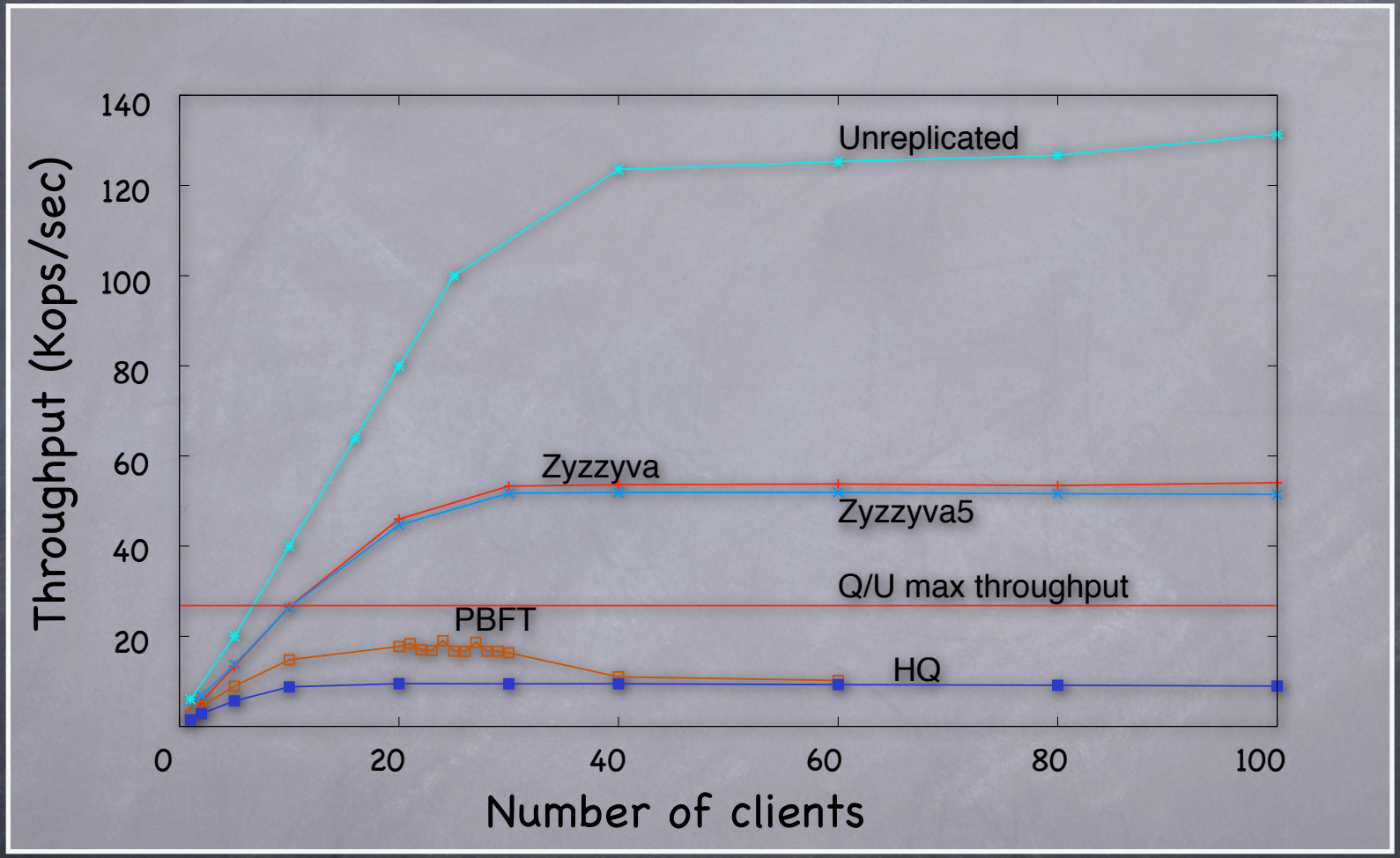
Zyzzyva: Outline

- Rethink state machine replication
- Speculation: Avoiding explicit replica agreement
- Speculative BFT: Double edged sword
- Implementation and Optimizations
- **Evaluation**

Evaluation setup

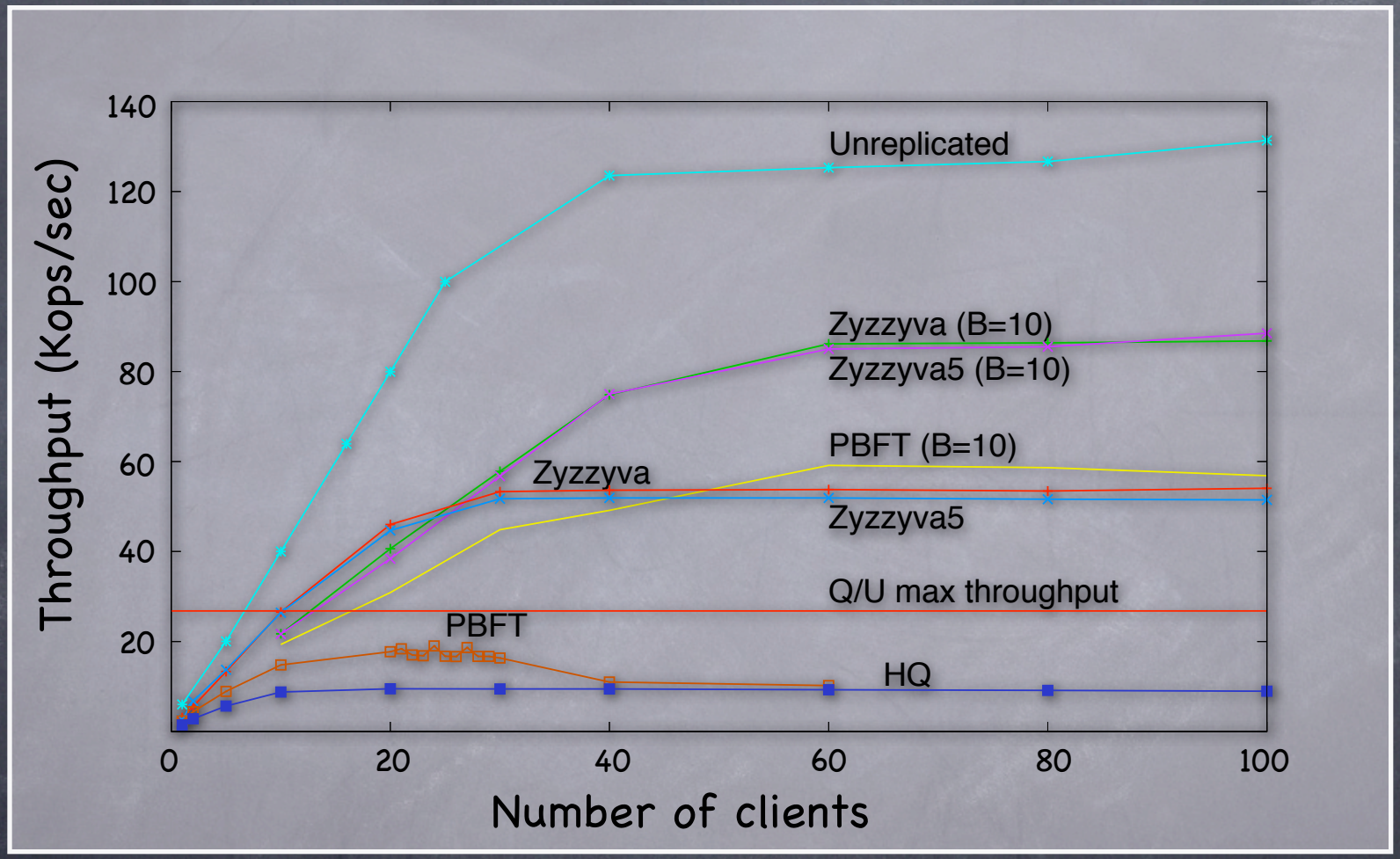
- Zyzzyva replication library
- Compare with other protocols
  - PBFT[OSDI'99], QU[SOSP'05], HQ[OSDI'06], Unreplicated
- Client-server workload
  - Different request/reply payloads
- Configuration: Tolerate 1 faulty node in the system
  - 20 Machines: 3.0 GHz running Linux 2.6 Kernel
  - LAN: 1 Gbps ethernet links

Throughput



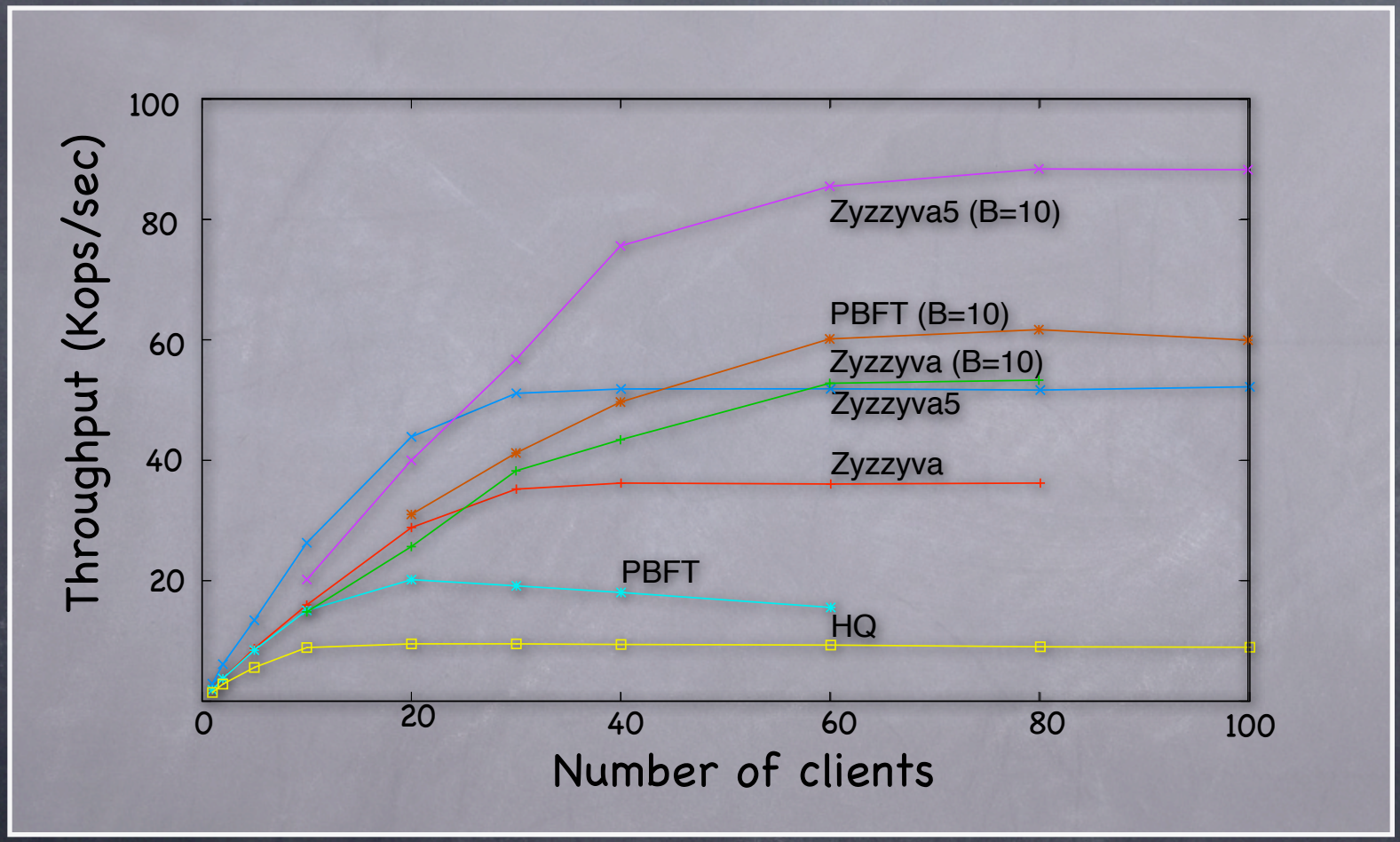
- Speculation improves throughput significantly

Throughput



- Speculation improves throughput significantly
- Zyzzyva within 35% of unreplicated service

Throughput: With a faulty backup node



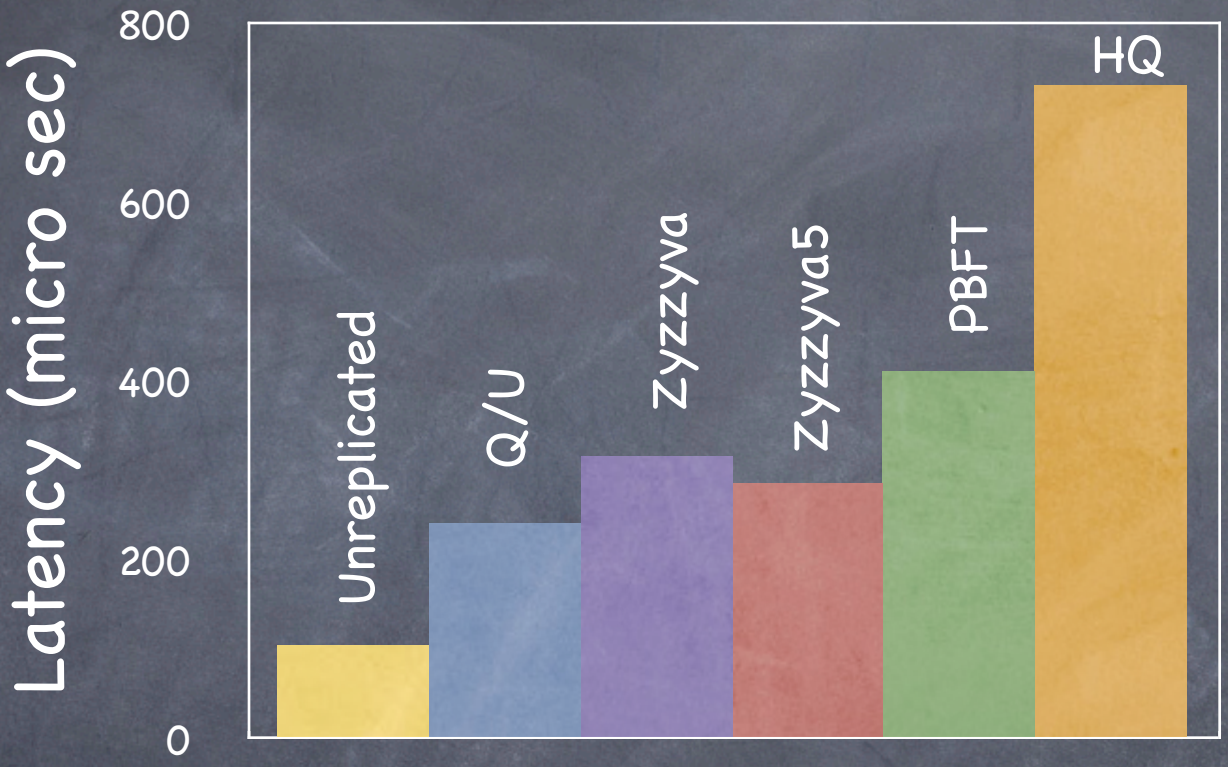
- Zyzzyva provides excellent performance

Latency

	Zyzzyva	Q/U
Replication cost App replicas	2f+1	5f+1
Latency (Updates) Message delays	3	2

- Q/U: Quorum based optimistic approach
  - Latency: 4 or more with request contention

Latency: Best case for Q/U



- Not significant: Q/U is 15% better than Zyzzyva5
  - No request/reply payloads, no contention, update
- Zyzzyva outperforms Q/U: contention, reads, load

Zyzzyva approaches optimal

	Optimal	Zyzzyva
Replication cost Total replicas	3f+1	3f+1
Replication cost App. replicas	2f+1	2f+1
Throughput Overhead: Crypto. ops	2	2+3f/b
Latency Message delays	3	3

- Throughput: Zyzzyva exploits batching
  - Overhead reduces with increasing batch size

Conclusion

Transform high-performance service to high-performance and reliable service

- Zyzzyva: Speculative BFT
  - Performance comparable to unreplicated service

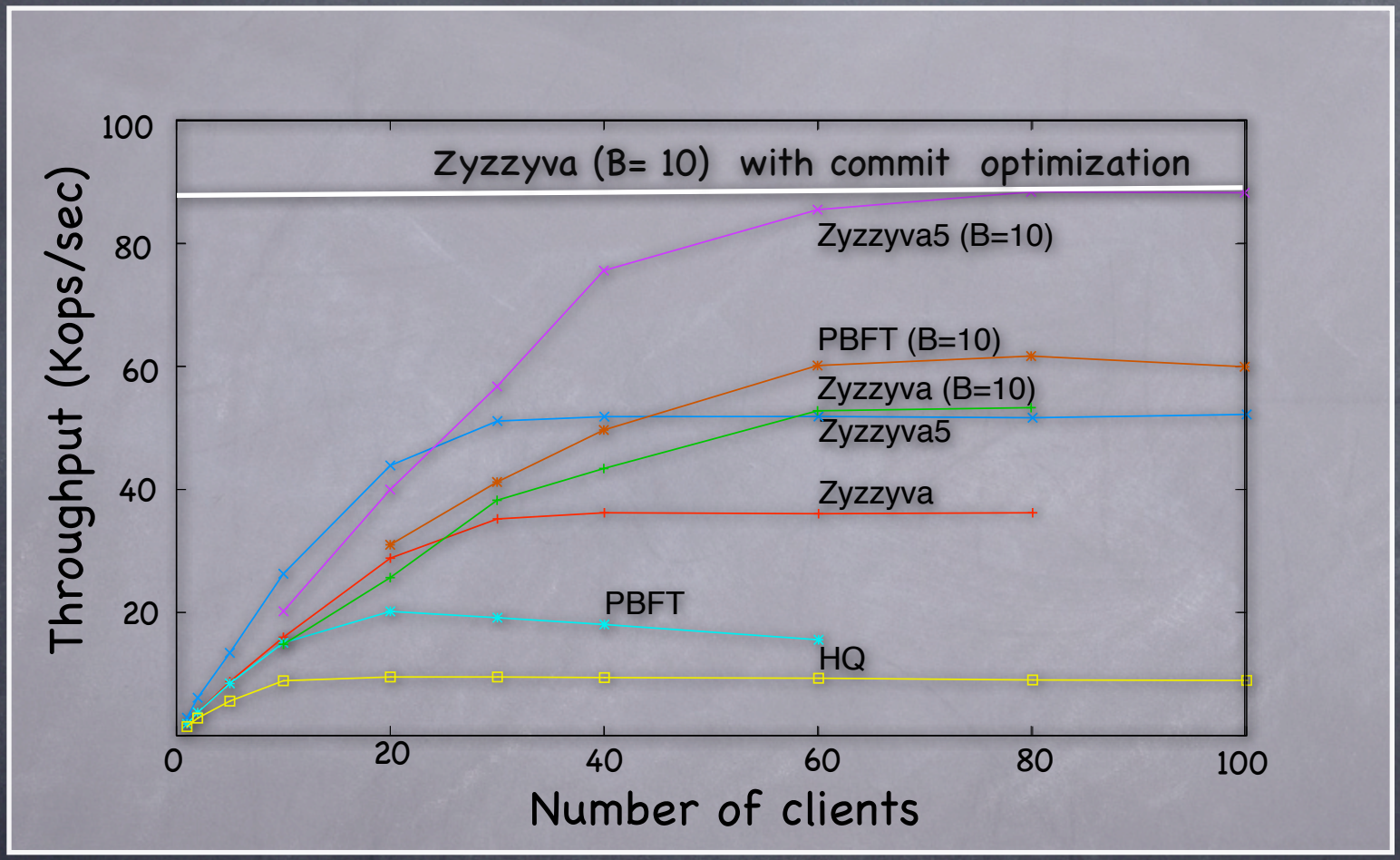
Thank you!

- Acknowledgements:
  - Hewlett-Packard - Travel grant
  - NSF research grants

BACKUP SLIDES

According to dictionary.com, a zyzzyva is "any of various tropical American weevils of the genus Zyzzyva, often destructive to plants."

Throughput: With a faulty backup node



- Failures: Zyzzyva outperforms other protocols
  - Zyzzyva5: 2+(5f+1)/b Zyzzyva(with opt): 2+(5f+1)/b