FLAIR: Accelerating Reads with Consistency-Aware Network Routing

Hatem Takruri, **Ibrahim Kettaneh**, Ahmed Alquraan, Samer Al-Kiswany

UNIVERSITY OF WATERLOO

Introduction

Modern cloud applications

- Are read intensive
  - R:W in Google's F1 advertising system is 380:1 [1]
  - R:W in Facebook's TAO is 500:1 [2]
- Require data reliability

Main approach: Replication

Strongly consistent replication protocols are popular

[1] J. Shute, R. Vingralek, B. Samwel, et al., F1: a distributed SQL database that scales. Proc. VLDB Endow., 2013. 6(11): p. 1068-1079.

[2] N. Bronson, Z. Amsden, G. Cabrera, et al. TAO: Facebook's distributed data store for the social graph. in Proceedings of the USENIX Technical Conference. 2013. San Jose, CA: USENIX

Inefficiency of current replication protocols

Modern strongly consistent protocols are inefficient for read-heavy workloads

Main reason: they are leader-based

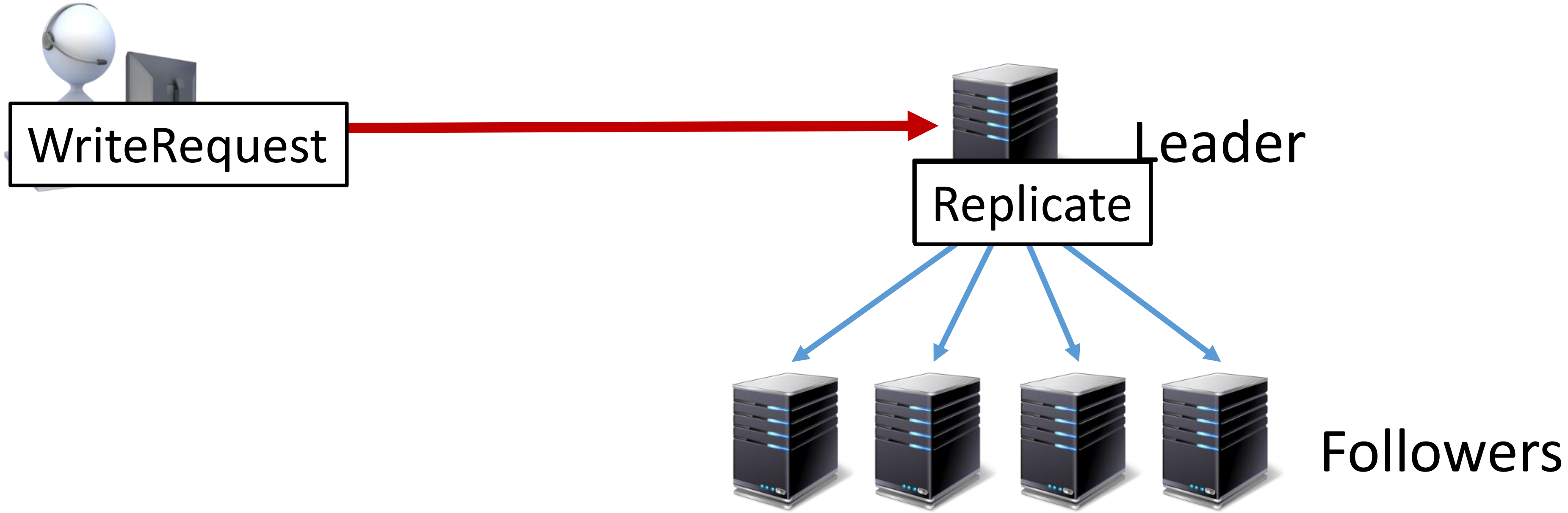
Raft

ZAB

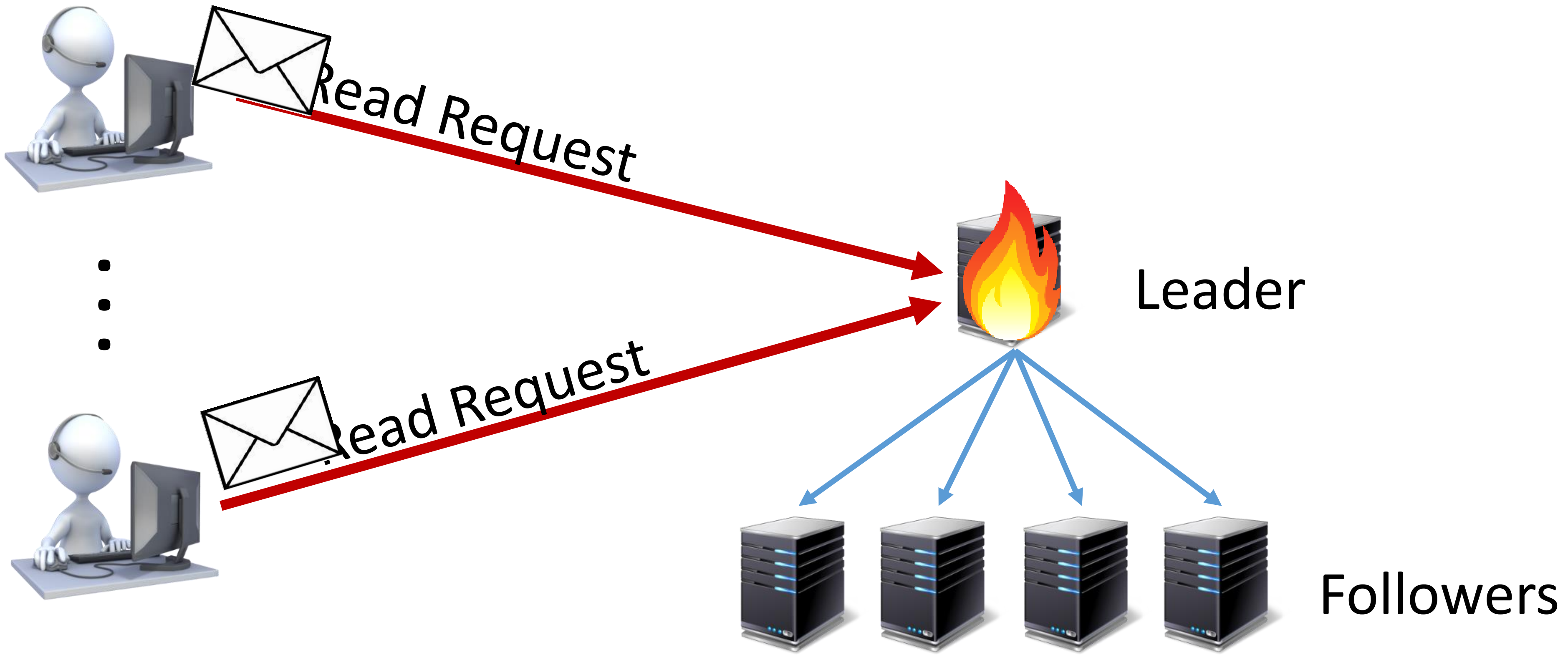
Paxos

Viewstamp Replication

Leader-based Consensus Inefficiency



Leader-based Consensus Inefficiency



Inefficient: Only the leader handles read/write requests

Missed opportunity: Utilize the followers to serve reads.

Current Approaches to Utilizing Followers

Read leases

- The leader grants read leases to followers
- With a valid lease: followers serve reads
- On write: leader revokes leases
- Drawbacks
    - Complicates lease management
    - Increases write latency
    - Complicates fault tolerance

Eventual consistency

- Replicas serve reads, albeit the possibility of returning stale data

FLAIR: Fast, Linearizable, Network Accelerated Client Reads

A novel approach to serve reads from followers while maintaining linearizability.

A shim layer atop current leader-based protocols.

Main idea

- The network detects read/write conflicts, and
- Load balance reads across consistent replicas

Enabler: Programmable switches

FLAIR in a Nutshell

- Network switches monitor write requests/responses
- Identify which objects are stable, and on which replicas
- Load balance reads across consistent replicas

FLAIR is an in-network consistency-aware load-balancing protocol

Evaluation Summary

- Implemented FLAIR using P4
- Evaluated FLAIR on a cluster with Barefoot Tofino switch

FLAIR achieves

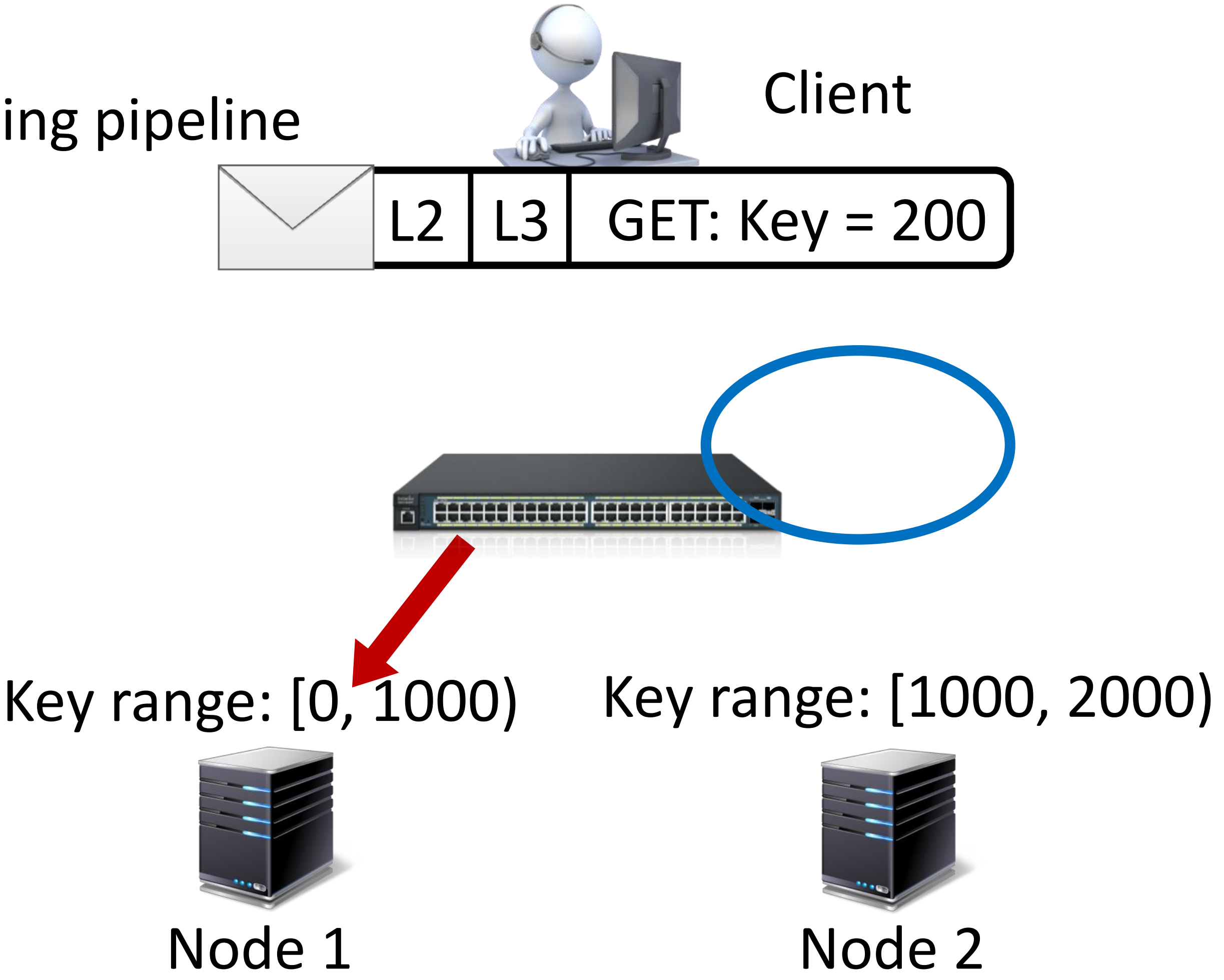
- 1.4× to 2.1× higher throughput
- 1.5× to 2.4× lower latency

Outline

- Overview of programmable switches
- FLAIR design
- Implementation
- Evaluation

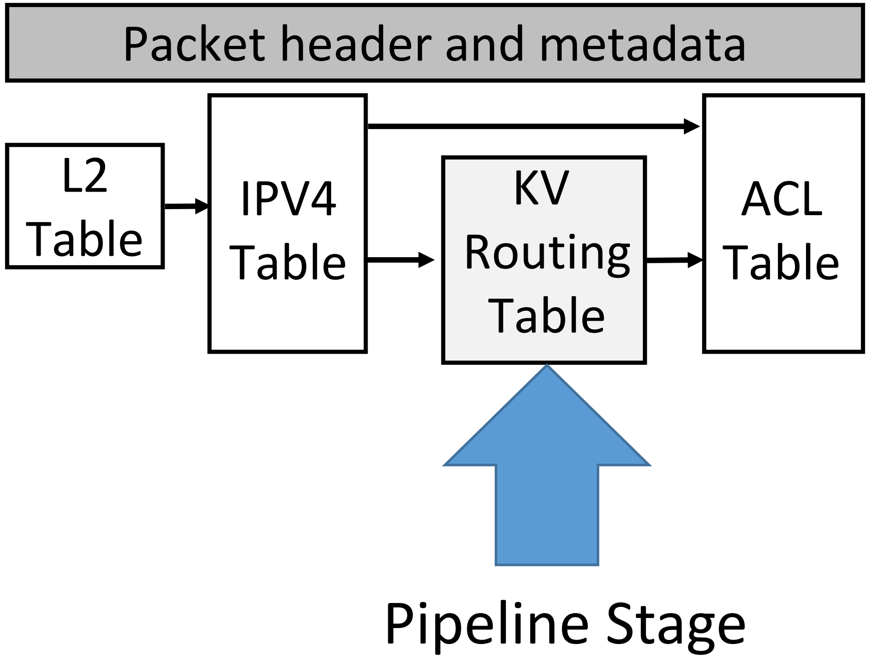
Programmable Switches Overview

Example:
key-based routing pipeline

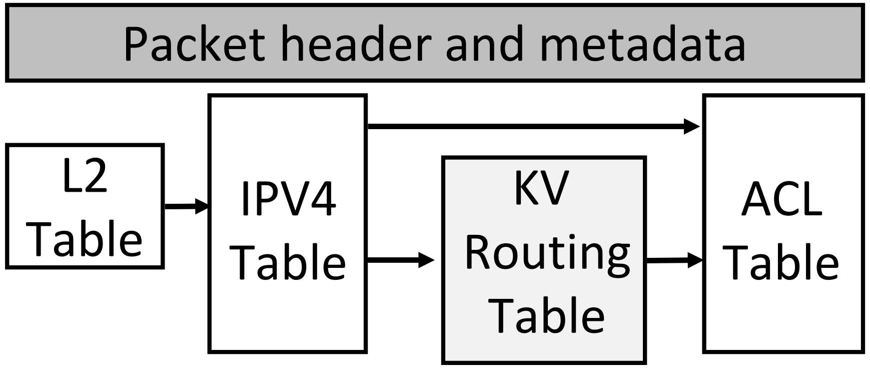


Programmable Switches Overview

Example:
key-based routing pipeline



Programmable Switches Overview



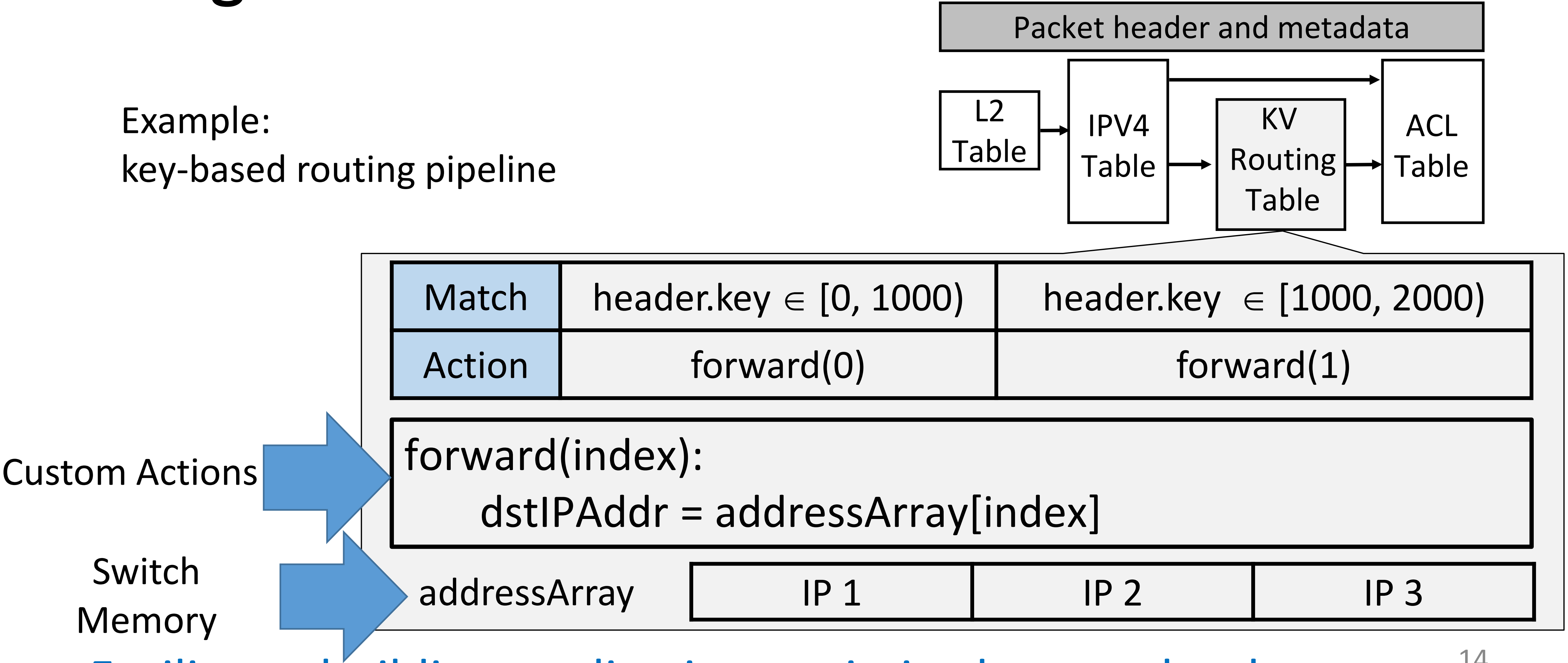
Example:
key-based routing pipeline

Match + Action Tables

Match	header.key $\in$ [0, 1000)	header.key $\in$ [1000, 2000)
Action	forward(0)	forward(1)

Programmable Switches Overview

Example:
key-based routing pipeline



Facilitates building application-optimized network substrate.

Programmable Switches Challenges

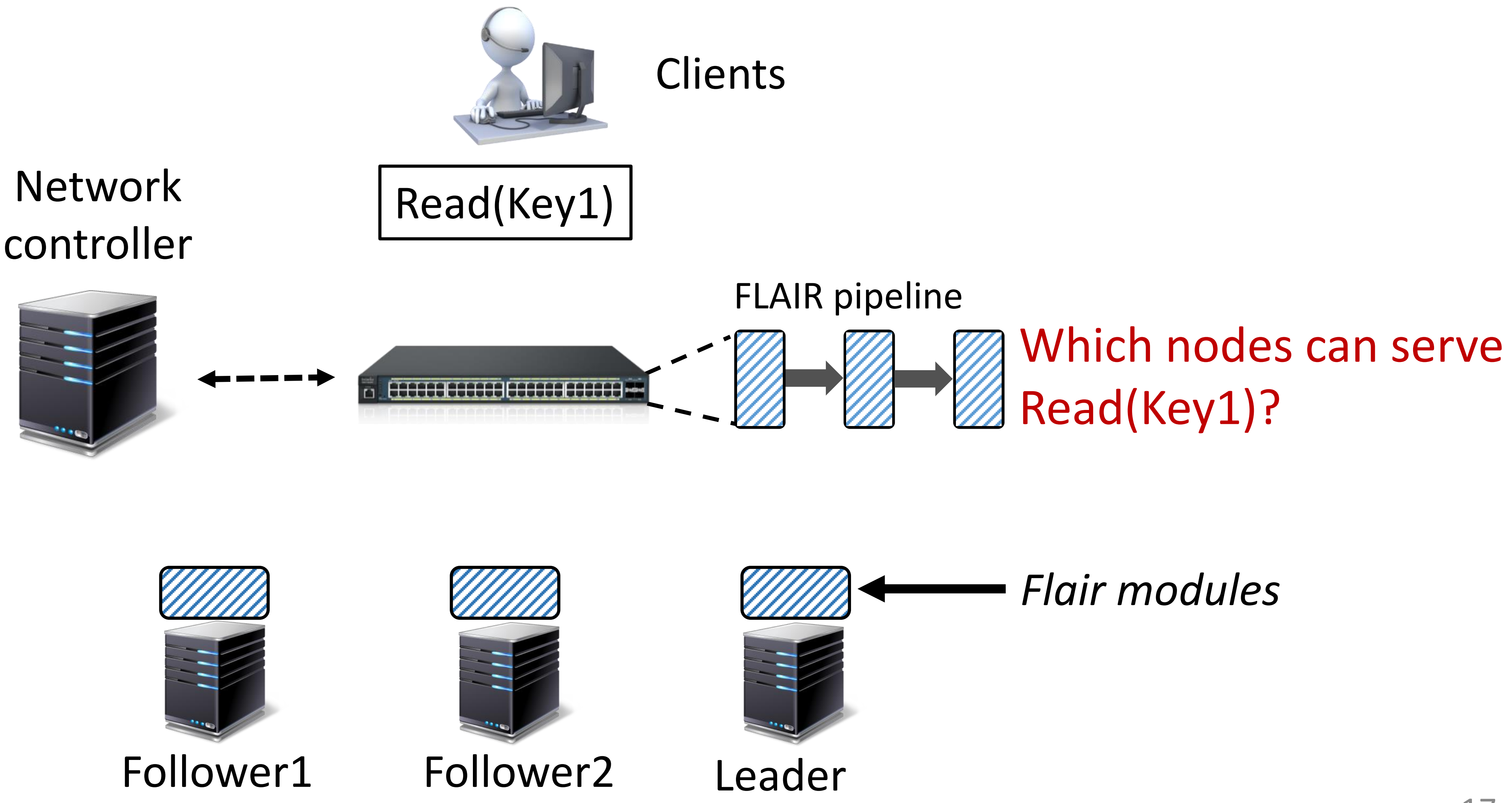
- No loops or recursion
- Restricted pipeline-based programming model
- Limited number of pipeline stages
- Limited computational power
- Restricted memory access model

Can we use programmable switches to build consistency-aware network routing protocol?

Outline

- Overview of programmable switches
- FLAIR design
- Implementation
- Evaluation

FLAIR Design



FLAIR's Object Stability Array

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Unstable	…
Stable Replicas	L, F1	All	-	…

FLAIR in Action

Read(KeyHash = 5000)

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	All	All	…

Key = 5000
Value = A

Leader

Key = 5000
Value = A

Follower 1

Key = 5000
Value = A

Follower 2

FLAIR in Action

Read(KeyHash = 5000)

Check

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	All	All	…

Key = 5000
Value = A

Leader

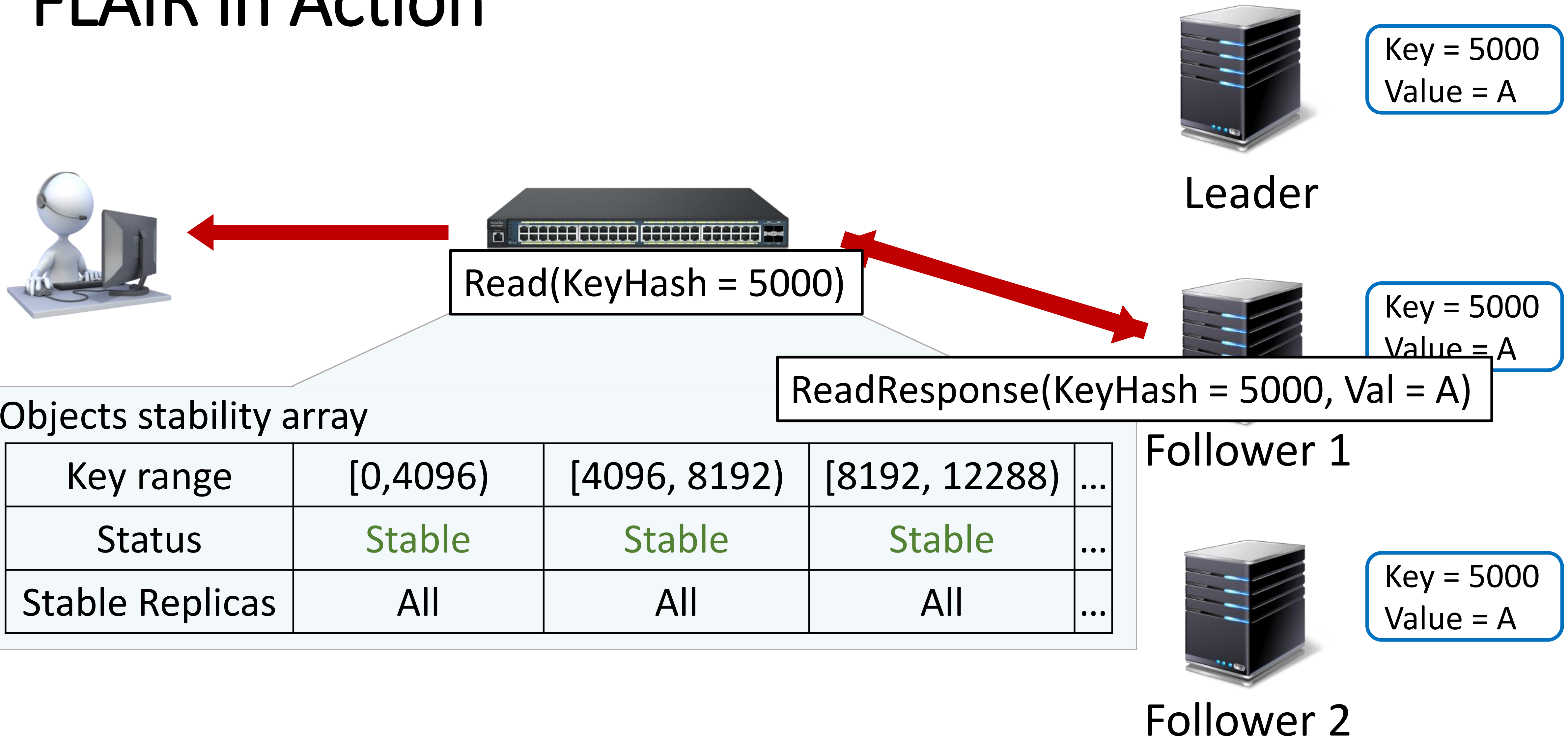
Key = 5000
Value = A

Follower 1

Key = 5000
Value = A

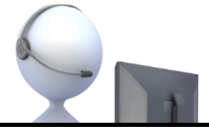
Follower 2

FLAIR in Action



Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	All	All	…

FLAIR in Action



Write(KeyHash = 5000, Val = B)

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	All	All	…

Key = 5000
Value = A

Leader

Key = 5000
Value = A

Follower 1

Key = 5000
Value = A

Follower 2

FLAIR in Action

Write(KeyHash = 5000, Val = B)

Update

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…

Key = 5000
Value = A

Leader

Key = 5000
Value = A

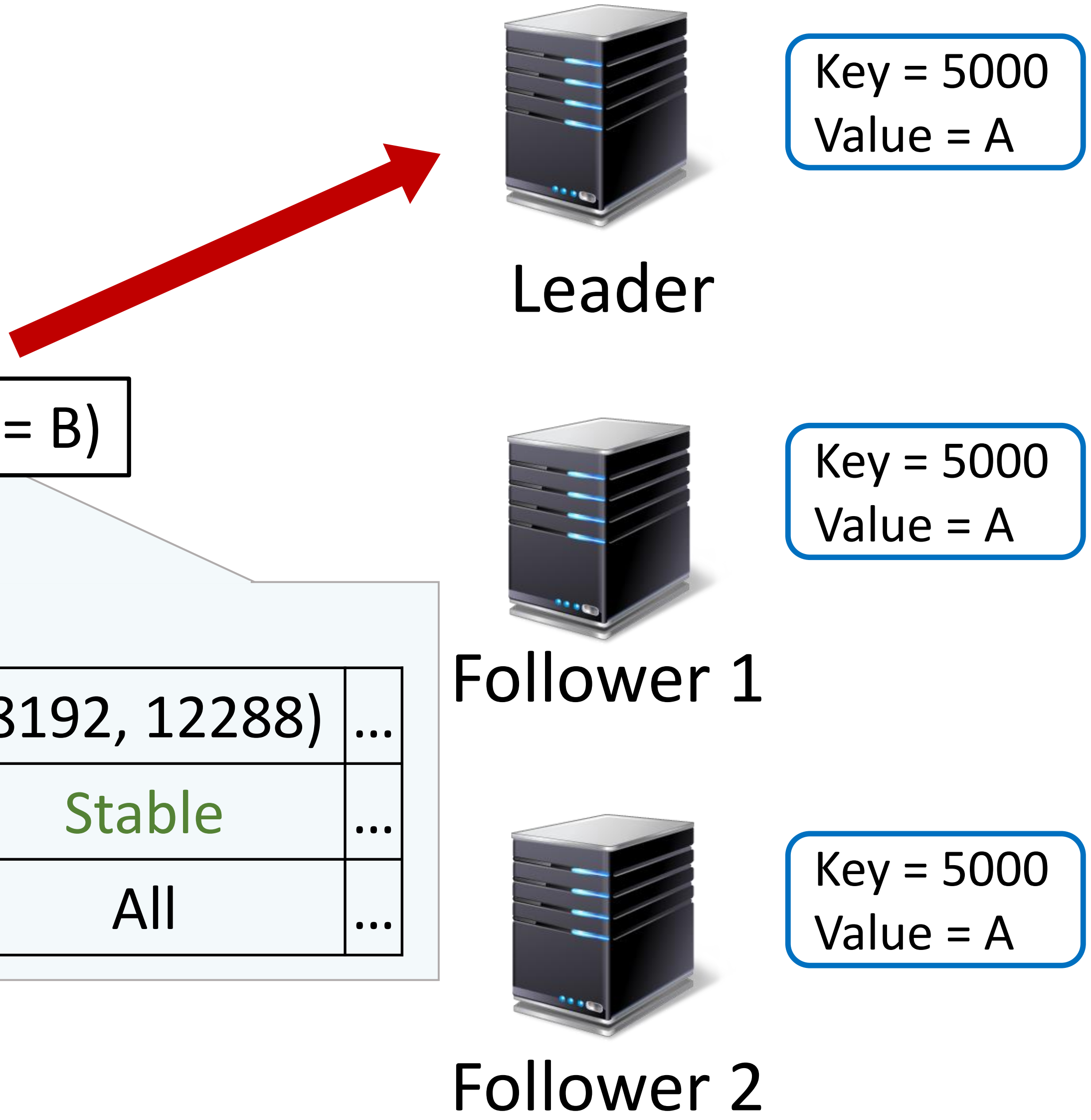
Follower 1

Key = 5000
Value = A

Follower 2

FLAIR in Action

Write(KeyHash = 5000, Val = B)



Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…

FLAIR in Action

Write(KeyHash = 5000, Val = B)

Key = 5000
Value = A

Leader

Key = 5000
Value = A

Follower 1

Key = 5000
Value = A

Follower 2

Read(KeyHash = 5000)

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…

FLAIR in Action

Write(KeyHash = 5000, Val = B)

Leader

Key = 5000
Value = A

Follower 1

Key = 5000
Value = A

Follower 2

Key = 5000
Value = A

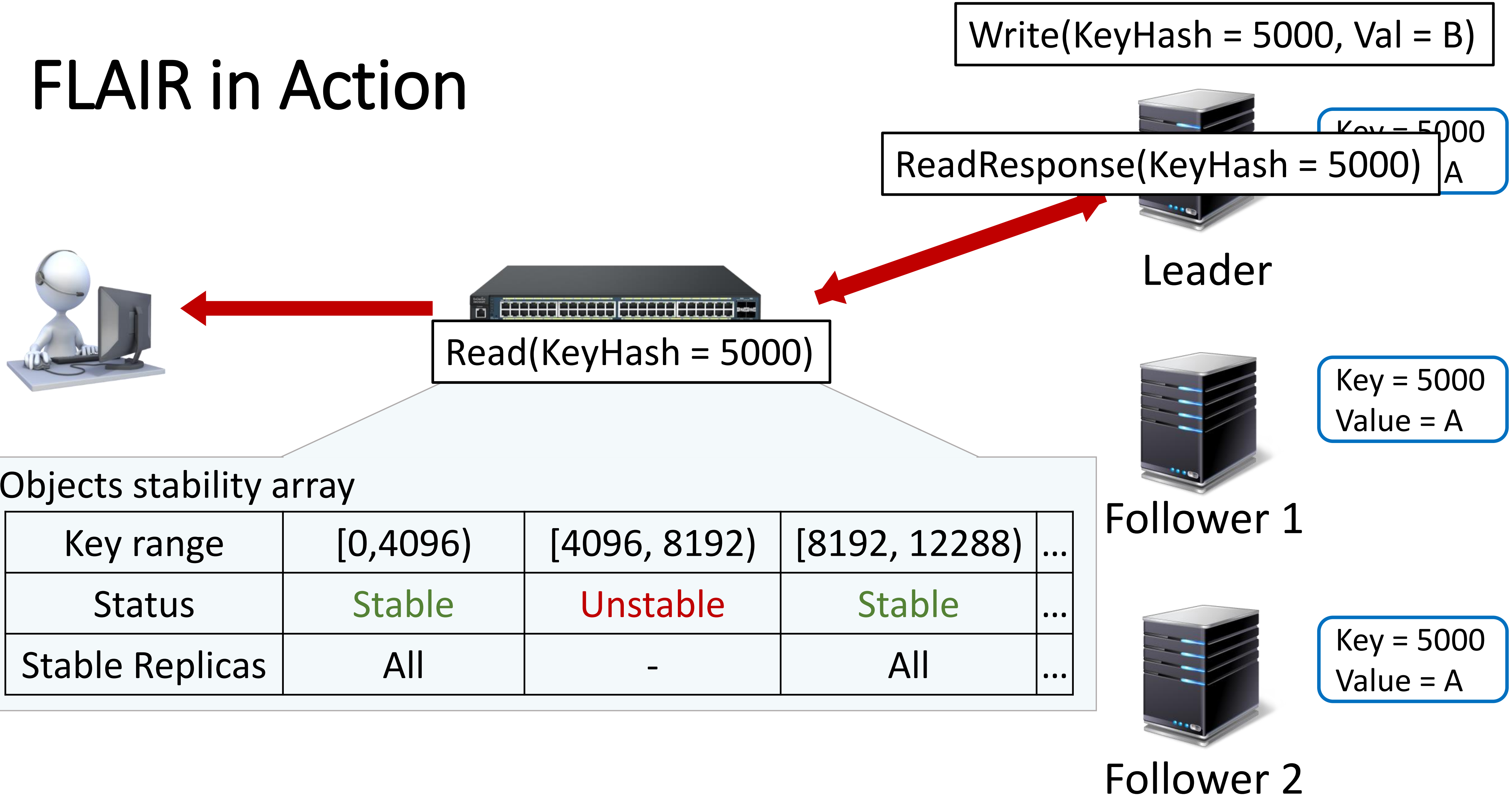
Read(KeyHash = 5000)

Check

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…

FLAIR in Action



Objects stability array

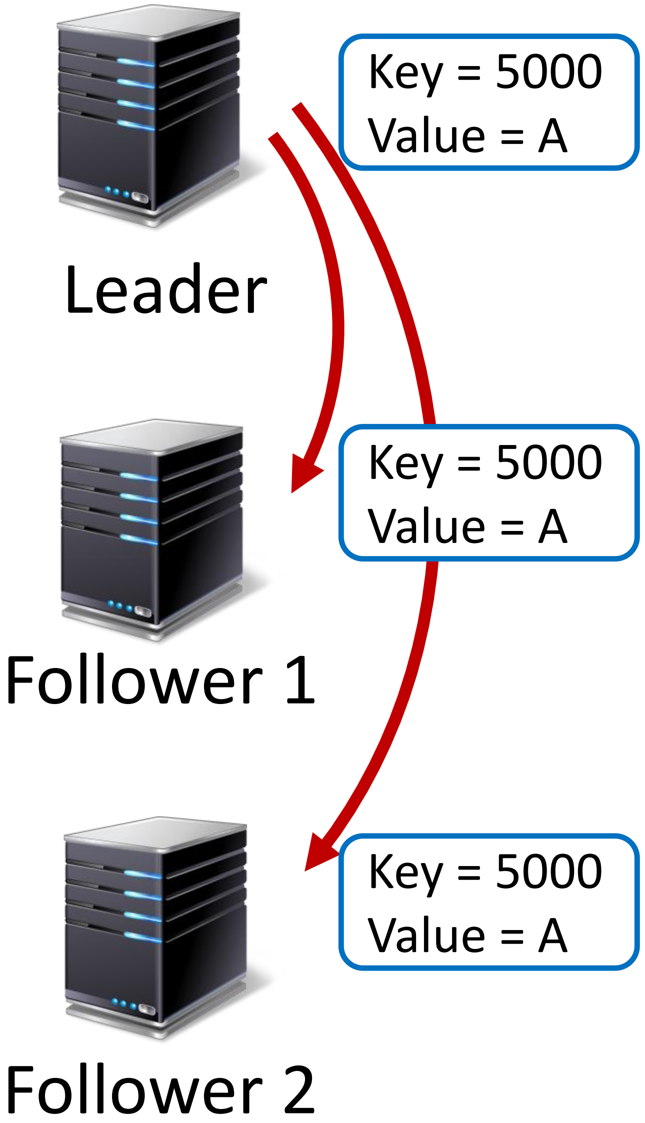
Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…

FLAIR in Action

Write(KeyHash = 5000, Val = B)

Objects stability array

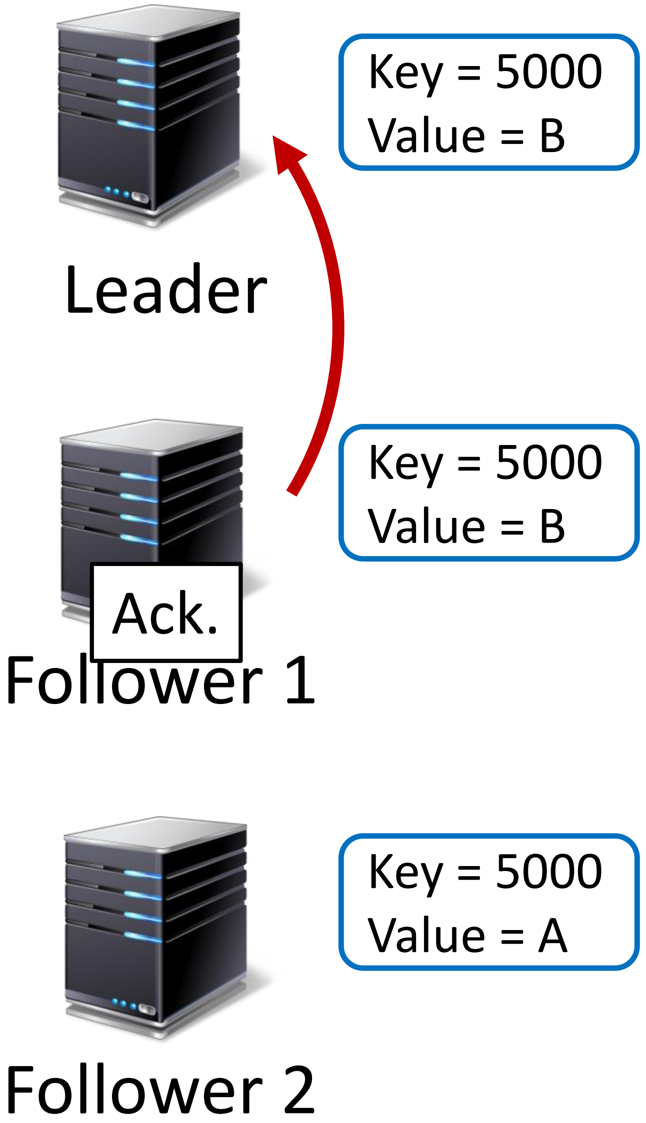
Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…



FLAIR in Action

Objects stability array

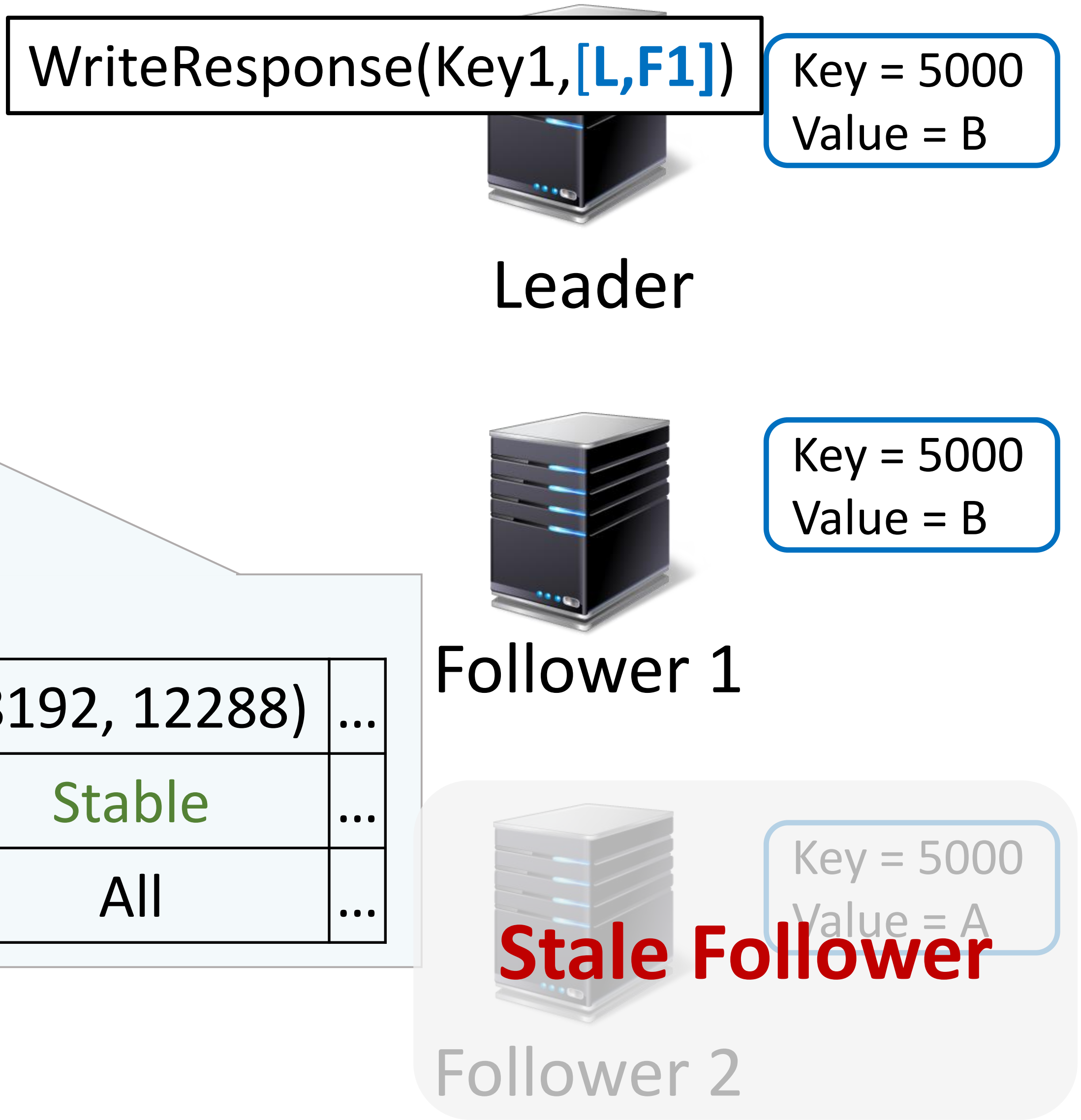
Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…



FLAIR in Action

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Unstable	Stable	…
Stable Replicas	All	-	All	…



FLAIR in Action

WriteResponse(Key1,**[L,F1]**)

Update

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	L, F1	All	…

Key = 5000
Value = B

Leader

Key = 5000
Value = B

Follower 1

Key = 5000
Value = A

Stale Follower

Follower 2

FLAIR in Action

WriteResponse(Key1,**[L,F1]**)

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	L, F1	All	…

FLAIR in Action

Read(KeyHash = 5000)

Check

Objects stability array

Key range	[0,4096)	[4096, 8192)	[8192, 12288)	…
Status	Stable	Stable	Stable	…
Stable Replicas	All	L, F1	All	…

Key = 5000
Value = B

Leader

Key = 5000
Value = B

Follower 1

Key = 5000
Value = A

Stale Follower

Follower 2

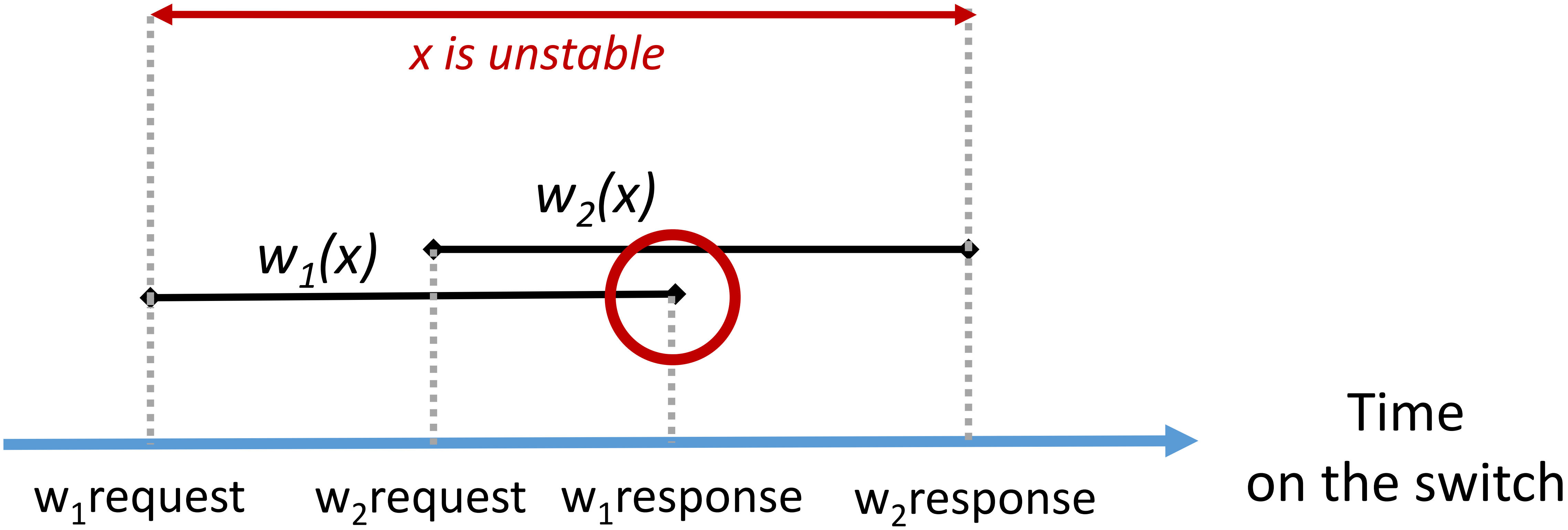
FLAIR Design

- Concurrent writes to the same object
- Packet reordering
- Failures
  - Switch failure
  - Leader failure
  - Follower failure
  - Network partitioning

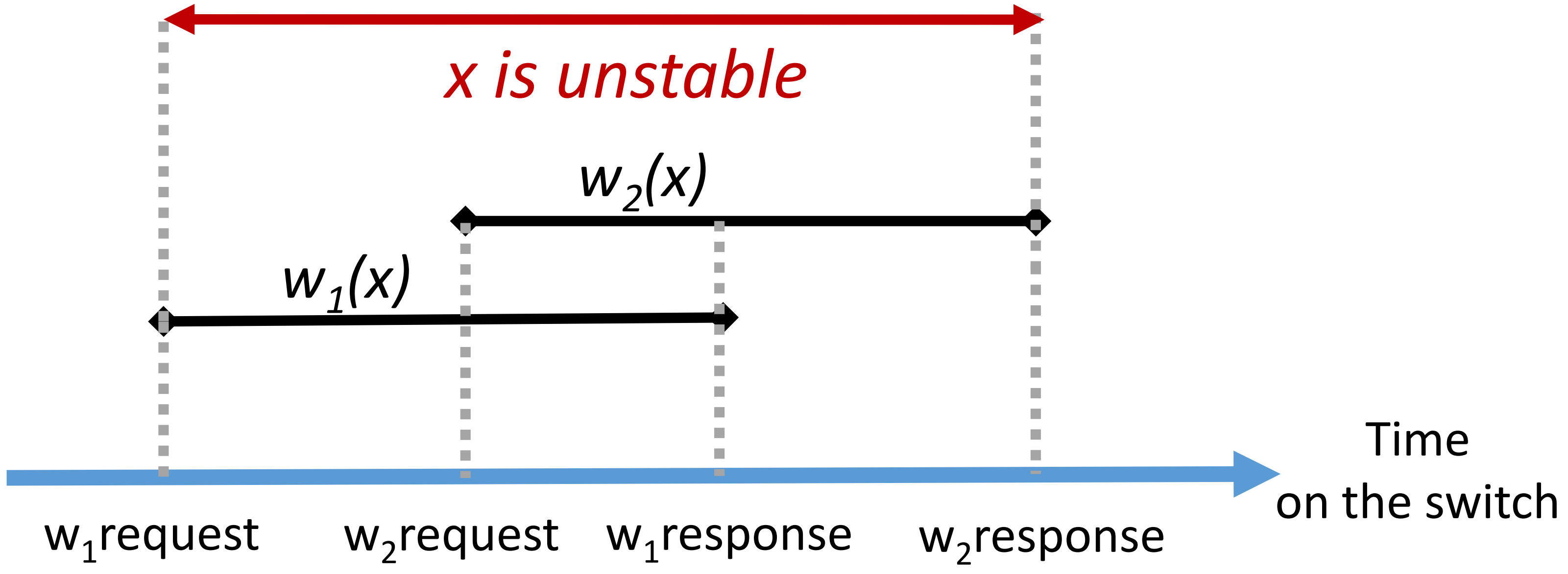
Protocol validation

  - Detailed proof
  - TLA+ model checking

Concurrent Writes

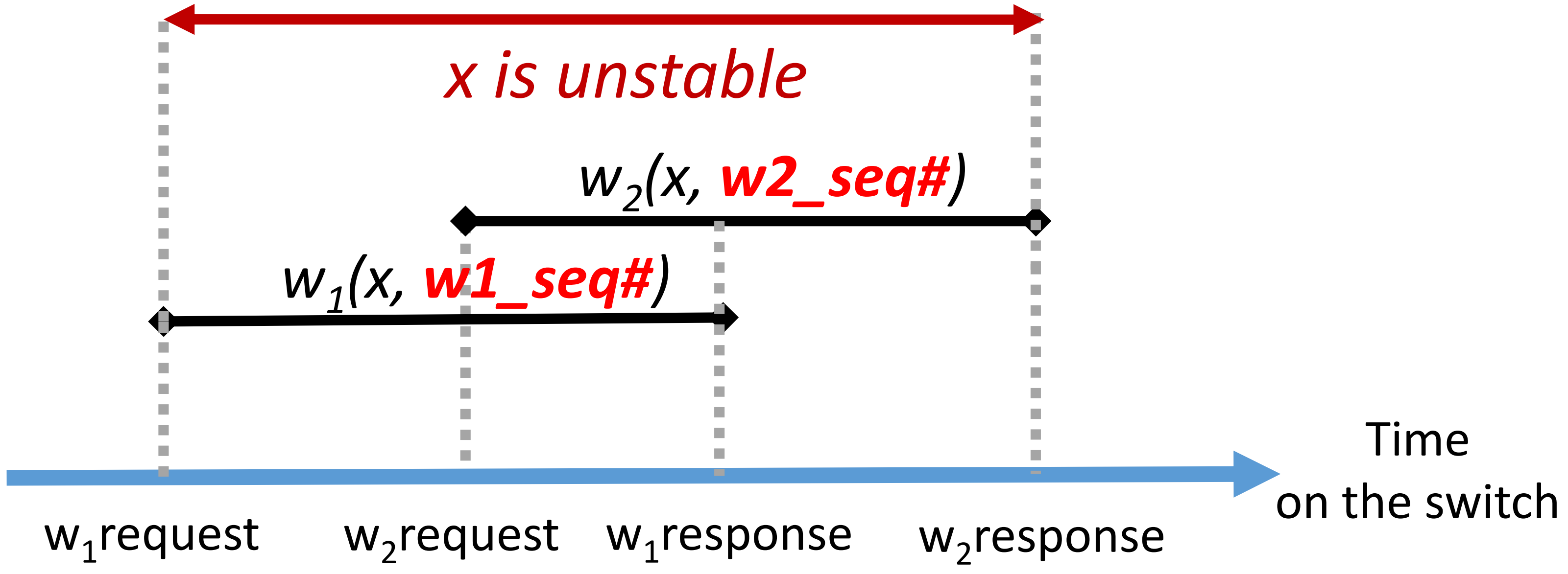


Concurrent Writes



Concurrent Writes

- Every write gets a unique sequence number
- Objects stability array stores the sequence number of the last write

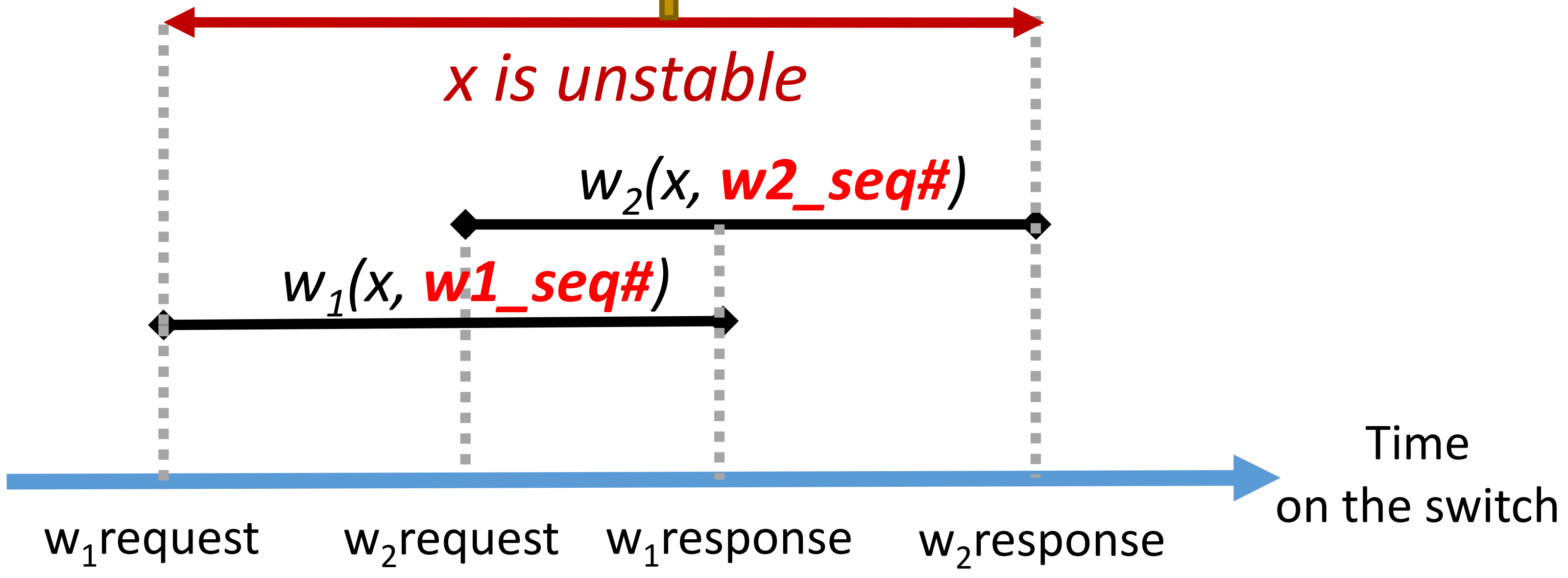


Concurrent Writes

- Every write gets a unique sequence number
- Objects stability array stores the sequence number of the last write
- Objects remain unstable until last sequence number is acknowledged

unstable objects array

Hash range	[0,4096)	…
Status	Unstable	…
Stable Replicas	-	…
Seq#	***w2_seq#***	…



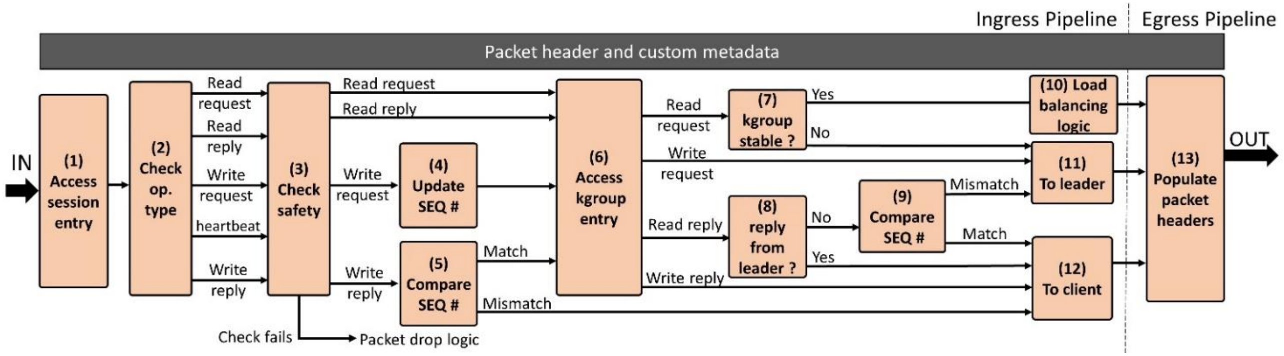
Outline

- Overview of programmable switches
- FLAIR architecture
- Implementation
- Evaluation

Implementation

FlairKV is a key-value store that optimizes Raft using FLAIR

- Implemented using P4 language
- Utilizes 12 registers and 30 tables (only 5% of ASIC memory)
- Implemented consistency-aware load balancing techniques
  - Random
  - Leader avoidance
  - Follower load awareness



Evaluation

Alternatives

- Leader-based (Raft, VR)
- Leader lease (Opt. Raft)
- Unreplicated
- Fast Paxos
- Follower-leases (Fleases)

Metrics

- Throughout
- Latency
- Load balancing efficiency

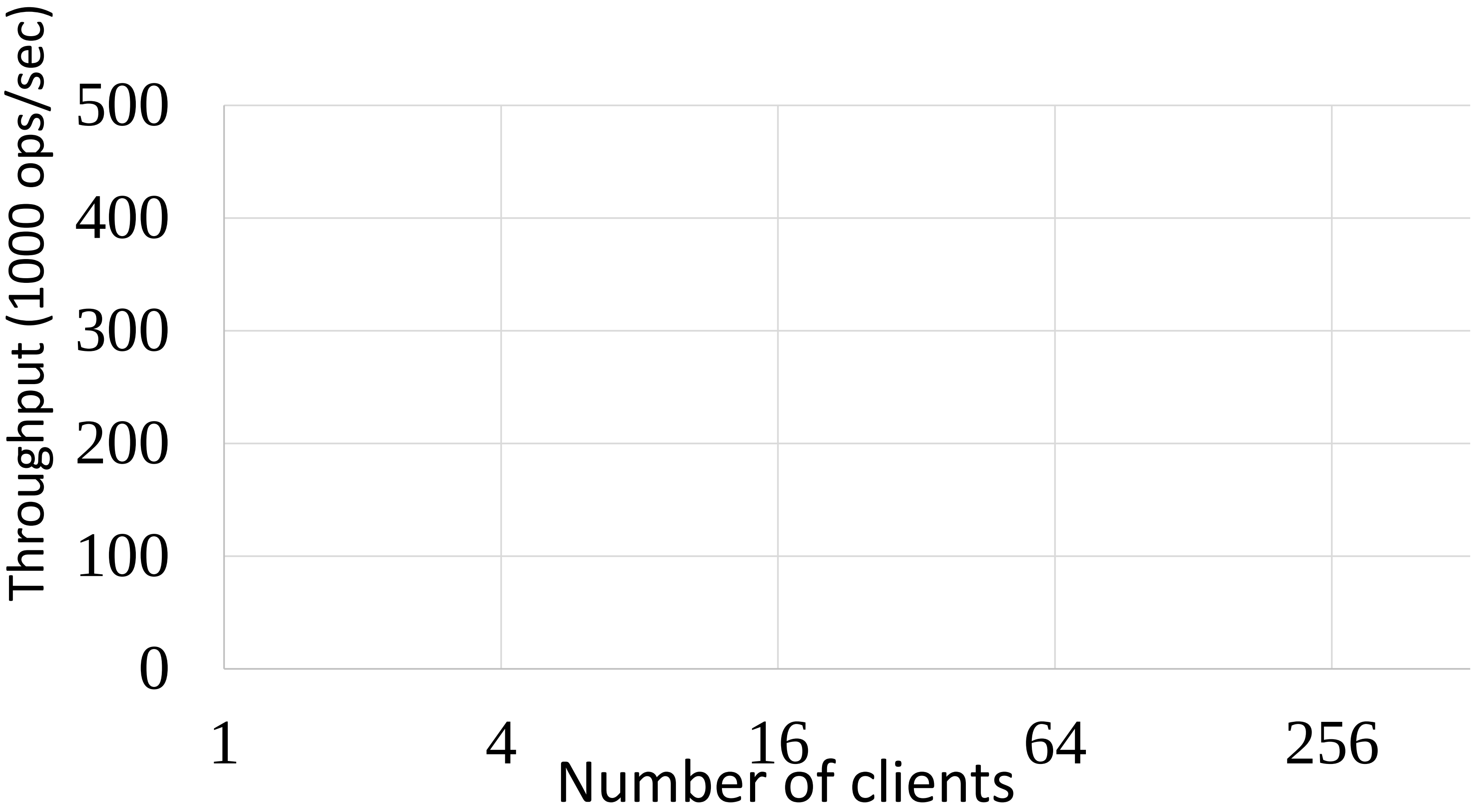
We varied:

- Number of replicas
- Number of clients
- Read to write ratio
- Workload skewness
- Data set size

Throughput

Workload
YCSB-B (95% reads)

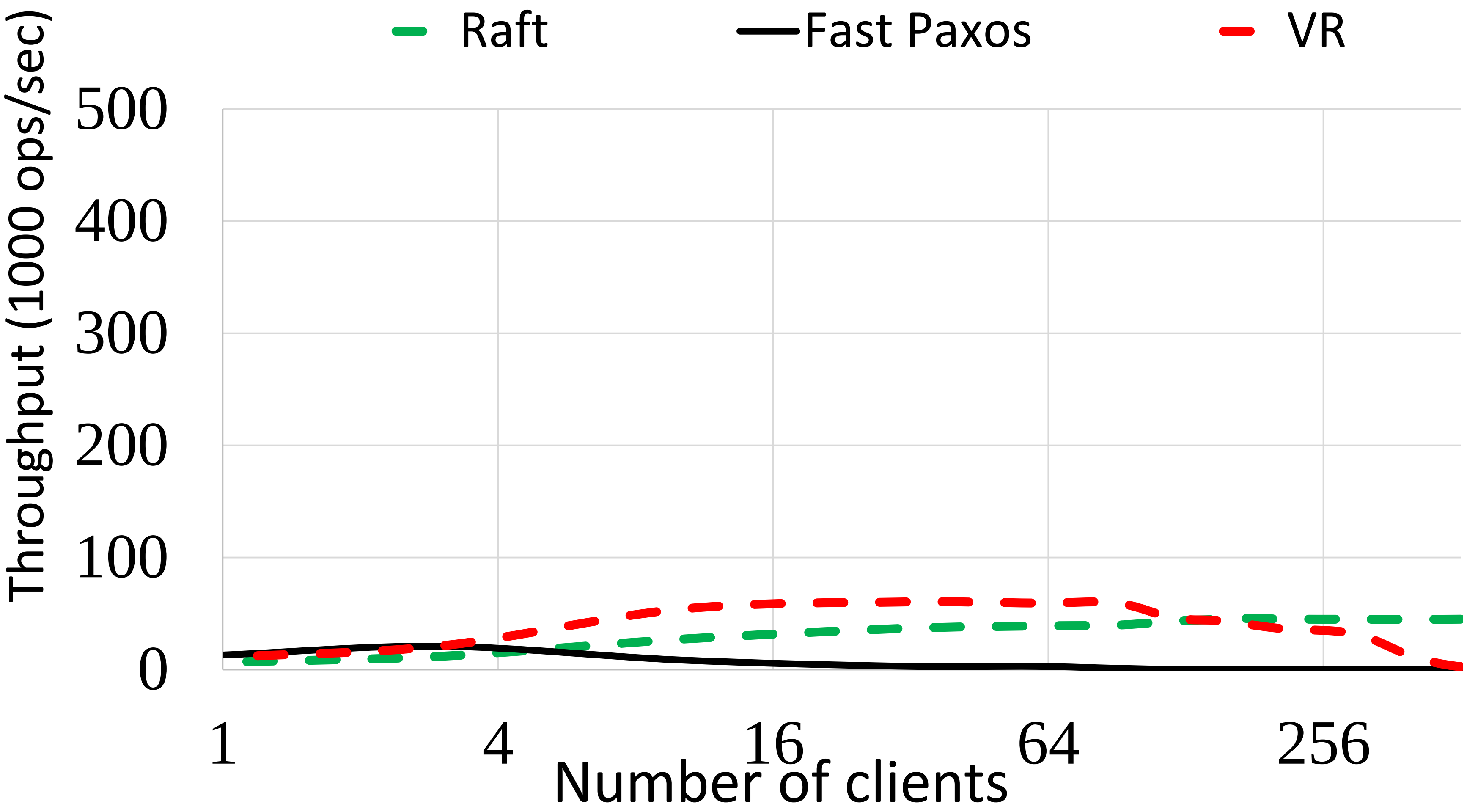
Uniform workload



Throughput

Workload
YCSB-B (95% reads)

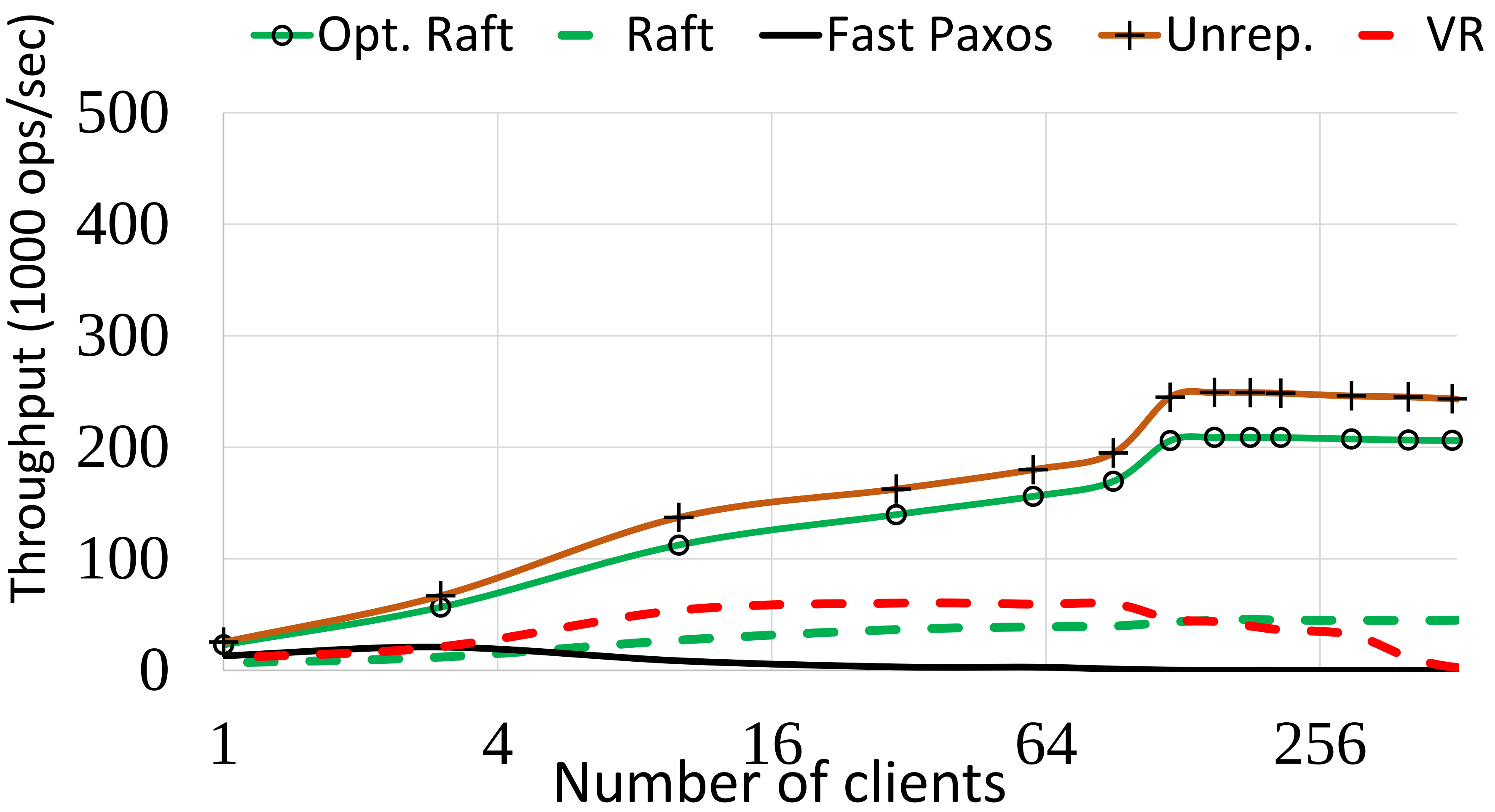
Uniform workload



Throughput

Workload
YCSB-B (95% reads)

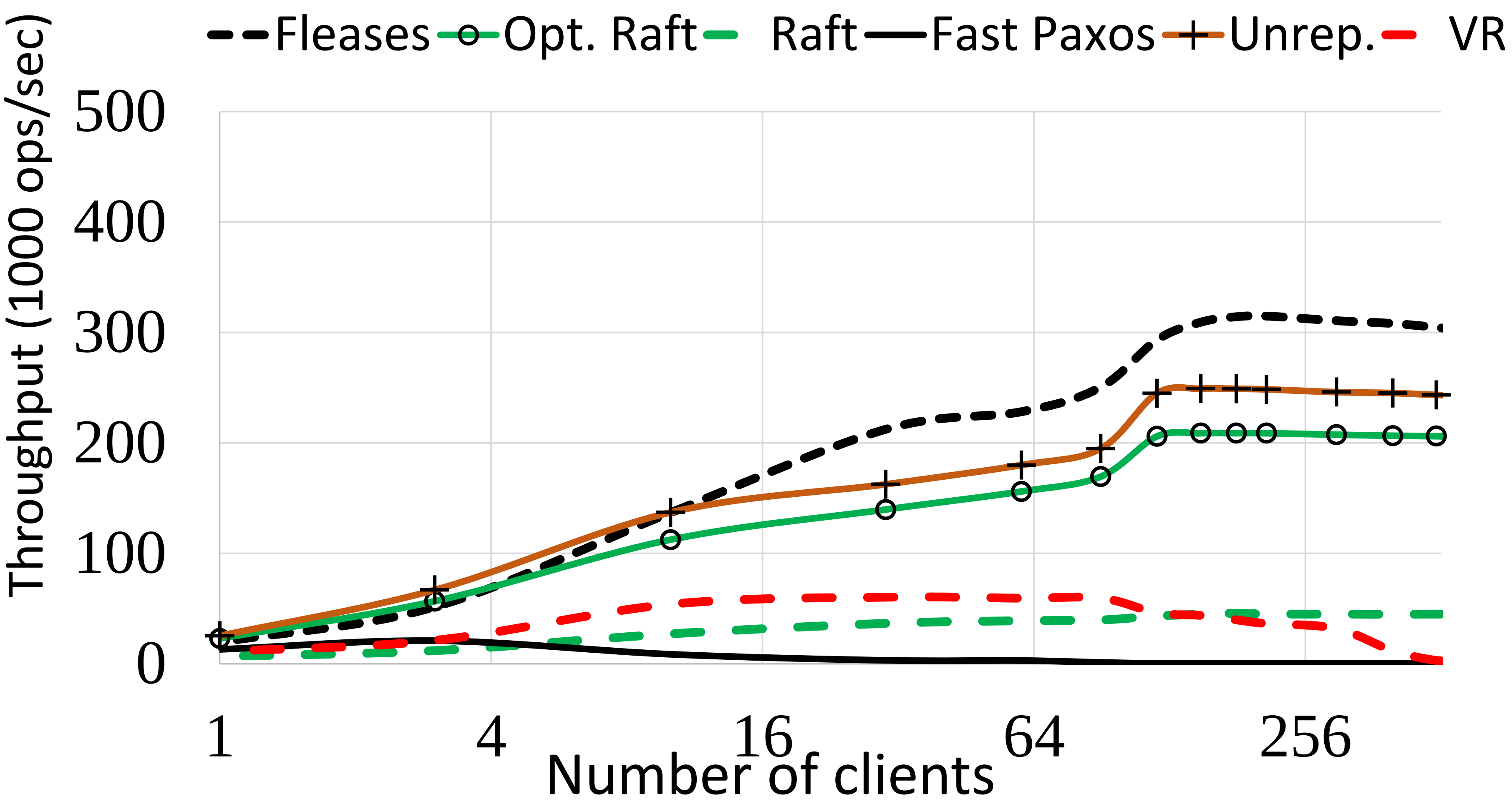
Uniform workload



Throughput

Workload
YCSB-B (95% reads)

Uniform workload

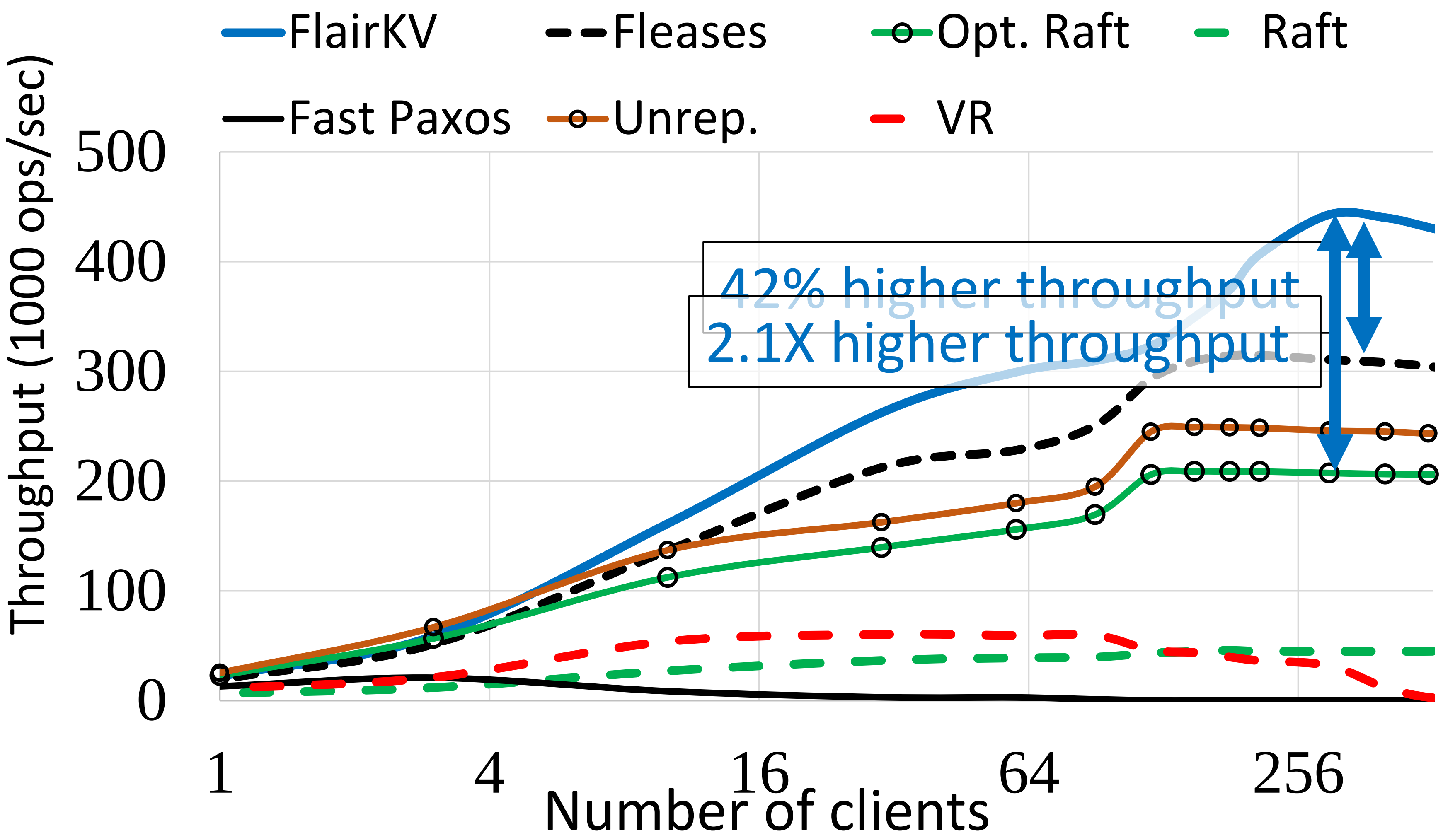


Throughput

Workload

YCSB-B (95% reads)

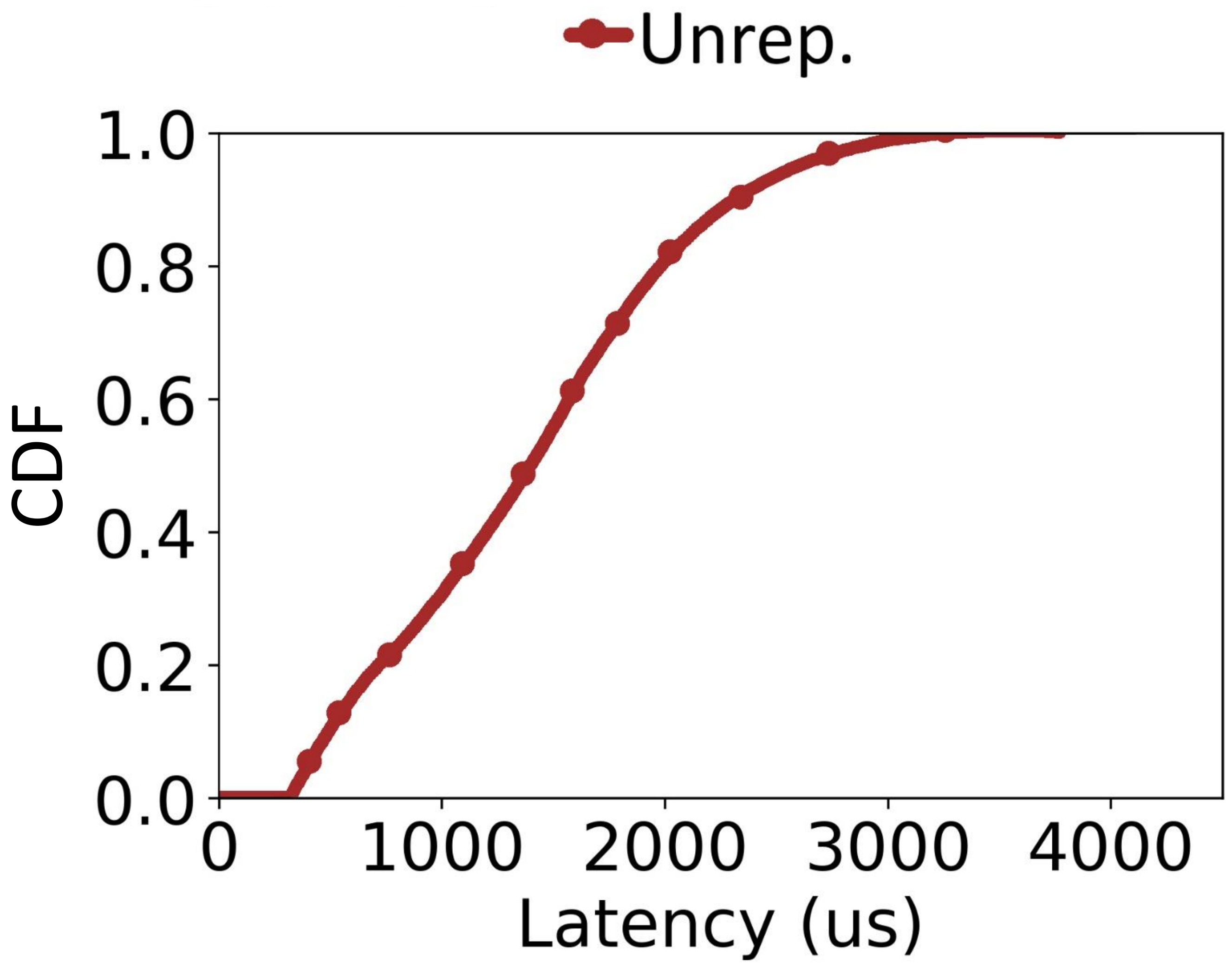
Uniform workload



Read Latency

Workload
YCSB-B (95% reads)

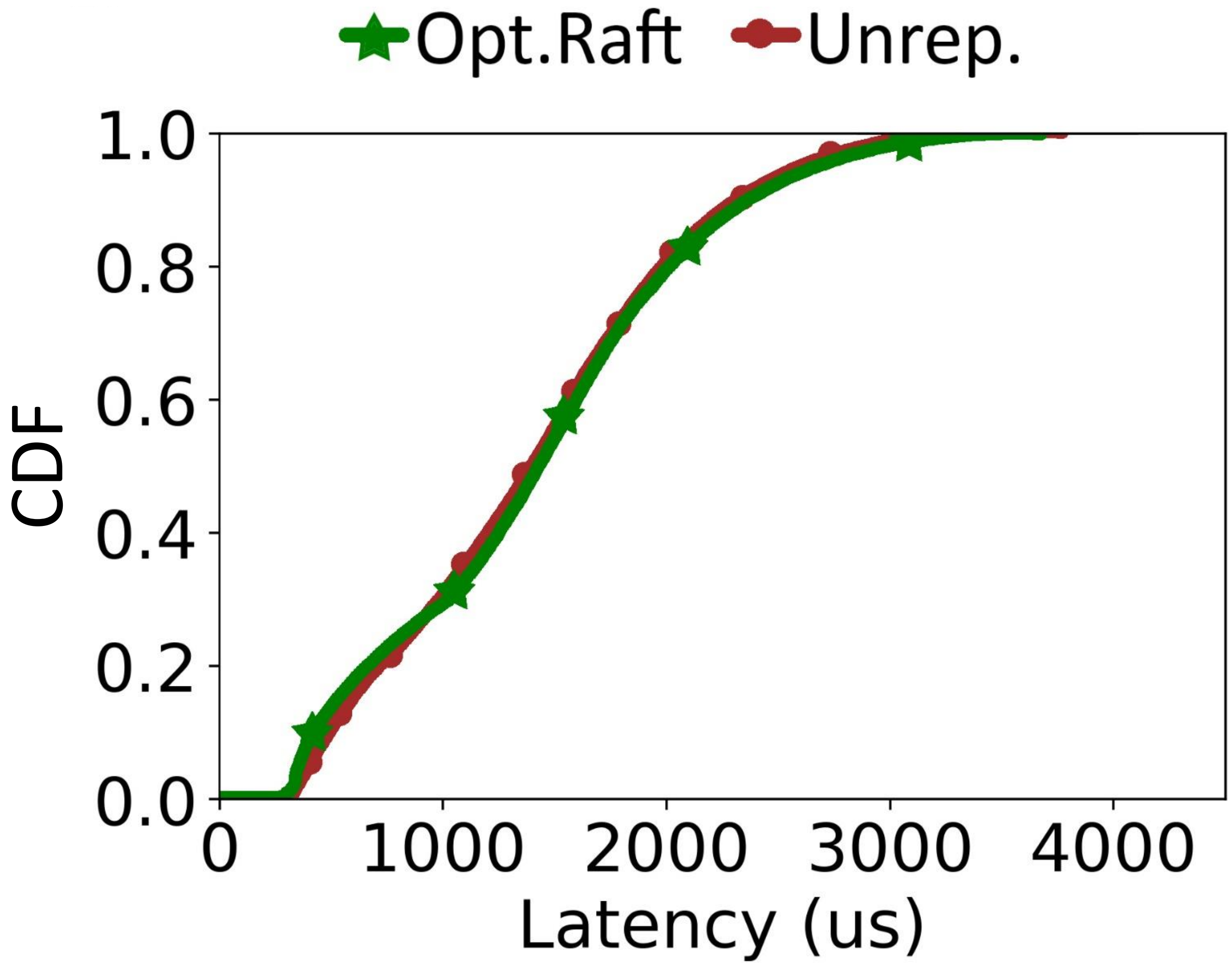
Uniform workload



Read Latency

Workload
YCSB-B (95% reads)

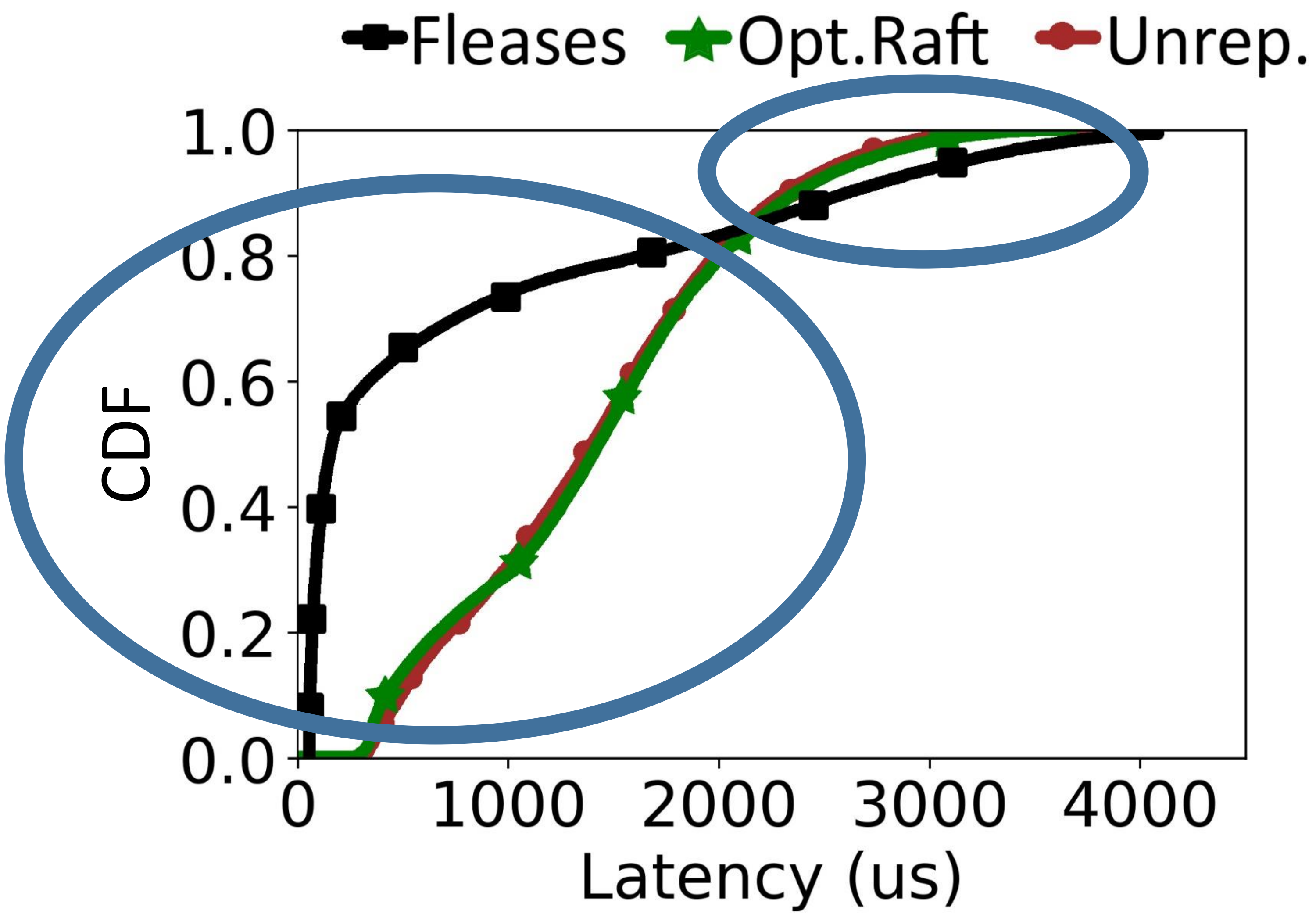
Uniform workload



Read Latency

Workload
YCSB-B (95% reads)

Uniform workload

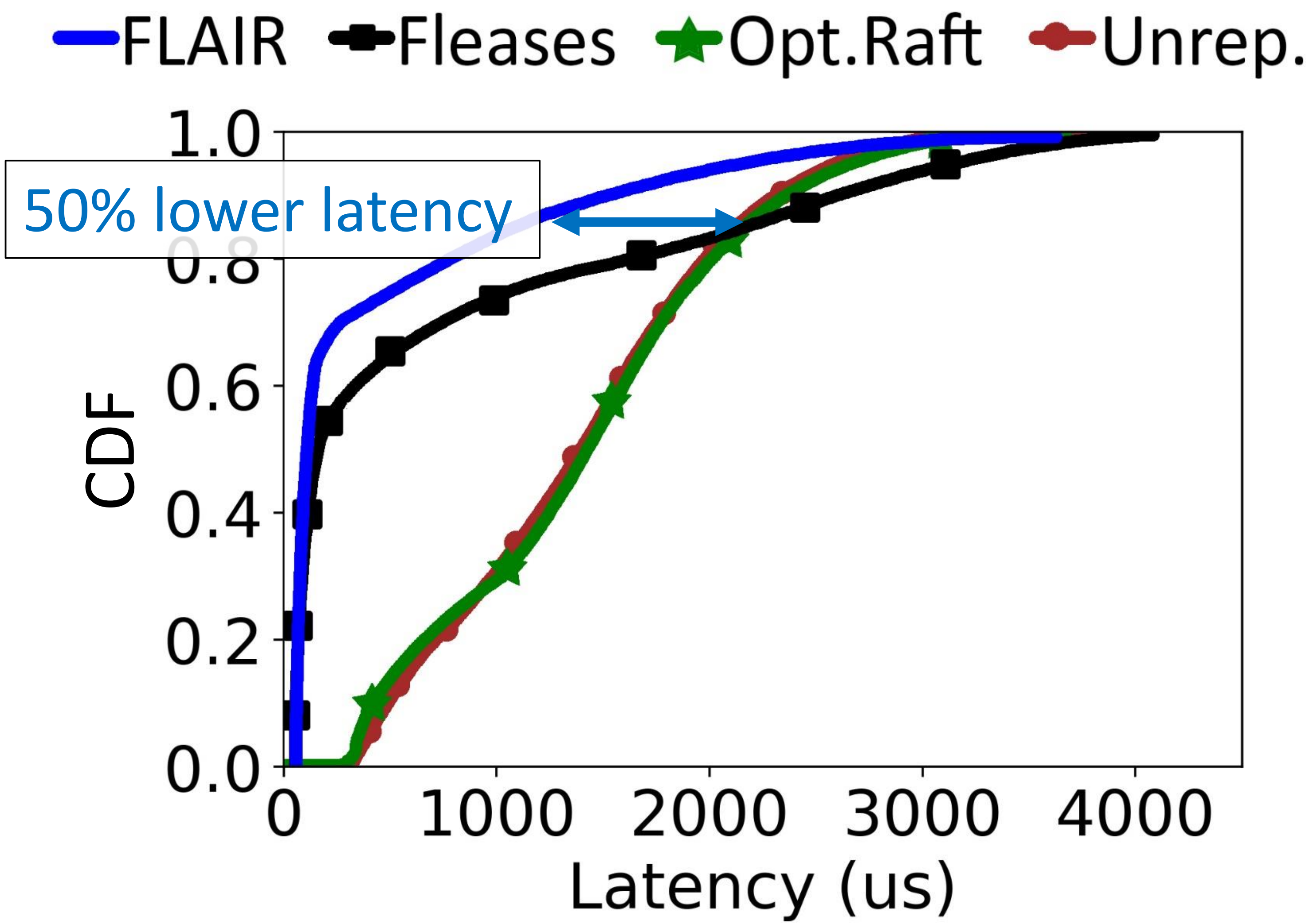


Read Latency

Workload
YCSB-B (95% reads)

Uniform workload

FLAIR reduces latency
- Avoid inconsistent replicas
- Avoid the leader



Conclusion

- FLAIR a shim layer atop leader-based consensus protocol
  - Exploits programmable switches
  - Builds in-network consistency-aware load balancing
  - Maintains linearizability
- FlairKV achieves
  - 2.1× higher throughput than classical approaches, and up to 1.4× than leases
  - 2.4× lower latency than classical approaches, and up to 1.5× than leases

Despite their limitations, programmable switches can be leveraged to accelerate complex system protocols.

FLAIR project: https://wasl.uwaterloo.ca/projects/flair/