# Adore: Atomic Distributed Objects with Certified Reconfiguration

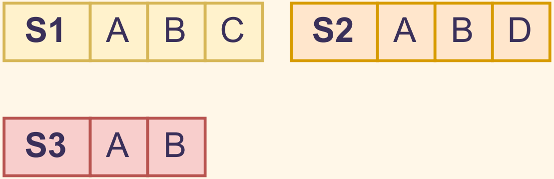
**Wolf Honoré**[1] Ji-Yong Shin[2] Jieung Kim[1] Zhong Shao[1]

[1]Yale University

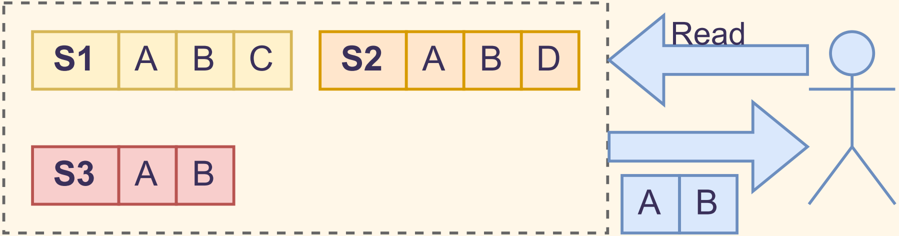
[2]Northeastern University

PLDI 2022

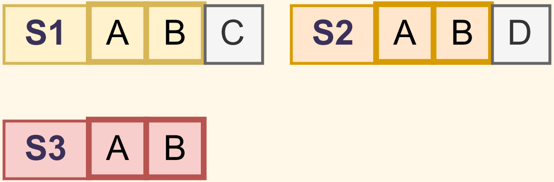
# Why Consensus?



# Why Consensus?



# Why Consensus?



# Why Consensus?

**IronFleet: Proving Practical Distributed Systems Correct**

Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch,
Bryan Parno, Michael L. Roberts, Srinath Setty, Brian Zill
Microsoft Research

**Verdi: A Framework for Implementing and Formally Verifying Distributed Systems**

James R. Wilcox Doug Woos Pavel Panchekha
Zachary Tatlock Xi Wang Michael D. Ernst Thomas Anderson
University of Washington, USA
{jrw12, dwoos, pavpan, ztatlock, xi, mernst, tom}@cs.washington.edu

**Abstract**

data loss and service outages [10, 42]. For example, in April 2011 a malfunction of failure recovery in Amazon Elastic Compute Cloud

**Programming and Proving with Distributed Protocols**

ILYA SERGEY, University College London, UK
JAMES R. WILCOX, University of Washington, USA
ZACHARY TATLOCK, University of Washington, USA

**I4: Incremental Inference of Inductive Invariants for Verification of Distributed Protocols**

Haojun Ma, Aman Goel, Jean-Baptiste Jeannin
Manos Kapritsos, Baris Kasikci, Karem A. Sakallah
University of Michigan
{mahaojun,amangoel,jeannin,manosk,barisk,karem}@umich.edu

**Abstract**

Designing and implementing distributed systems correctly is a very challenging task. Recently, formal verification has been successfully used to prove the correctness of distributed systems. At the heart of formal verification lies a computer-checked proof with an inductive invariant. Finding this inductive invariant, however, is the most difficult part of the proof. Alas, current proof techniques require inductive invariants to be found manually—and painstakingly—by the developer.

In this paper, we present a new approach, *Incremental Inference of Inductive Invariants* (I4), to automatically generate

which may manifest during production, resulting in loss of availability, revenue, and company reputation [14, 53, 54, 57]. This has led many researchers and companies to look for alternative ways to develop software with strong correctness guarantees.

Thankfully, the increasing need for availability has been paralleled by an increase in the capabilities of formal verification techniques. Over the last decade, a number of techniques and tools have been built to formally verify the correctness of complex systems software [9, 10, 30, 31, 38, 44, 45].

Unfortunately, existing approaches to formally verifying complex systems have a major scalability bottleneck.

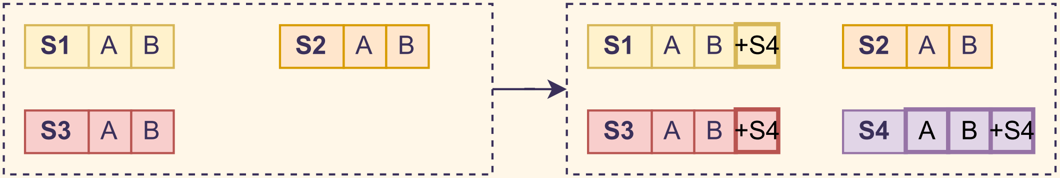
**Aneris: A Mechanised Logic for Modular Reasoning about Distributed Systems**

Morten Krogh-Jespersen, Amin Timany*, Marit Edna Ohlenbusch,
Simon Oddershede Gregersen, and Lars Birkedal
Aarhus University, Aarhus, Denmark

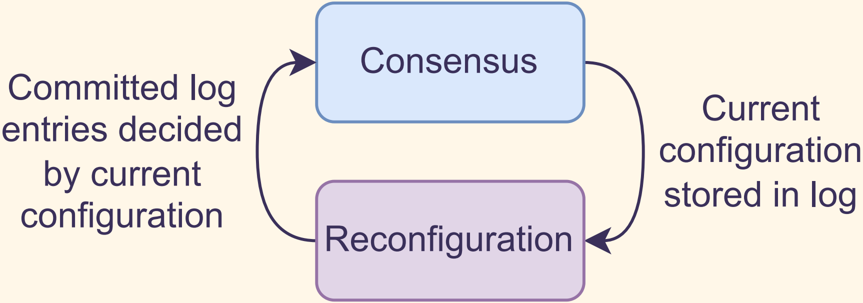
**Abstract.** Building network-connected programs and distributed systems is a powerful way to provide scalability and availability in a digital, always-connected era. However, with great power comes great complexity. Reasoning about distributed systems is well-known to be difficult.
In this paper we present Aneris, a novel framework based on separation logic supporting modular, node-local reasoning about concurrent and distributed systems. The logic is higher-order, concurrent, with higher-order store and network sockets, and is fully mechanized in the Coq proof assistant. We use our framework to verify an implementation of a load balancer that uses multi-threading to distribute load amongst multiple servers and an implementation of the *two-phase-commit* protocol with a replicated logging service as a client. The two examples certify that Aneris is well-suited for both horizontal and vertical modular reasoning.

# Why Reconfiguration?

Hot Reconfiguration



# Why Reconfiguration?



# Why Reconfiguration?

**bug in single-server membership changes** Subscribe

4784 views

**onga...@gmail.com** Jul 10, 2015, 12:58:53 AM

to raft...@googlegroups.com

Hi raft-dev,

Unfortunately, I need to announce a bug in the dissertation version of membership changes (the single-server changes, not joint consensus). The bug is potentially severe, but the fix I'm proposing is easy to implement.

Diego Ongaro
(Raft designer)

| Raft hot reconfiguration published | Critical bug discovered | Fix proposed | No complete safety proofs | Safety proved in Adore |
|---|---|---|---|---|
| August 2014 | May 2015 | July 2015 | | June 2021 |

# Contributions

- Adore: A novel abstraction for consensus with a generic hot reconfiguration scheme.

# Contributions

- Adore: A novel abstraction for consensus with a generic hot reconfiguration scheme.
- Coq proof of Adore's safety. First mechanized safety proof of reconfigurable consensus.

# Contributions

- Adore: A novel abstraction for consensus with a generic hot reconfiguration scheme.
- Coq proof of Adore's safety. First mechanized safety proof of reconfigurable consensus.
- Several practical instantiations of Adore's generic reconfiguration.

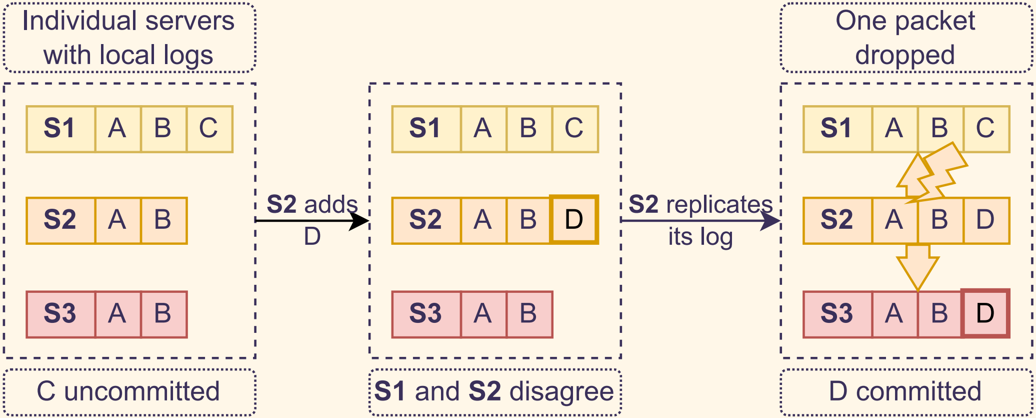
# Contributions

- Adore: A novel abstraction for consensus with a generic hot reconfiguration scheme.
- Coq proof of Adore's safety. First mechanized safety proof of reconfigurable consensus.
- Several practical instantiations of Adore's generic reconfiguration.
- Coq proof that Raft refines Adore.

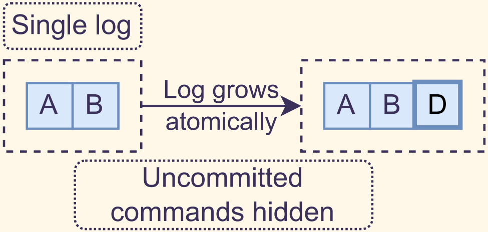
# Contributions

- Adore: A novel abstraction for consensus with a generic hot reconfiguration scheme.
- Coq proof of Adore's safety. First mechanized safety proof of reconfigurable consensus.
- Several practical instantiations of Adore's generic reconfiguration.
- Coq proof that Raft refines Adore.
- Automated extraction from the Coq Raft specification to executable OCaml.

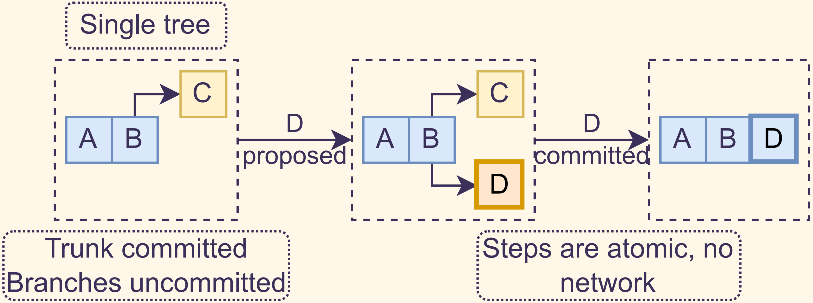
# Network-Based Abstractions



# State Machine Replication (SMR)



# Atomic Distributed Object (ADO)



# The Best of Both Worlds

Network-Based Models

- ✓ Exposes enough detail for protocol-level reasoning.
- × Mixes implementation and protocol-level logic.

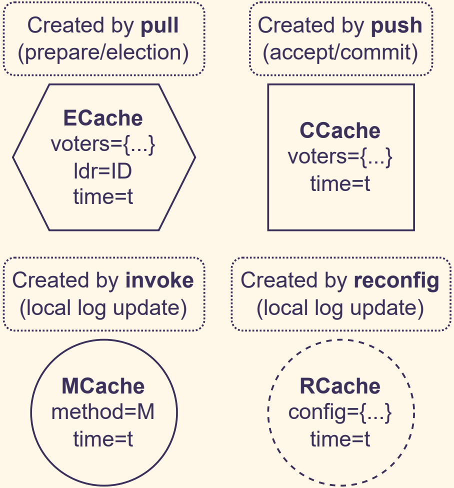
SMR/ADO

- ✓ Atomic object model is more convenient.
- × Loses too many important details.

**Adore**

- ✓ Atomic object model hides network-level communication.
- ✓ Retains enough information about local state for safety reasoning.

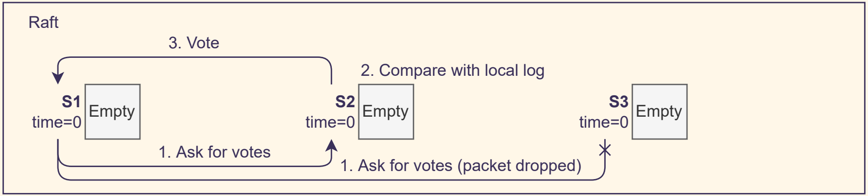
# Adore State


Created by **pull**
(prepare/election)
**ECache**
voters={...}
ldr=ID
time=t
Created by **push**
(accept/commit)
**CCache**
voters={...}
time=t
Created by **invoke**
(local log update)
**MCache**
method=M
time=t
Created by **reconfig**
(local log update)
**RCache**
config={...}
time=t

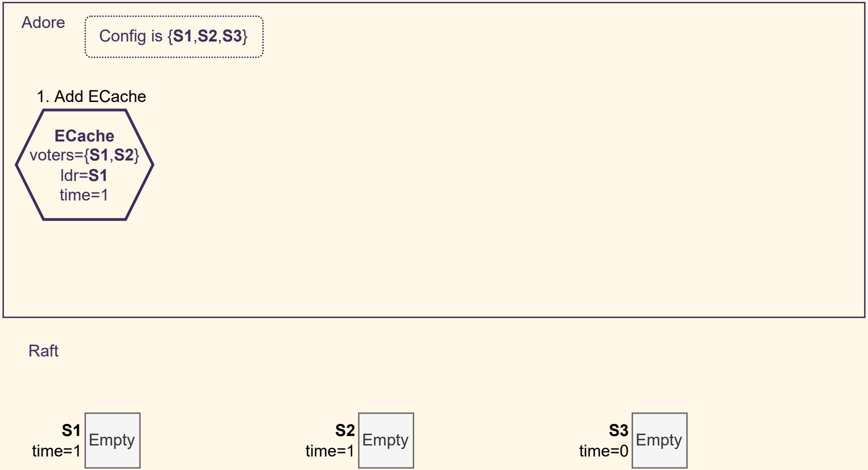
# Adore API — Pull

Adore

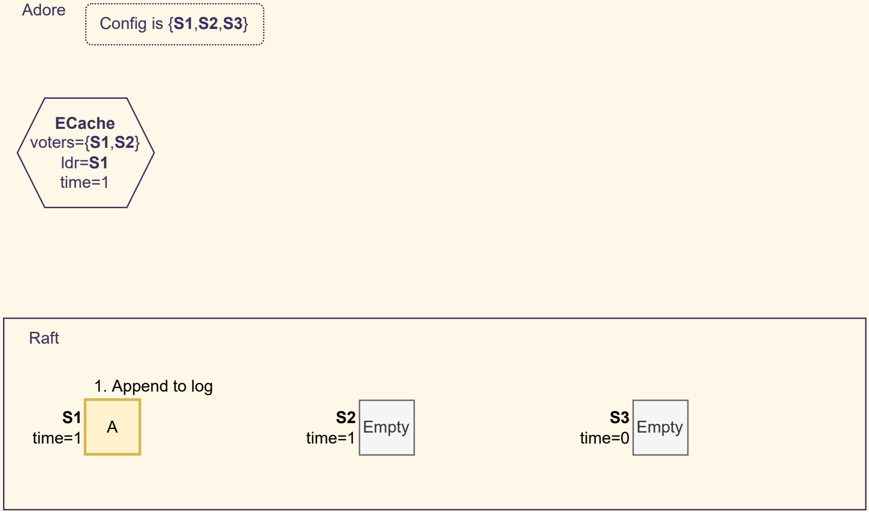
Config is {**S1**,**S2**,**S3**}



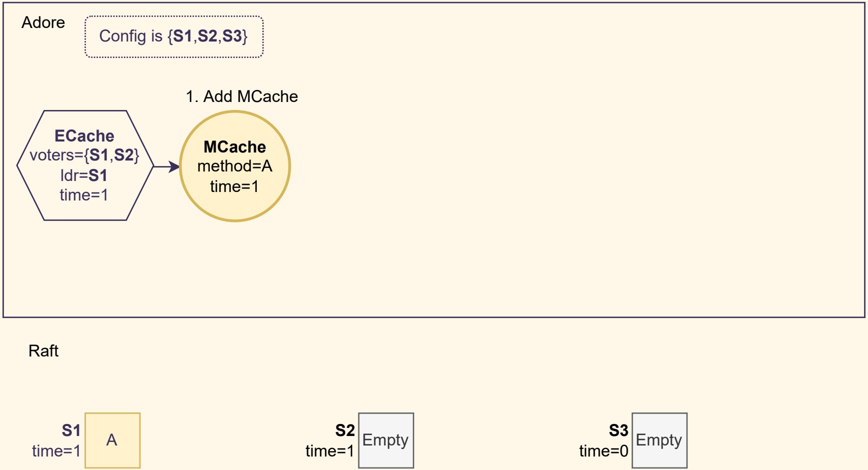
# Adore API — Pull



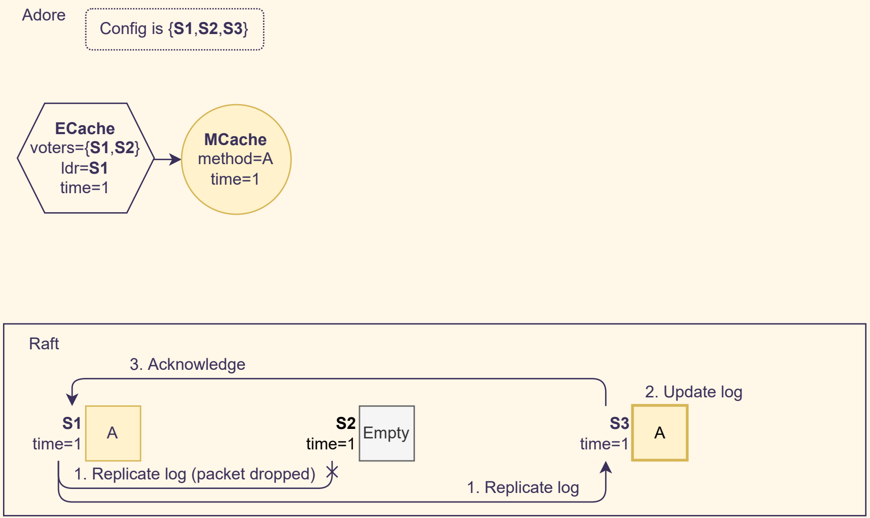
# Adore API — Invoke



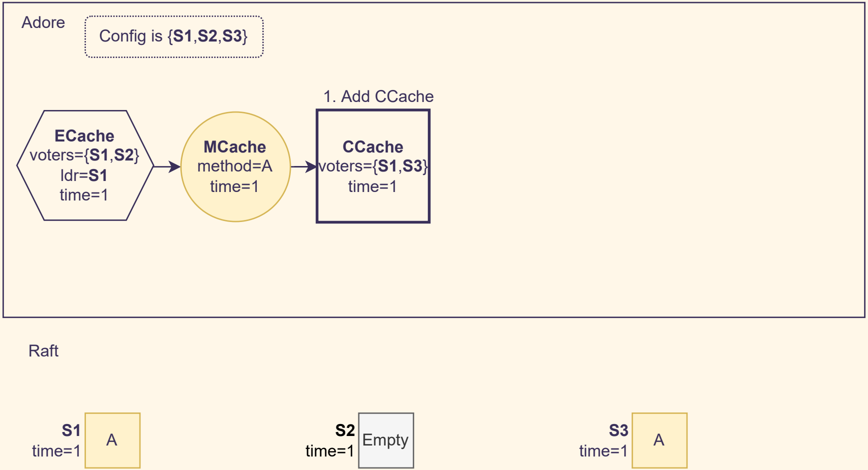
# Adore API — Invoke



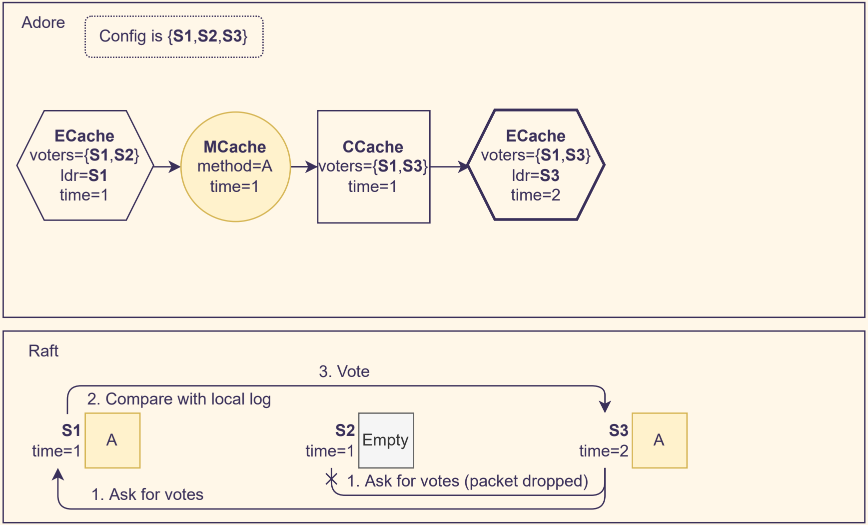
# Adore API — Push



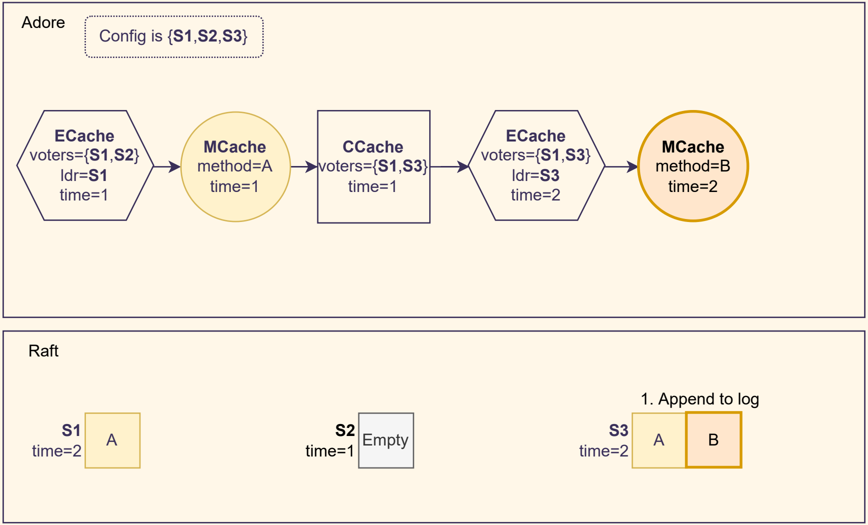
# Adore API — Push



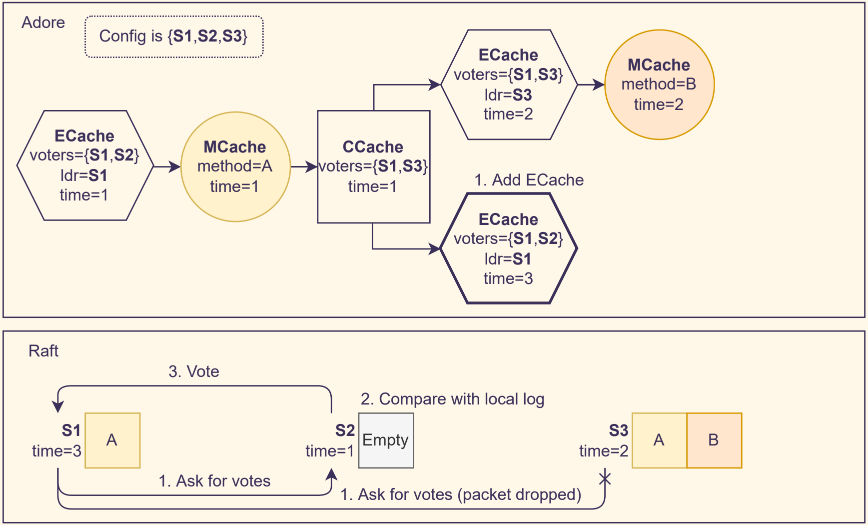
# Adore API — Steady State



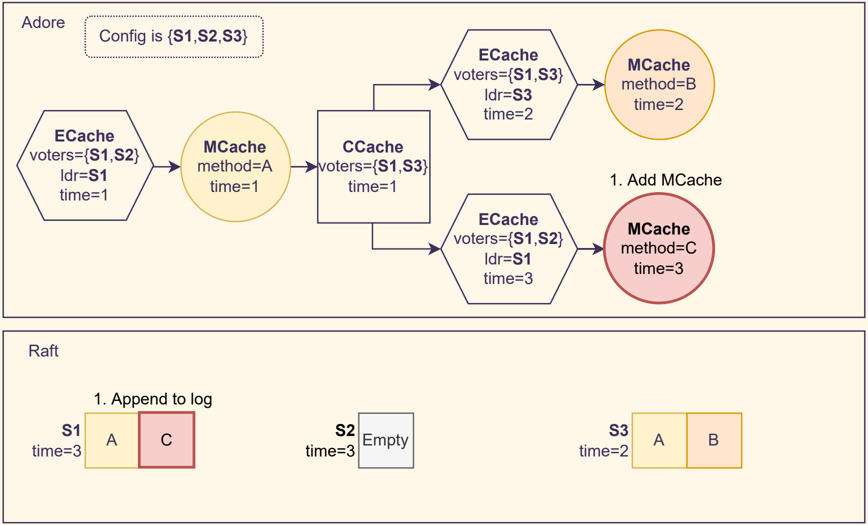
# Adore API — Steady State



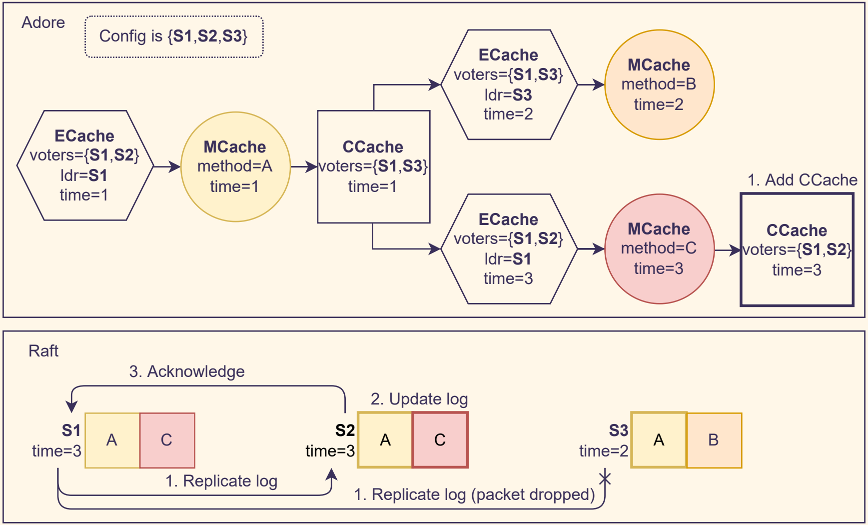
# Adore API — Branching



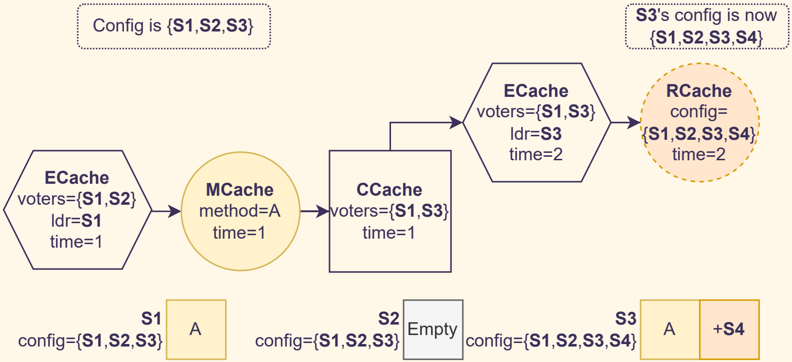
# Adore API — Branching



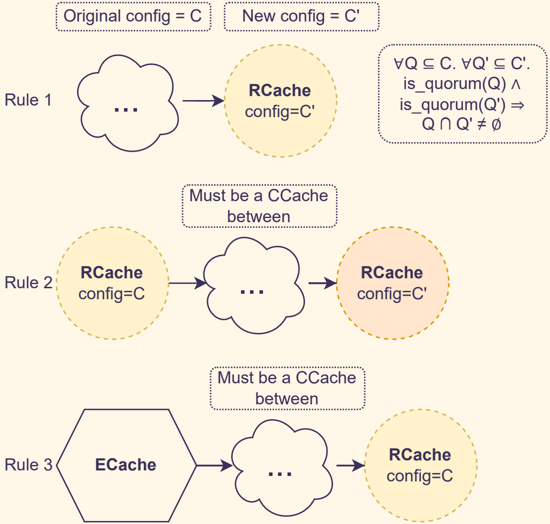
# Adore API — Branching



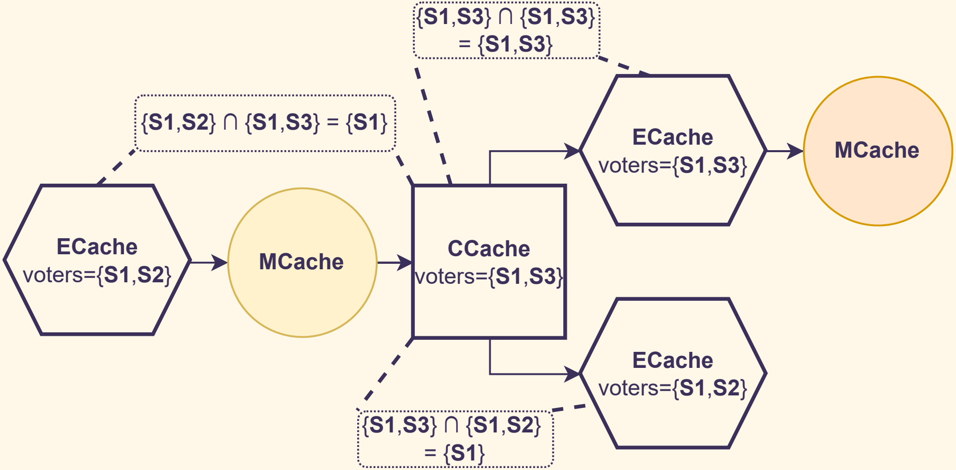
# Reconfiguration in Adore



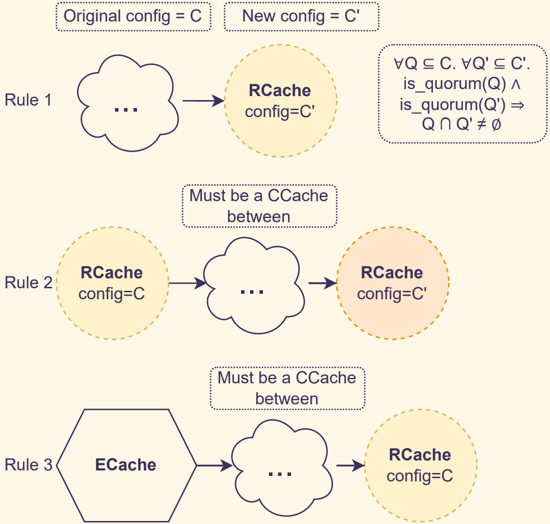
# Reconfiguration Rules



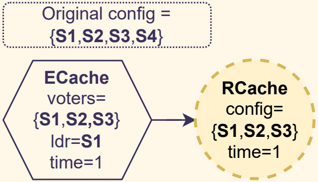
# Reconfiguration Rules



# Reconfiguration Rules

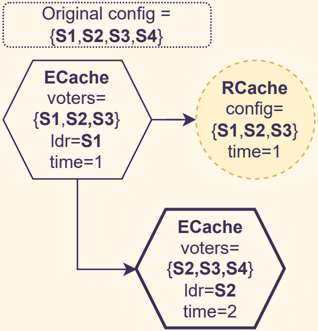


# Reconfiguration Rules



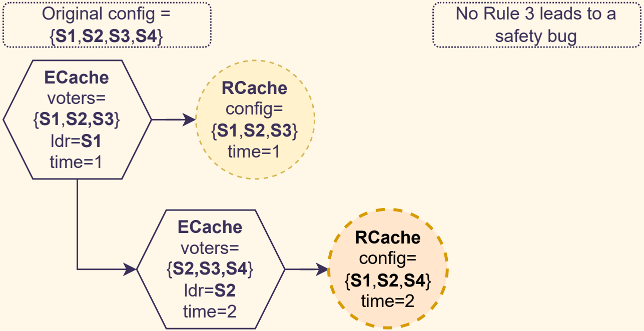

No Rule 3 leads to a safety bug

# Reconfiguration Rules

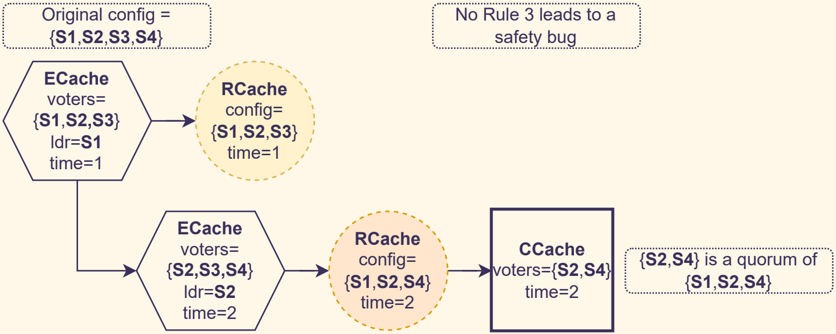



No Rule 3 leads to a safety bug

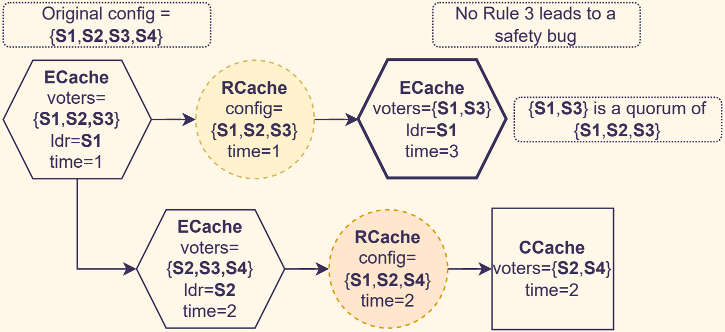
# Reconfiguration Rules



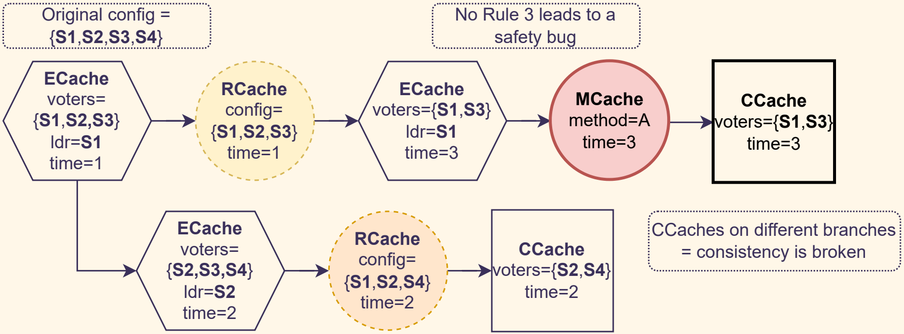
# Reconfiguration Rules



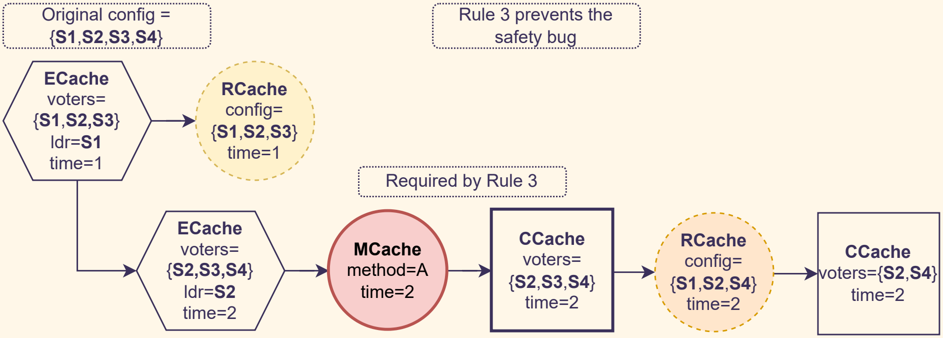
# Reconfiguration Rules



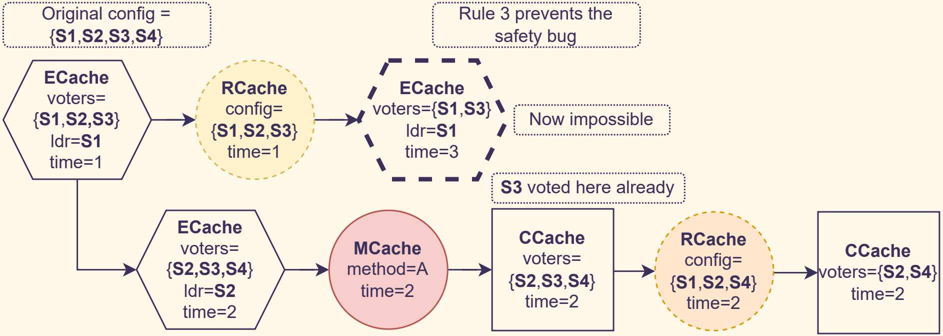
# Reconfiguration Rules



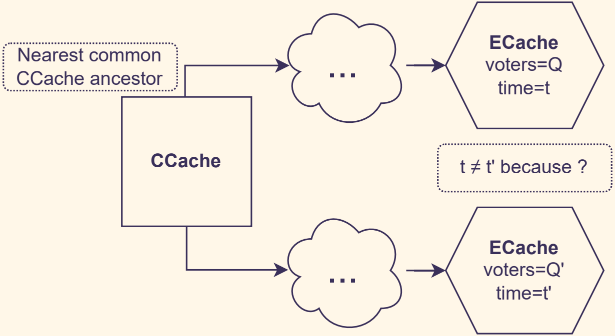
# Reconfiguration Rules



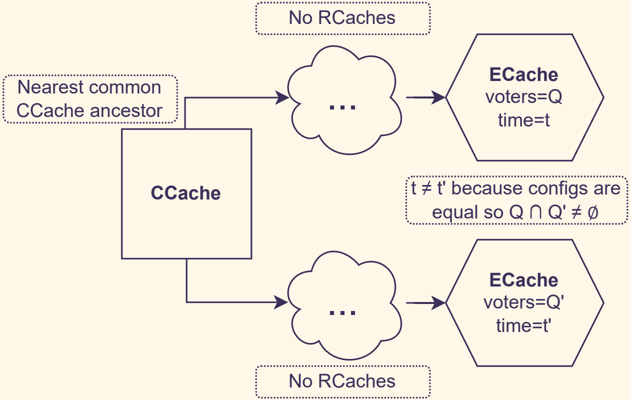
# Reconfiguration Rules



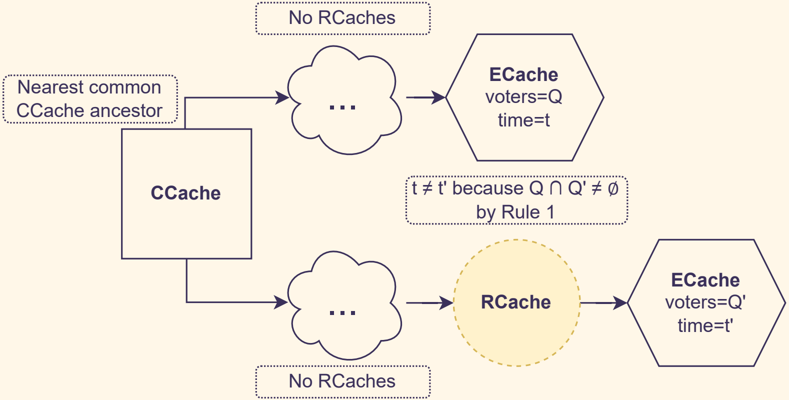
# Proving Safety



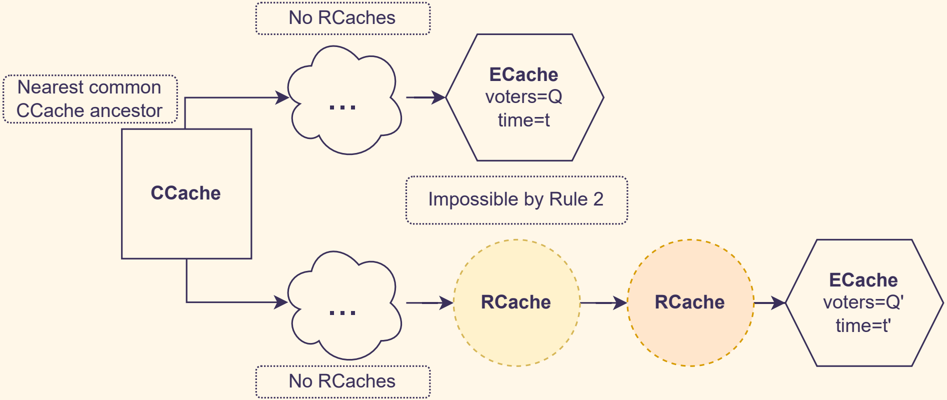
# Proving Safety



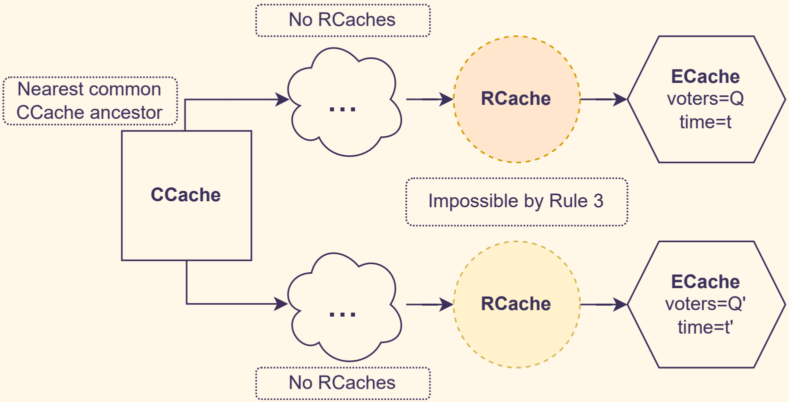
# Proving Safety



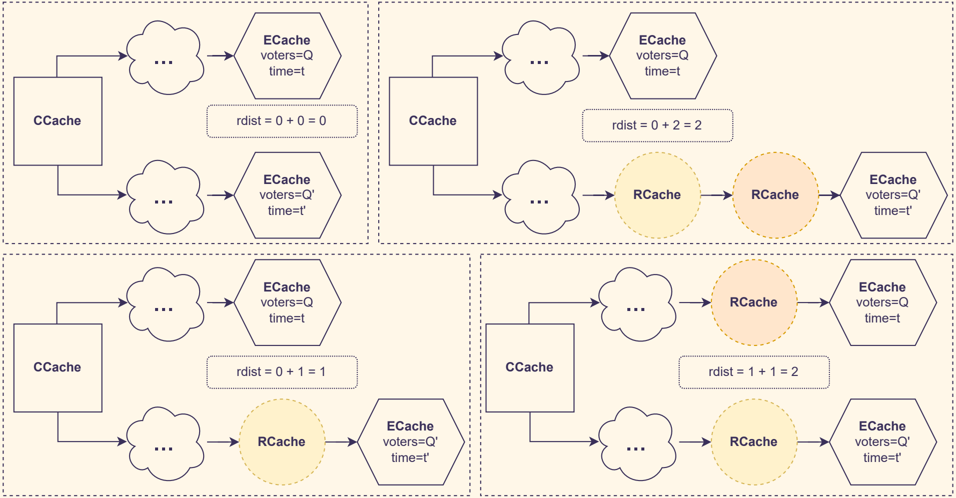
# Proving Safety



# Proving Safety

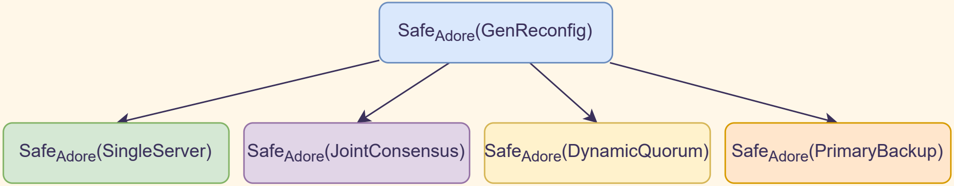


# Proving Safety



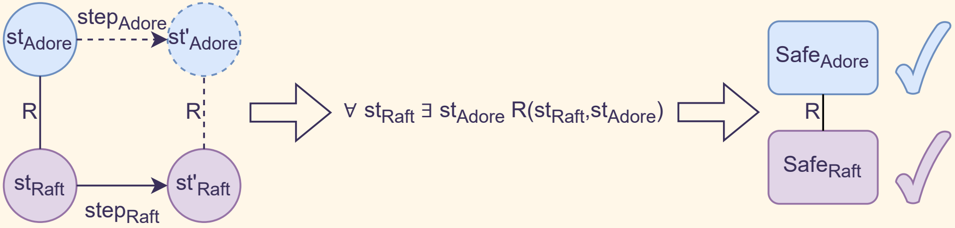
# Generalized Quorums

- Safety proved once for generic reconfiguration scheme.
- Can be instantiated many times with minimal proof effort.
- Details in Section 6 of the paper.



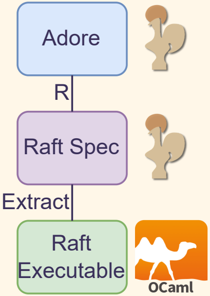
# Refinement

- Refinement between Raft network-based specification and Adore.
- Also generic with respect to reconfiguration scheme.
- Details in Section 5 of the paper.



# Extraction

- Automated extraction from Coq specification to executable OCaml.
- Safety guaranteed through Adore and refinement.
- Evaluation in Section 7 of the paper.



# Conclusion

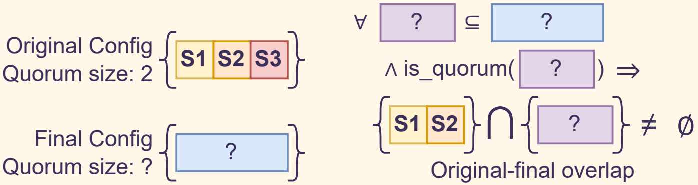
- Adore: A novel protocol-level abstraction for consensus.
- First safety proof for consensus with generic hot reconfiguration schemes.
- Refinement with network-level specification and extraction to executable.

# Proof Effort

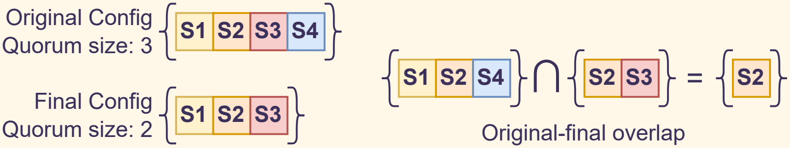
| | Language | Proof LOC | Proof Time |
|---|---|---|---|
| **Adore** | | | |
| Reusable Library | | ∼6k | 2 person-weeks |
| Safety Proof | Coq | ∼4k | 3 person-weeks |
| Total | | ∼10k | 5 person-weeks |
| MongoRaftReconfig[1] | TLA$^+$ | ∼3k | 5–6 person-months |

[1] William Schultz, Ian Dardik, and Stavros Tripakis. 2022. Formal Verification of a Distributed Dynamic Reconfiguration Protocol. CPP '22

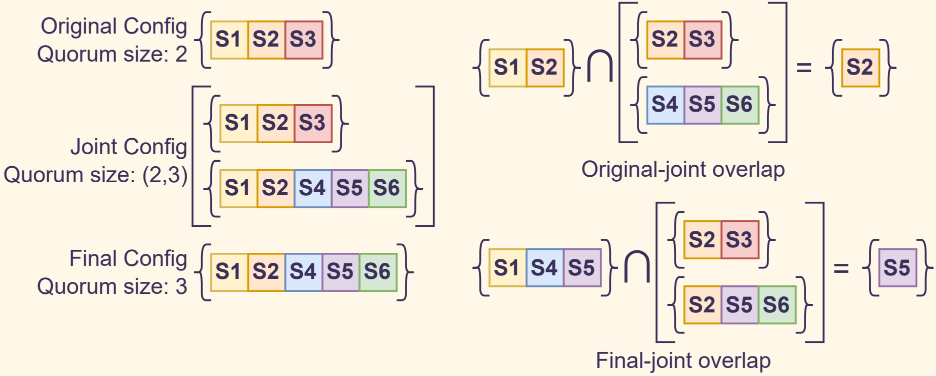
# Generalized Quorums



# Generalized Quorums



# Generalized Quorums



# Generalized Quorums