Convergence is Half Way to Consensus

Gowtham Kaki
University of Colorado Boulder

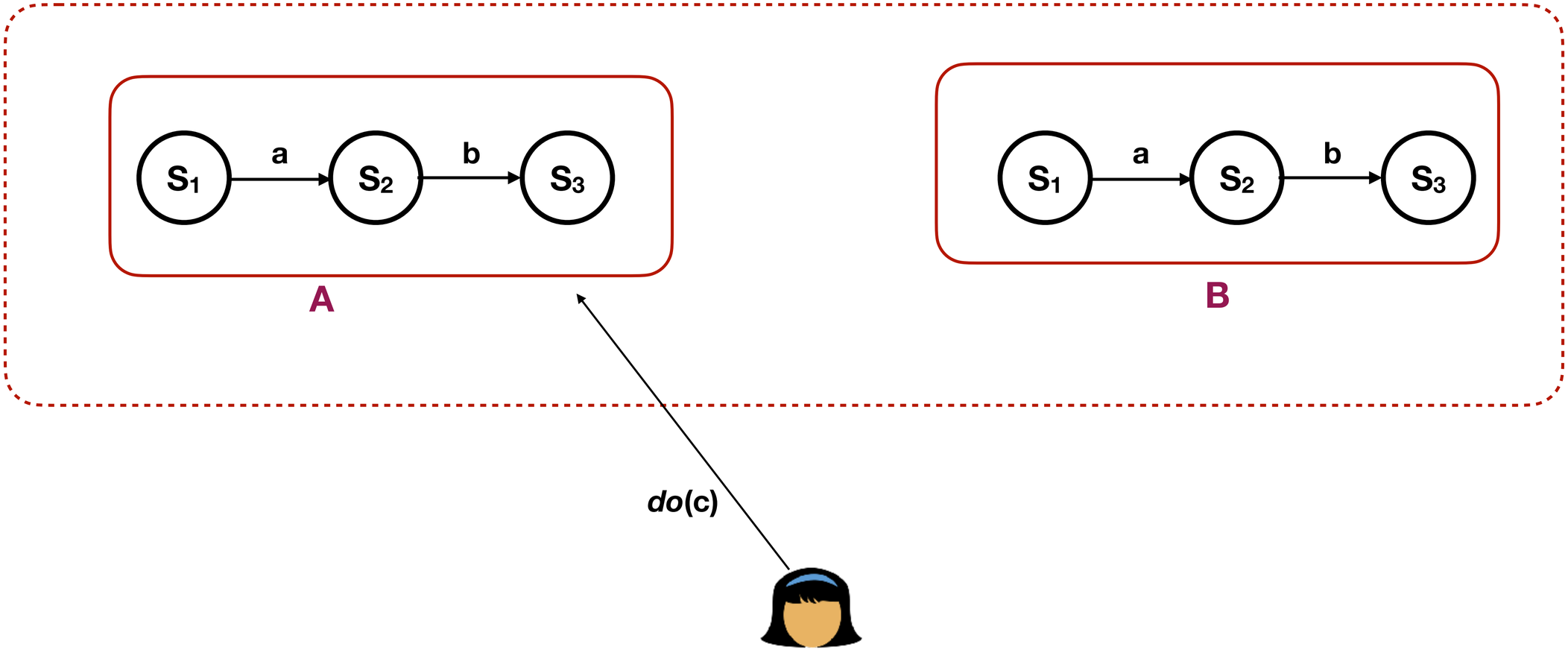
CU Programming Languages
& Verification

Distributed Convergence is Half Way to Distributed Consensus

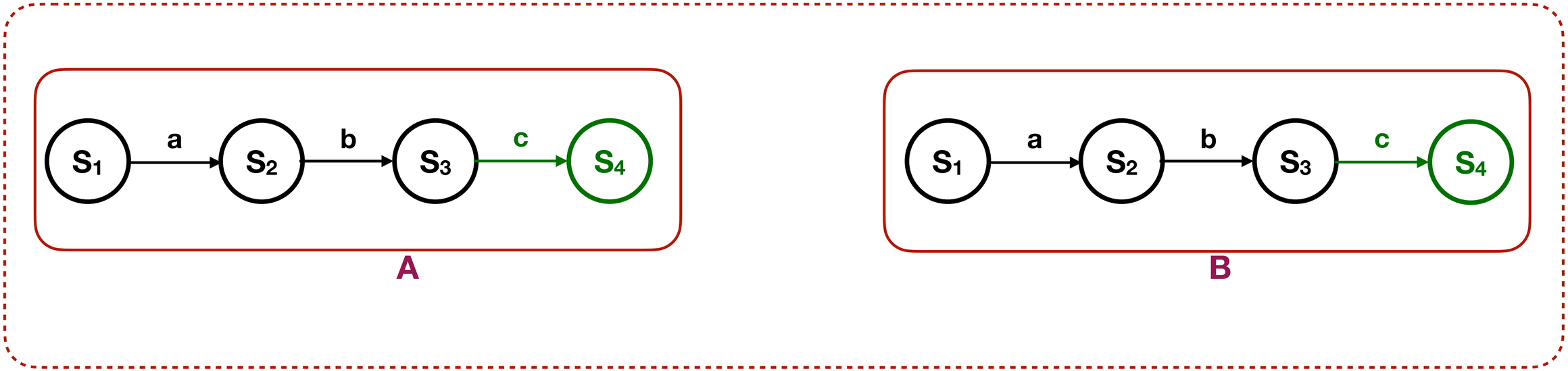
Gowtham Kaki
University of Colorado Boulder

CU Programming Languages
& Verification

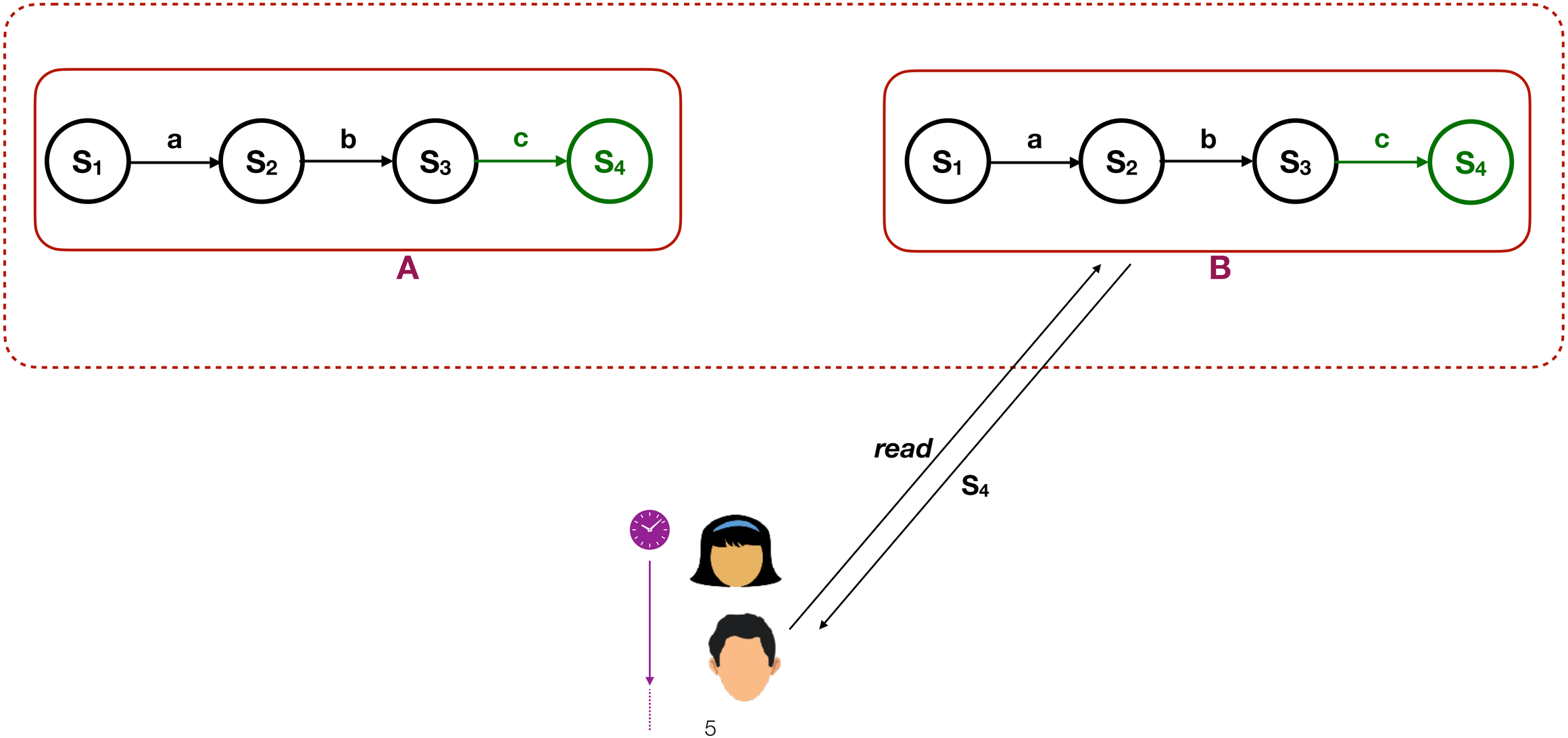
Replicated State Machines



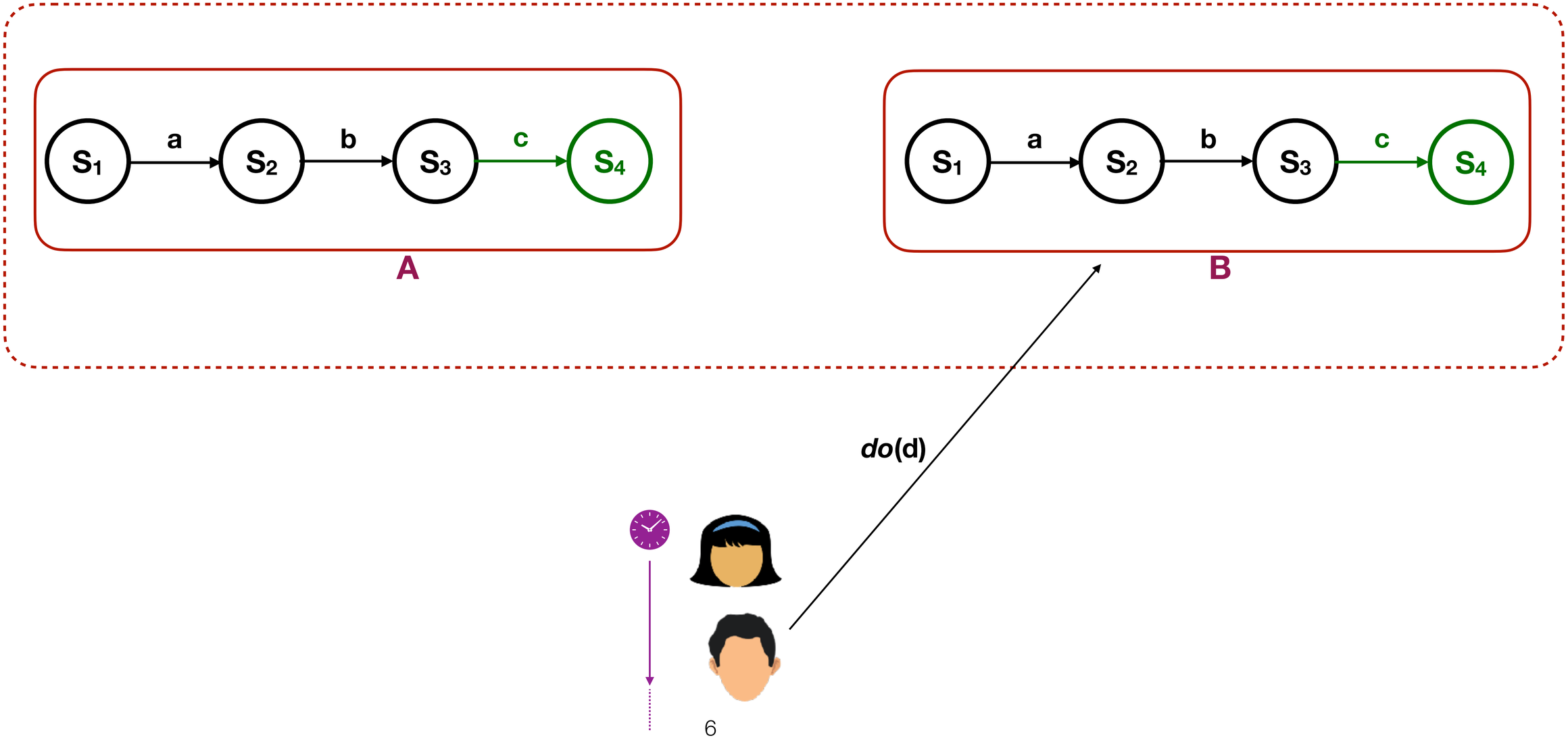
Replicated State Machines



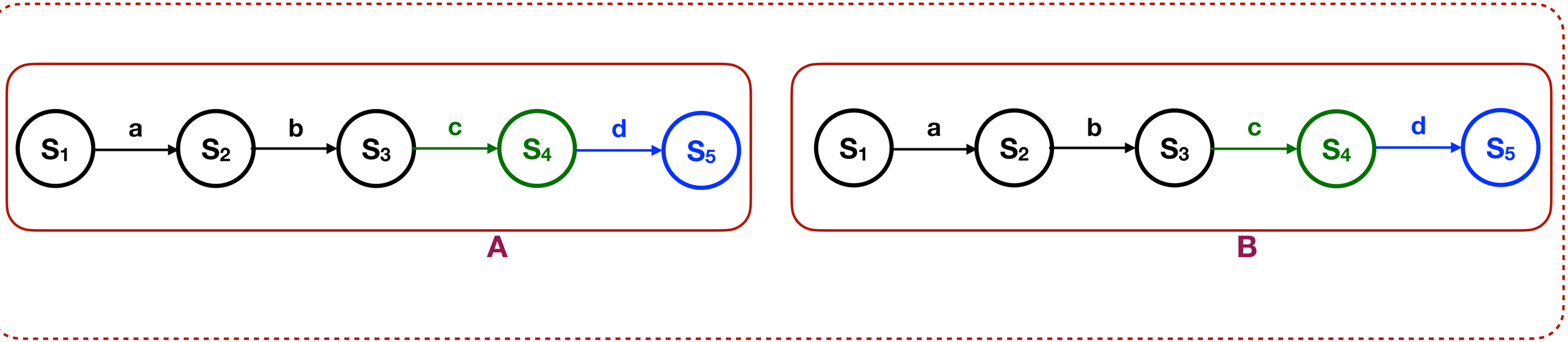
Replicated State Machines

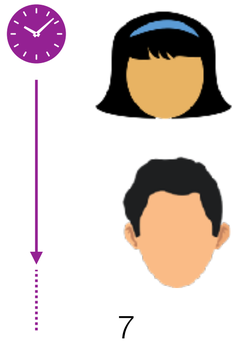


Replicated State Machines



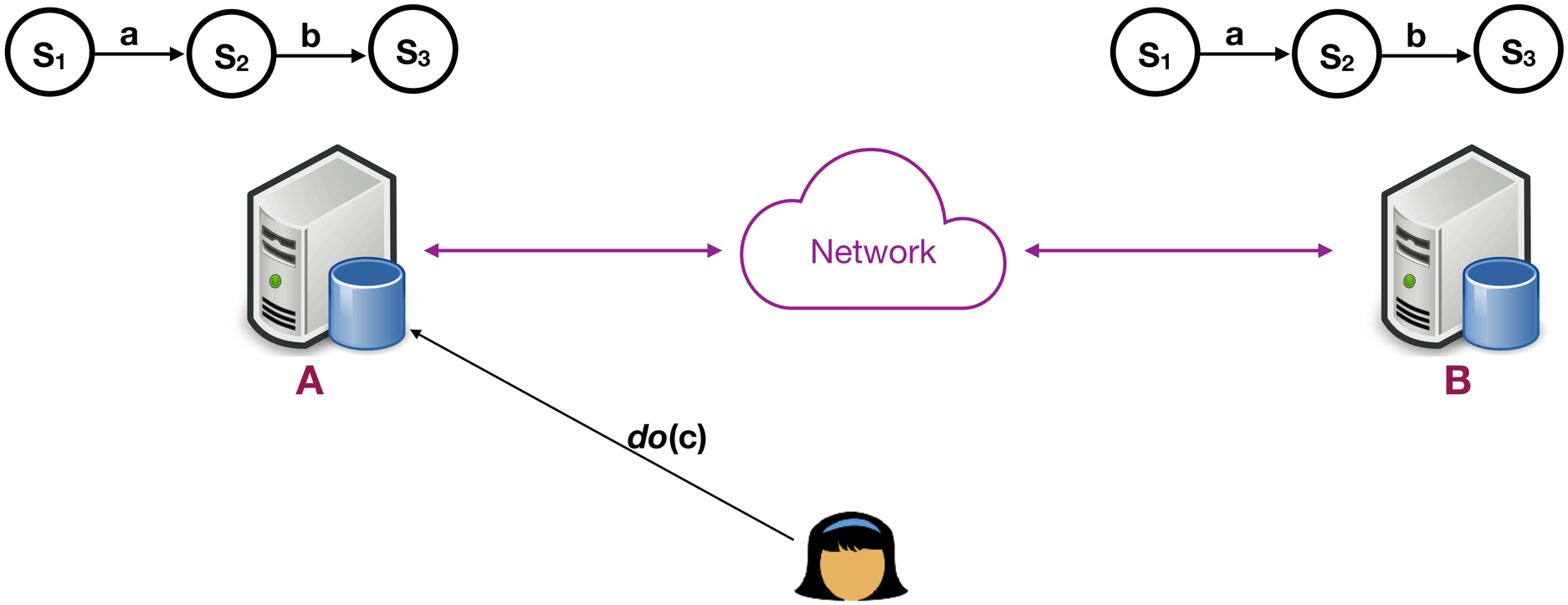
Replicated State Machines



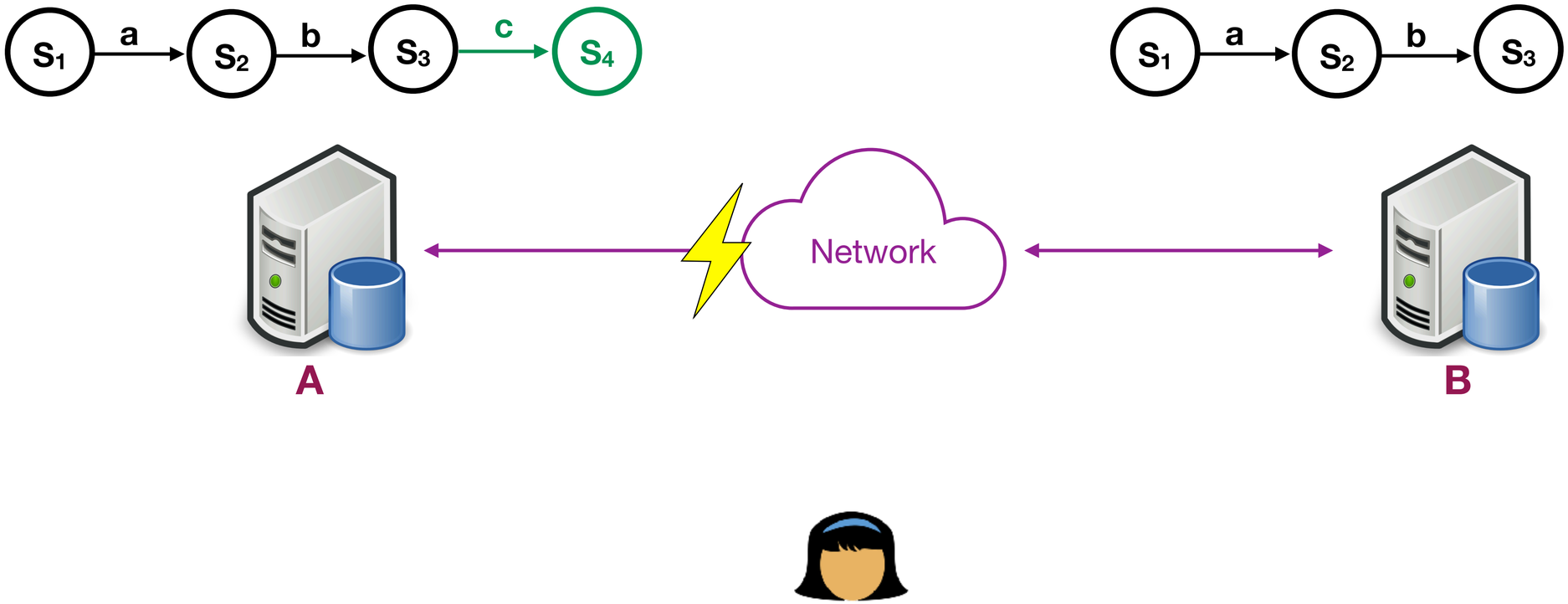


“Strong Consistency”

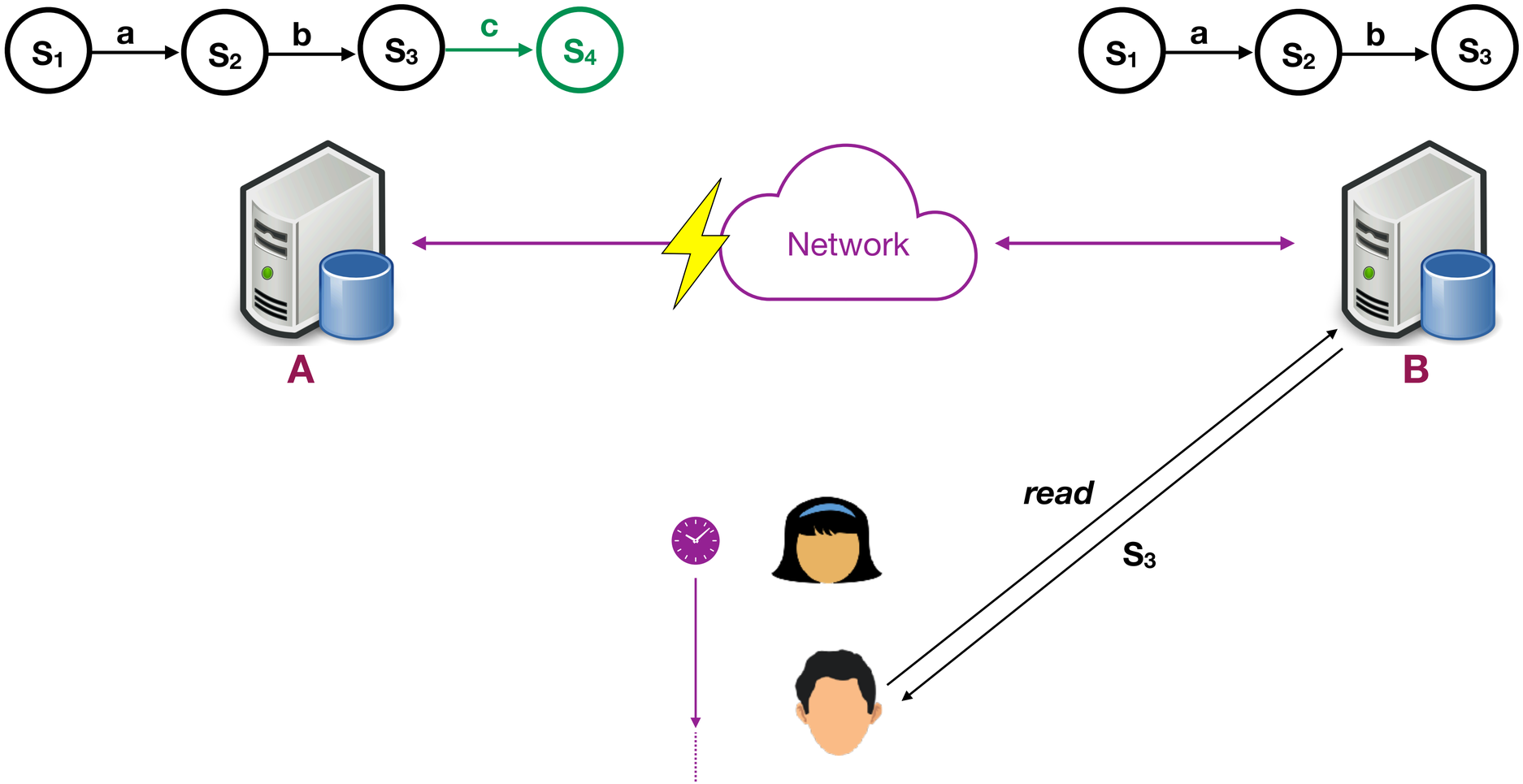
Replicated State Machines: In Practice



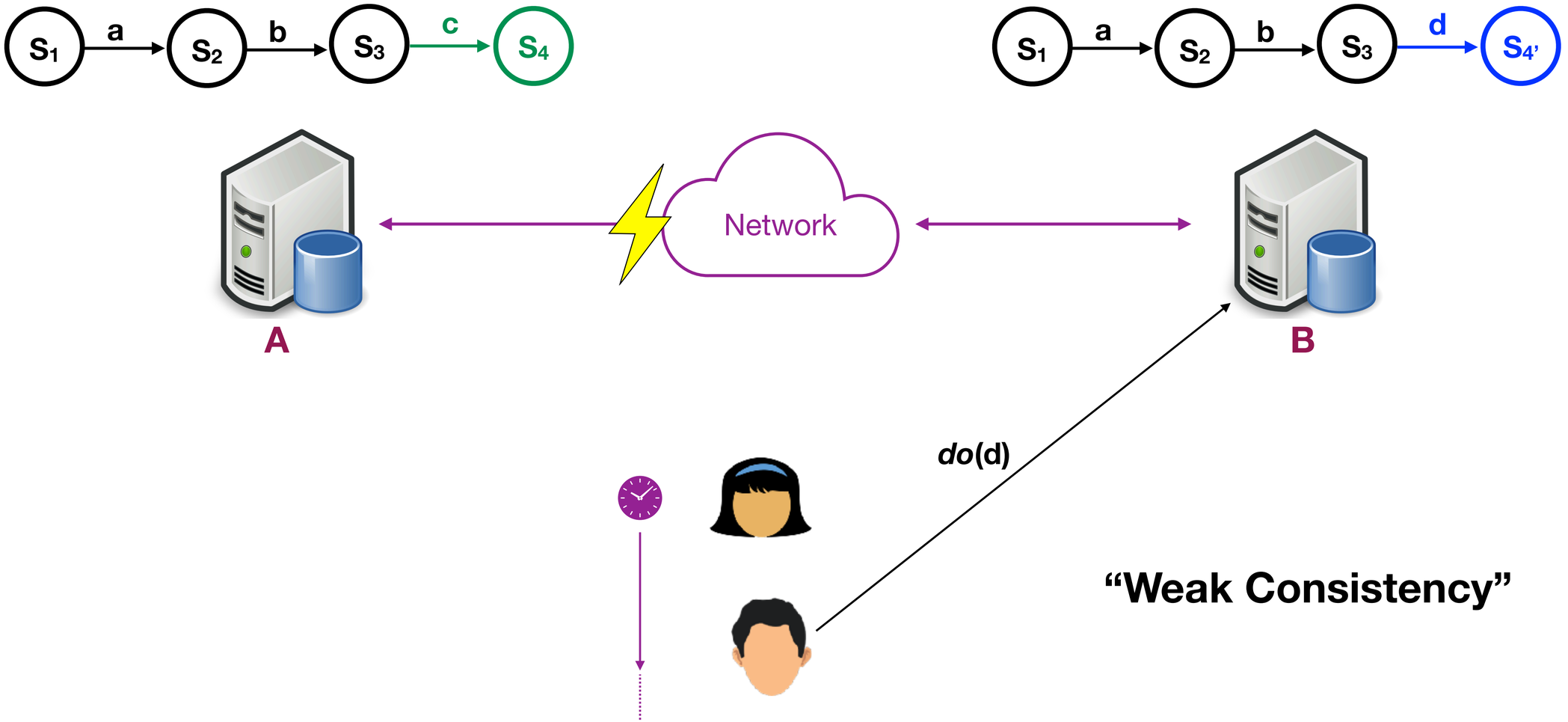
Replicated State Machines: In Practice



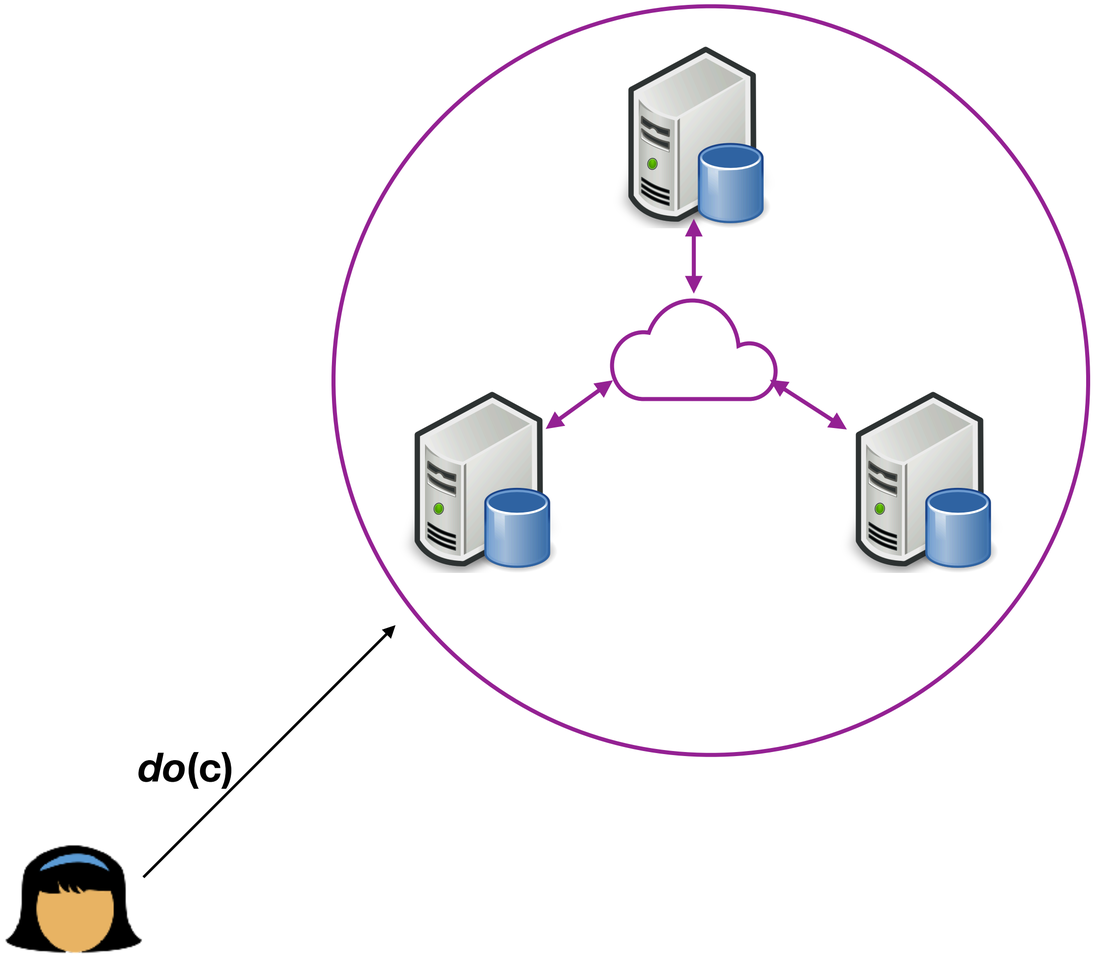
Replicated State Machines: In Practice

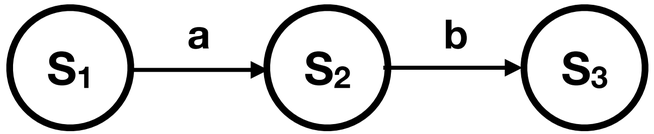


Replicated State Machines: In Practice

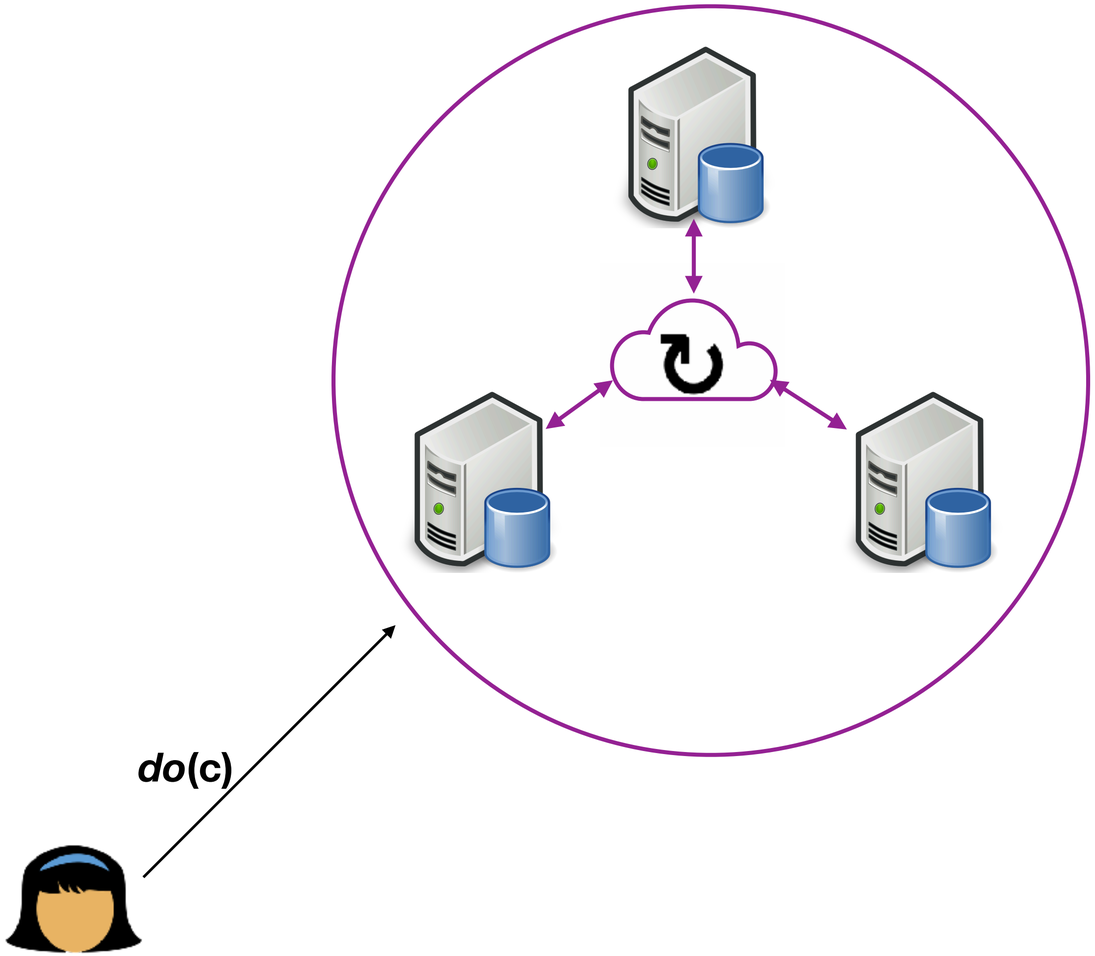


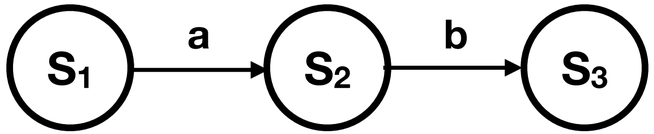
Consensus in Replicated State Machines



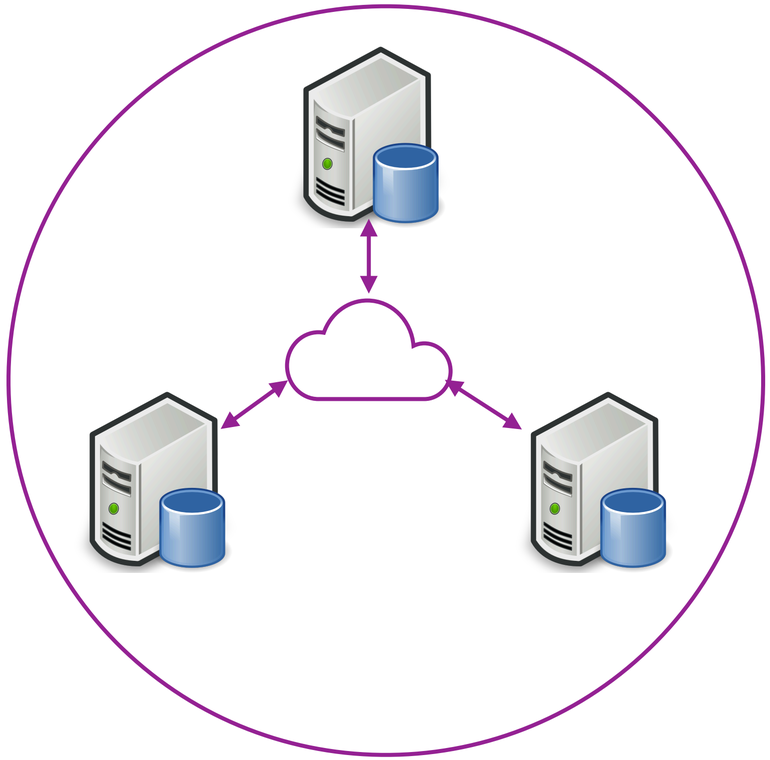


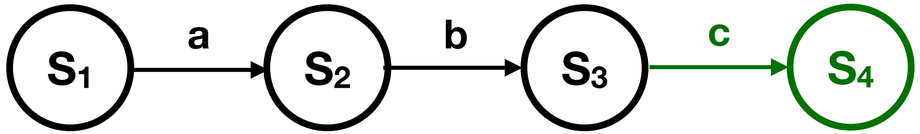
Consensus in Replicated State Machines



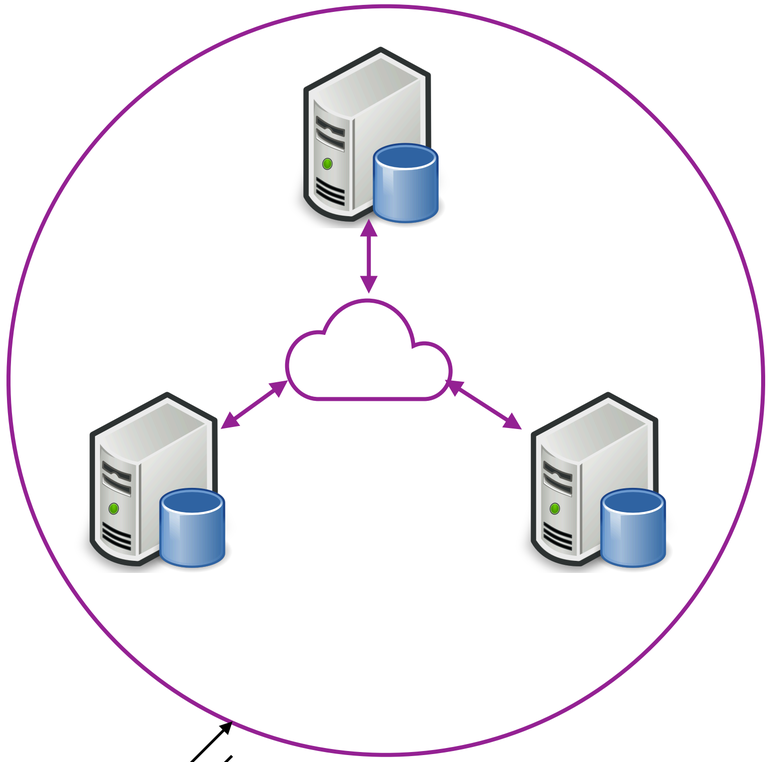


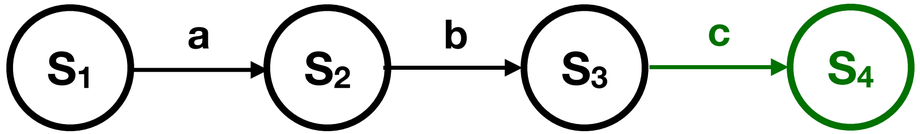
Consensus in Replicated State Machines

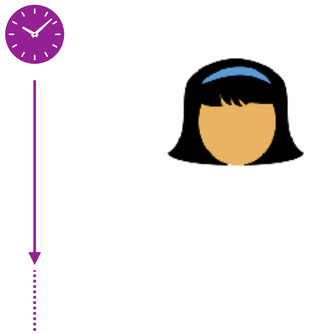


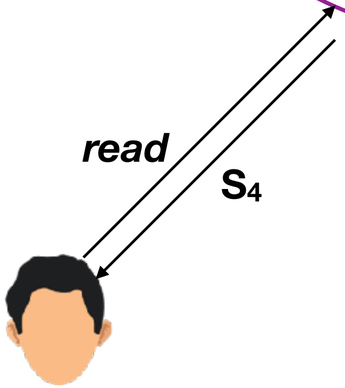


Consensus in Replicated State Machines

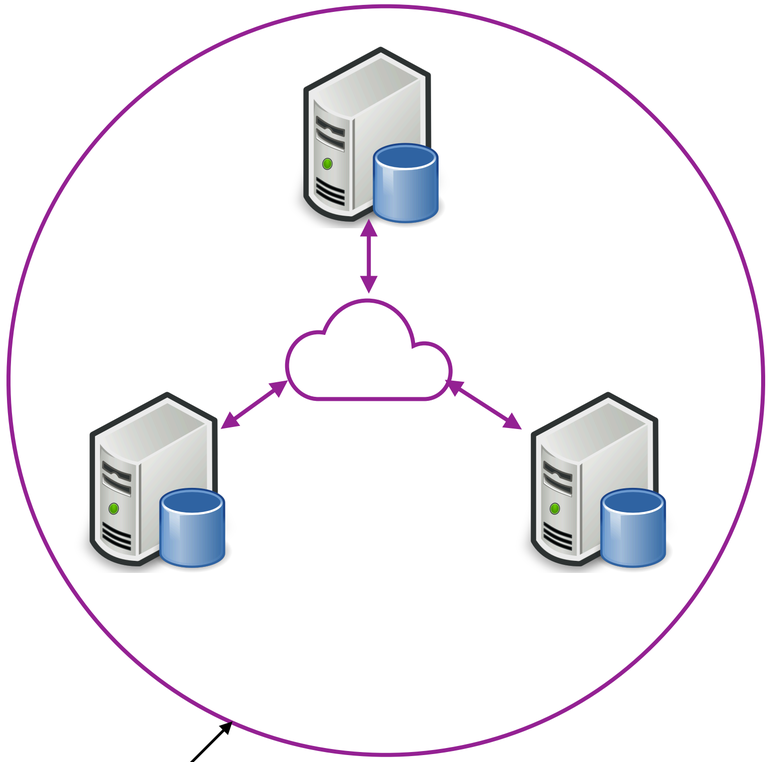


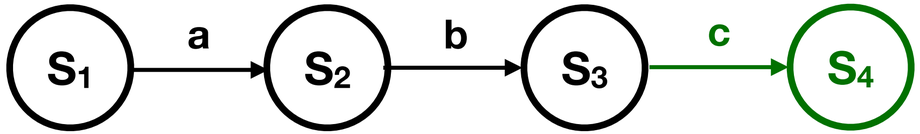


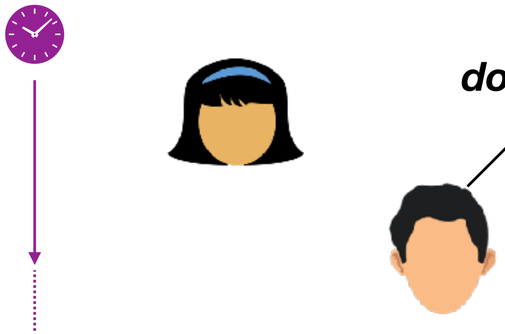




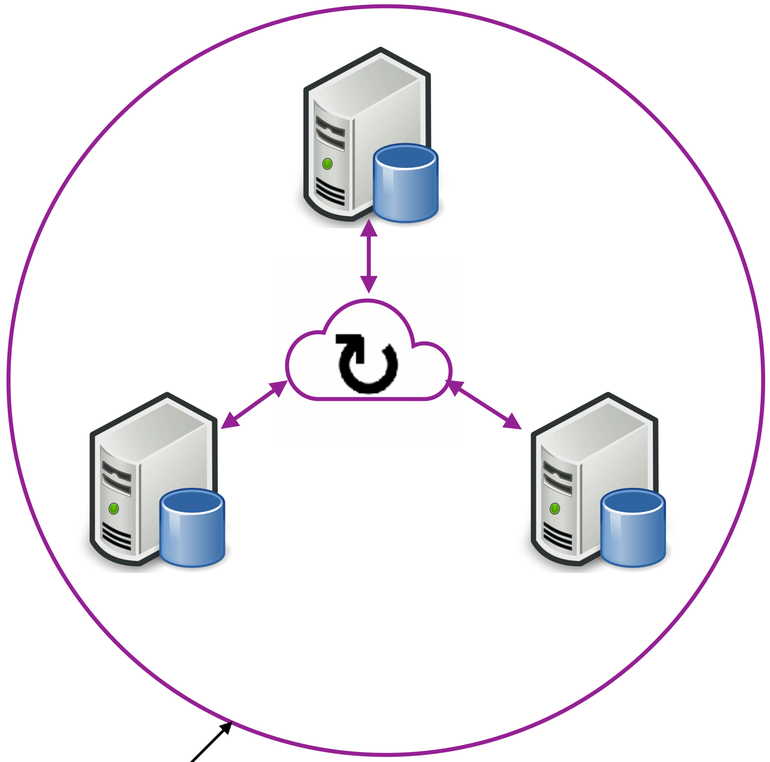
Consensus in Replicated State Machines

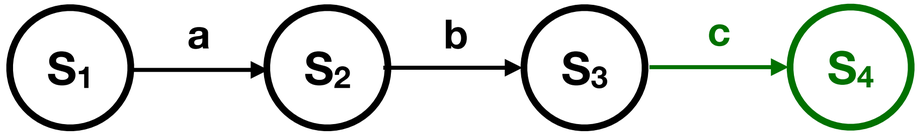


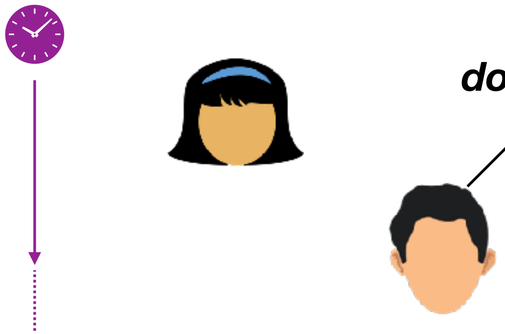




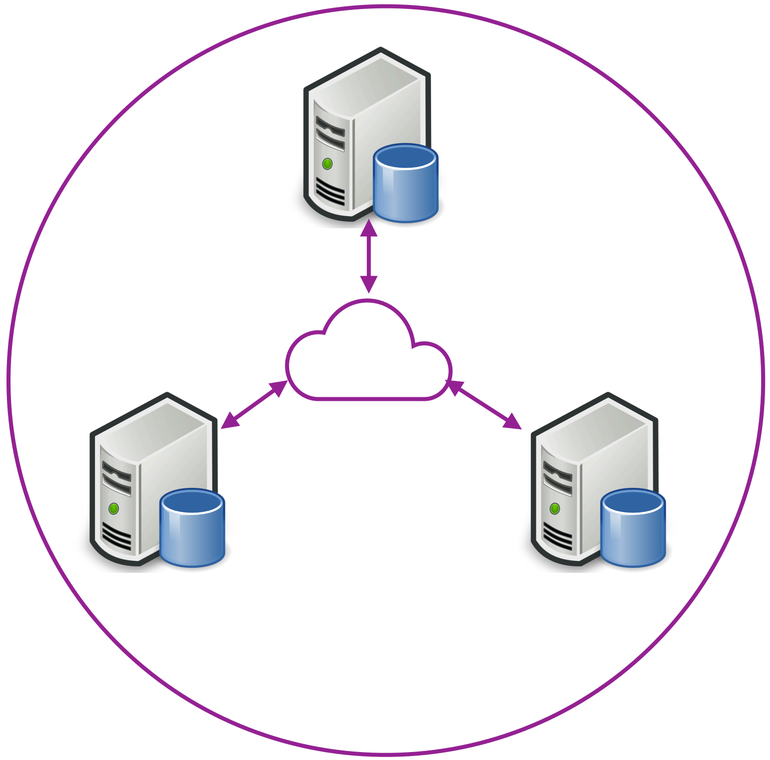
Consensus in Replicated State Machines

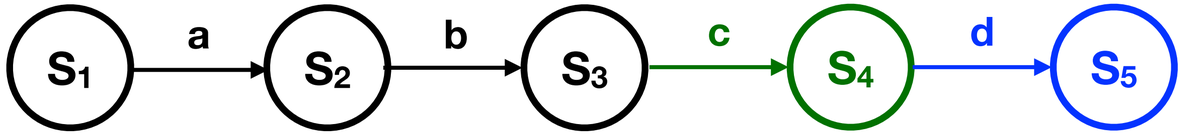


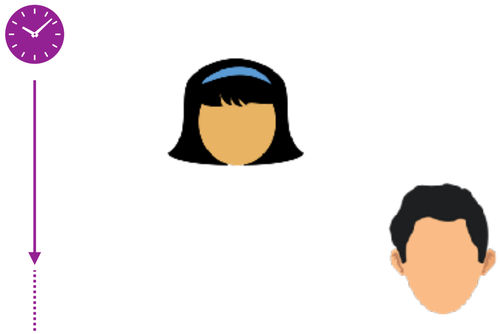




Consensus in Replicated State Machines







Consensus Algorithms

- Paxos [Lamport 1998]
  - Multi-Paxos [Lamport 2001]
  - Disk Paxos [Gafni et al 2002]
  - Fast Paxos [Lamport 2006]
  - Stoppable Paxos [Malkhi et al 2008]
  - …
- Raft [Ongaro et al 2014]
  - Muti-Region Raft [Cockroach DB, TikV DB]
  - FlexiRaft [Yadav et al 2023]
- Viewstamped Replication [Oki et al 1988]
- Zookeeper [Junqueira et al 2011]
- Chubby [2006]
- …

- Impossible to guarantee availability (CAP)
- Hard to guarantee correctness

Via formal verification

Problem with Formal Verification: Too Much Work!

- Hawblitzel et al. 2015 *IronFleet: Proving Practical Distributed Systems Correct.*
  "…developing IronFleet and applying it to two real systems required approximately 3.7 person-years"
- Wilcox et al. 2015 *Verdi: A Framework for Implementing and Formally Verifying Distributed Systems.*
- Woos et al. 2016 *Planning for Change in a Formal Verification of the Raft Consensus Protocol.*
  "…required iteratively discovering and proving 90 system invariants"
  "…530 lines of code and 50,000 lines of proof."
- Padon et al. 2017 *Paxos made EPR: Decidable Reasoning about Distributed Protocols.*
- Taube et al. 2018 *Modularity for Decidability of Deductive Verification with Applications to Distributed Systems.*
  "…took approximately 3 person-months …300 [lines] of invariants and ghost code."
- Gleissenthall et al. 2019. *Pretend Synchrony: Synchronous Verification of Asynchronous Distributed Systems.*

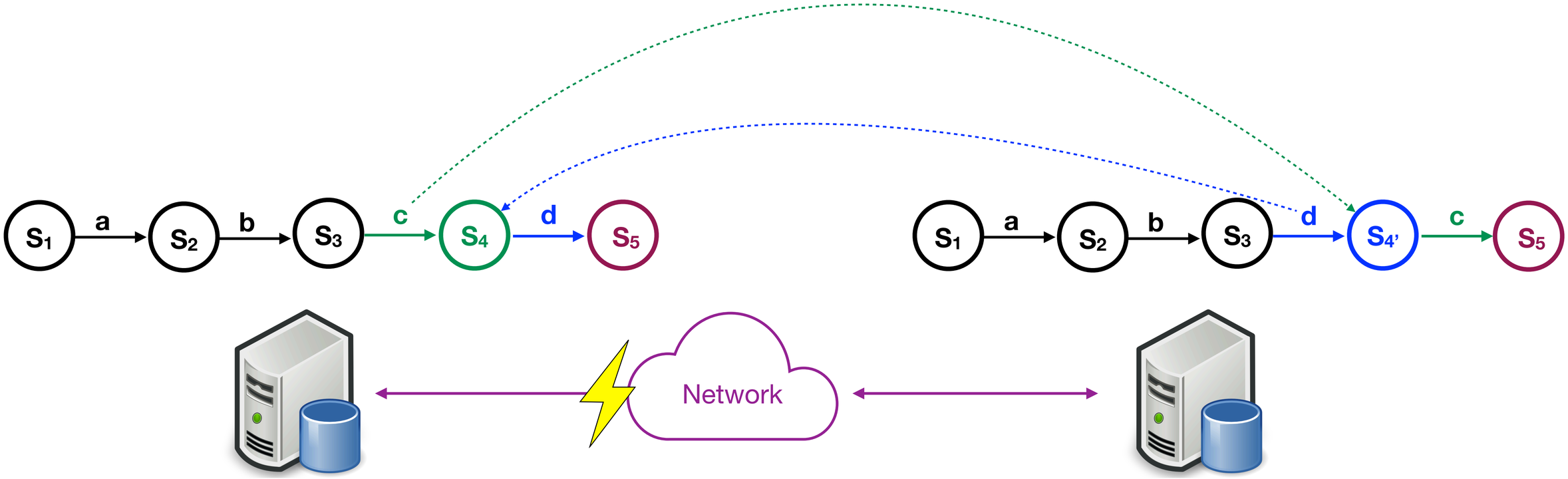
Key Questions

Can we avoid strong consistency/consensus?

- Impossible to guarantee availability (CAP)
- Hard to guarantee correctness

Can we make formal verification easier?

Eventual Consistency, i.e., Convergence



Operations `f` and `g` on data type `T` are commutative iff:

$$\forall s : T,\ \ f(g(s)) = g(f(s))$$

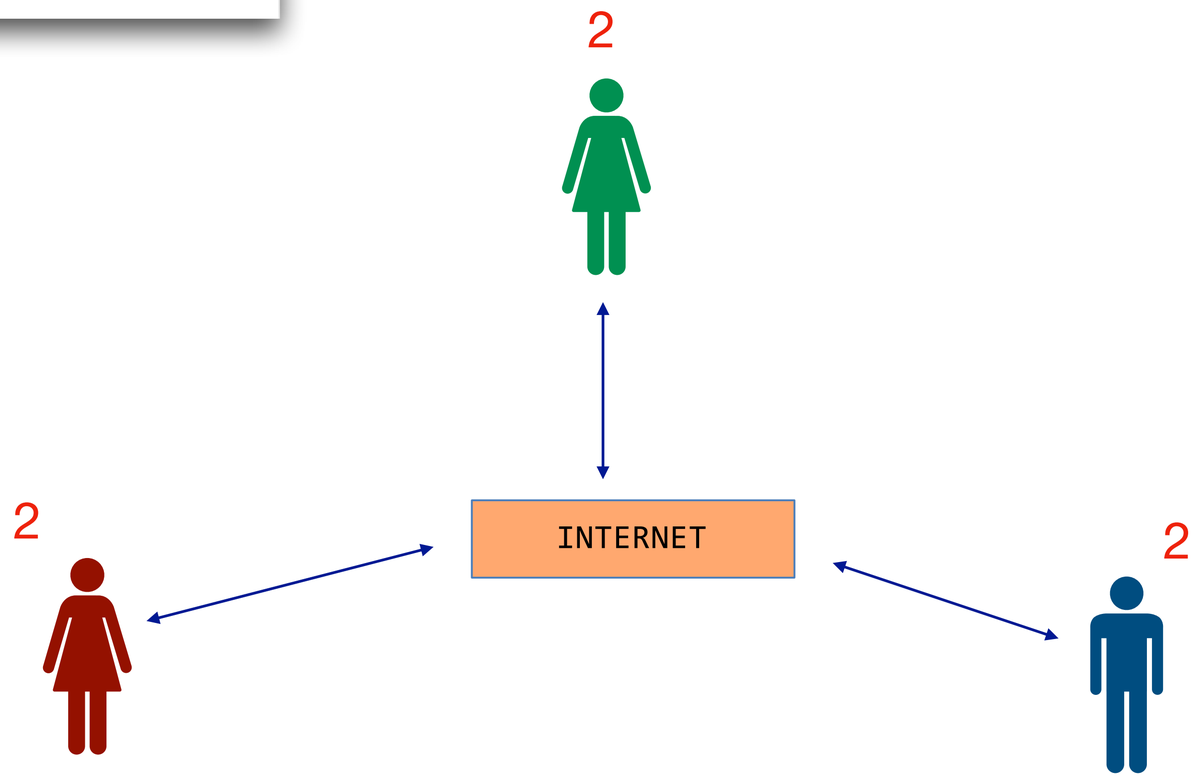
If all operations on `T` are commutative then `T` is a conflict-free replicated data type (CRDT)

CRDTs

Conflict-free Replicated Data Types *

Marc Shapiro, INRIA & LIP6, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Carlos Baquero, Universidade do Minho, Portugal
Marek Zawirski, INRIA & UPMC, Paris, France

- Define data types in terms of commutative operations
- E.g., an Integer Counter CRDT with `add` and `sub`:

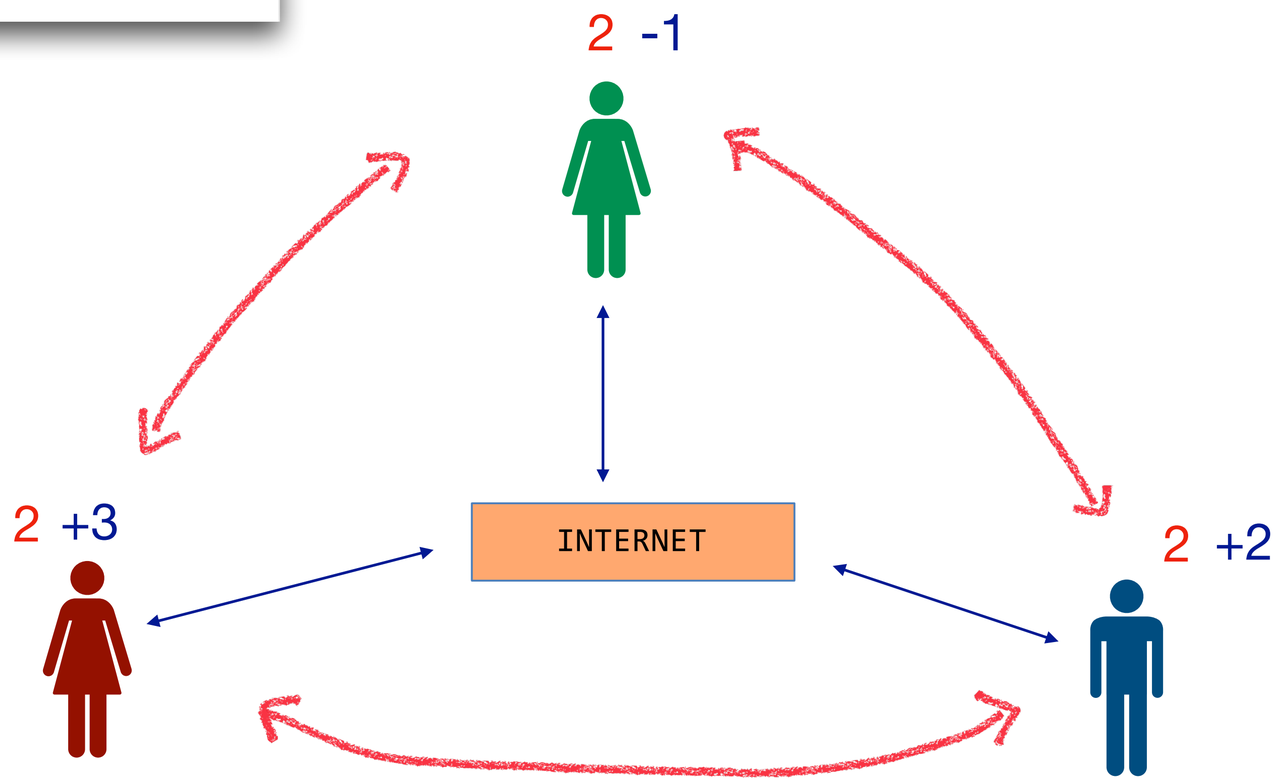


CRDTs

Conflict-free Replicated Data Types *

Marc Shapiro, INRIA & LIP6, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Carlos Baquero, Universidade do Minho, Portugal
Marek Zawirski, INRIA & UPMC, Paris, France

- Define data types in terms of commutative operations
- E.g., an Integer Counter CRDT with `add` and `sub`:

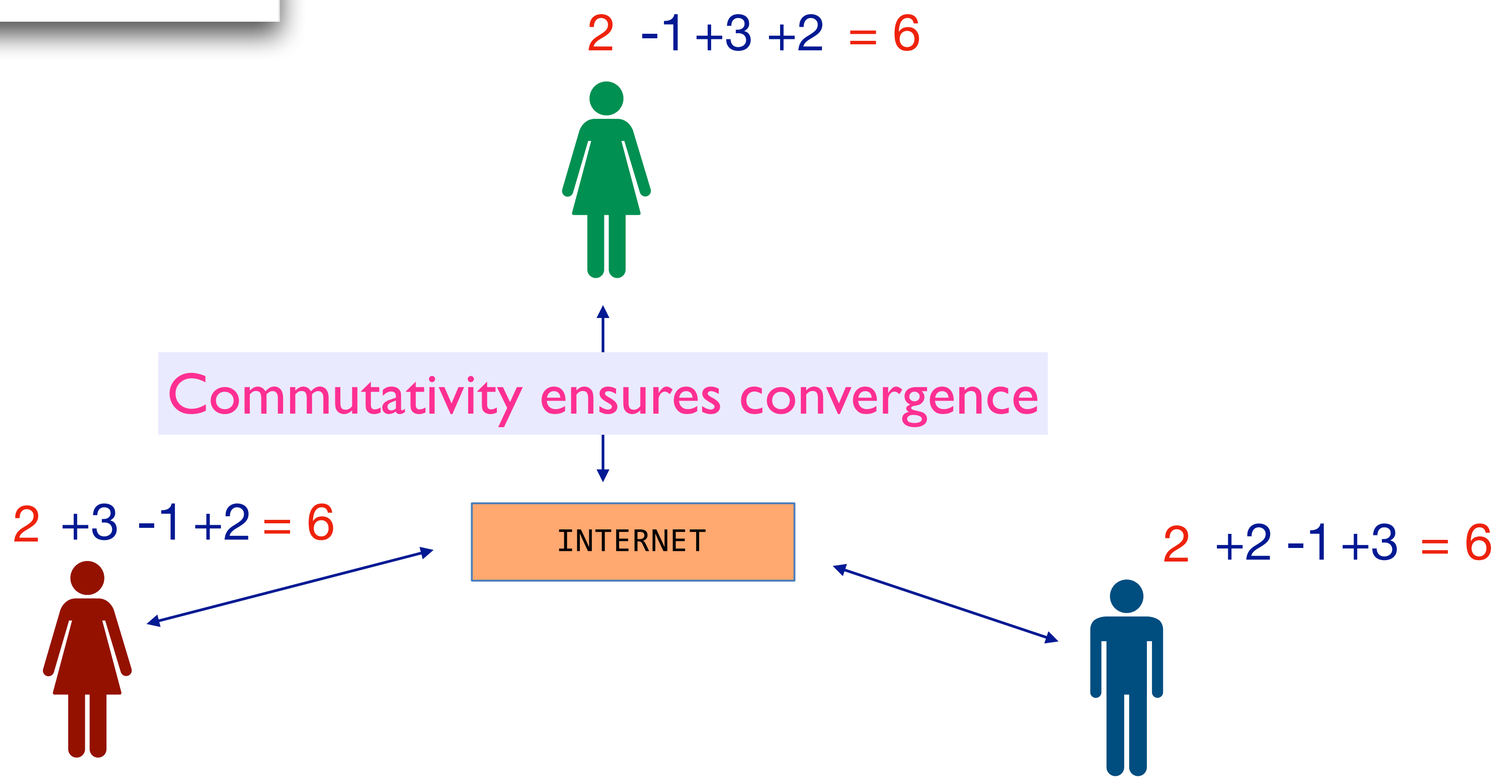


CRDTs

Conflict-free Replicated Data Types *

Marc Shapiro, INRIA & LIP6, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Carlos Baquero, Universidade do Minho, Portugal
Marek Zawirski, INRIA & UPMC, Paris, France

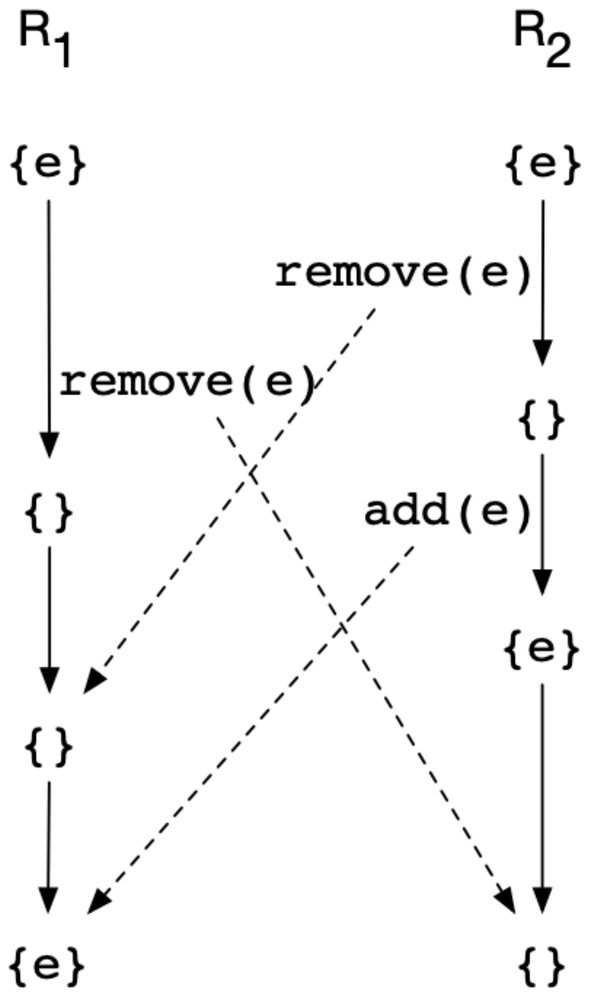
- Define data types in terms of commutative operations
- E.g., an Integer Counter CRDT with `add` and `sub`:



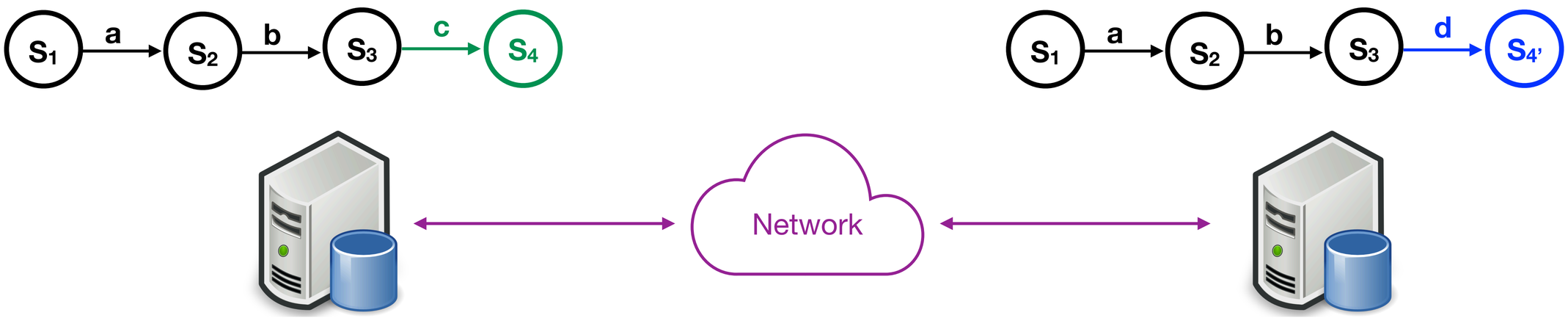
Commutativity is not common

E.g., a set data type with `add` and `remove`

- R_1 and R_2 start with a singleton set `{e}`
- Both execute `remove(e)`
- R_2 executes `add(e)`
- Eventually all operations are delivered to both replicas
- Since R_1's `remove` and R_2's `add` are concurrent, they are executed in different orders at R_1 and R_2
- Divergence!
- Solution: let one of `add` or `remove` "win" in case of concurrency (*add-wins* set or *remove-wins* set)
- Works for set but has limitations in general.

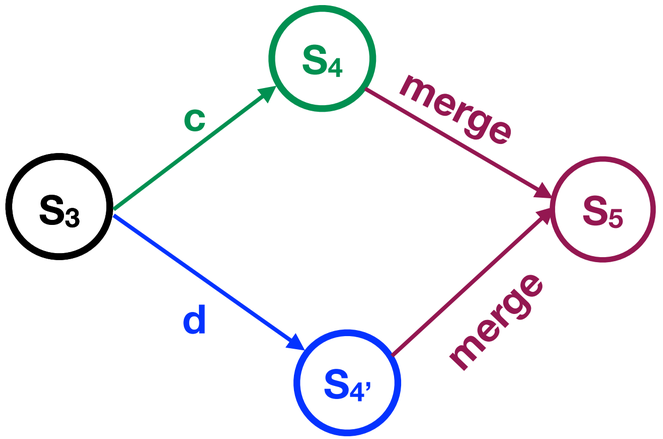


Divergence

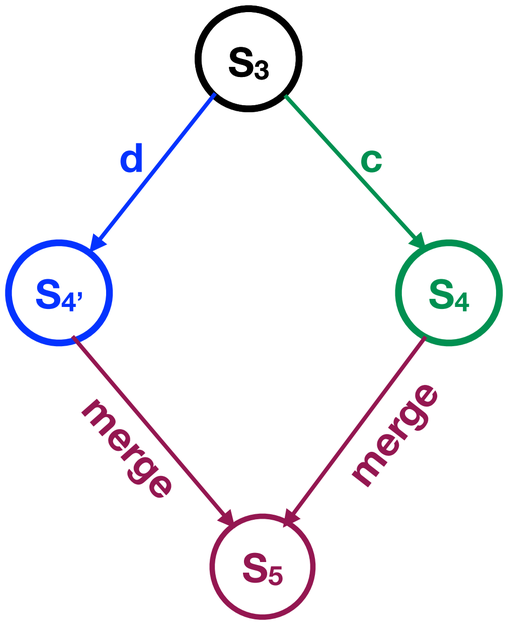


Qn: What to do If `c` & `d` are not commutative?

Divergence: Zoomed In

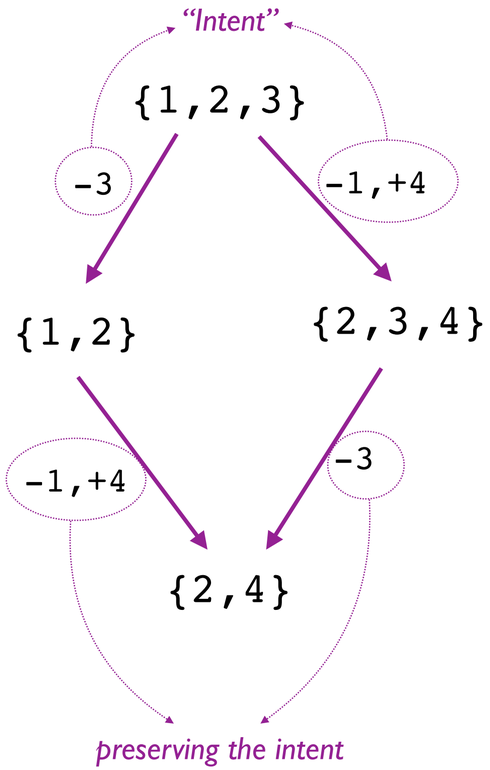


Merge



merge(S3 , S4' , S4) = S5

Set Merge



$$set\_merge(s, s_1, s_2) = (s \cap s_1 \cap s_2) \cup (s_1 - s) \cup (s_2 - s)$$

★ Merged set contains:

- ★ Elements in s that are not deleted in s_1 and s_2
- ★ Elements newly added in s_1
- ★ Elements newly added in s_2

Generality & Practicality of MRDTs

Mergeable Replicated Data Types

GOWTHAM KAKI, Purdue University, USA
SWARN PRIYA, Purdue University, USA
KC SIVARAMAKRISHNAN, IIT Madras, India
SURESH JAGANNATHAN, Purdue University, USA

[OOPSLA 2019]

Table 1. Characteristic relations for various data types

Data Type	Characteristic Relations
Binary Heap	Membership (R_{mem}), Ancestor ($R_{ans} \subseteq R_{mem} \times R_{mem}$)
Priority Queue	Membership (R_{mem})
Set	Membership (R_{mem})
Graph	Vertex (R_V), Edge (R_E)
Functional Map	Key-Value (R_{kv})
List	Membership (R_{mem}), Order (R_{ob})
Binary Tree	Membership (R_{mem}), Tree-order ($R_{to} \subseteq R_{mem} \times \texttt{label} \times R_{mem}$)
Binary Search Tree	Membership (R_{mem})

Generality & Practicality of MRDTs

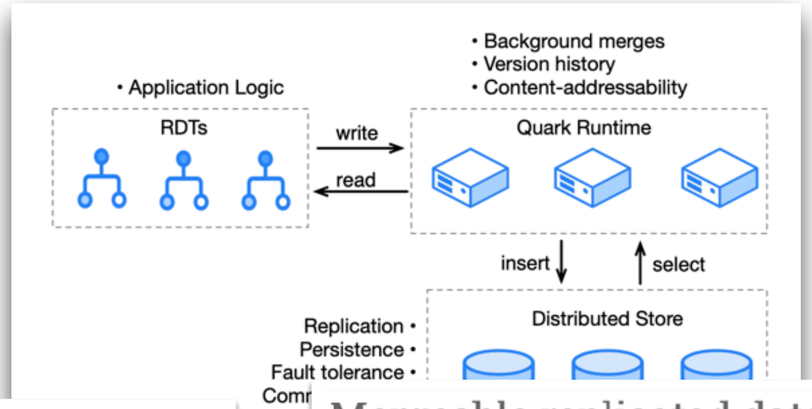
RunTime-Assisted Convergence in Replicated Data Types

Gowtham Kaki
University of Colorado Boulder
Boulder, USA
gowtham.kaki@colorado.edu

Prasanth Prahladan
University of Colorado Boulder
Boulder, USA
prasanth.prahladan@colorado.edu

Nicholas V. Lewchenko
University of Colorado Boulder
Boulder, USA
nile1033@colorado.edu

[PLDI 2022]



Y **Hacker News** new | threads | past | comments | ask | show

Mergeable replicated data types – Part I (acolyer.org)

104 points by telotortium 75 days ago | hide | past | web | un-favorite | 21 comments

Mergeable replicated data types – Part I

NOVEMBER 25, 2019

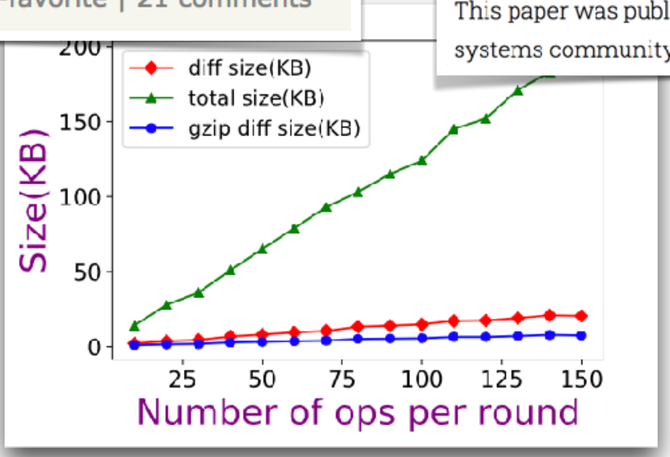
Mergeable replicated data types Kaki et al., *OOPSLA'19*

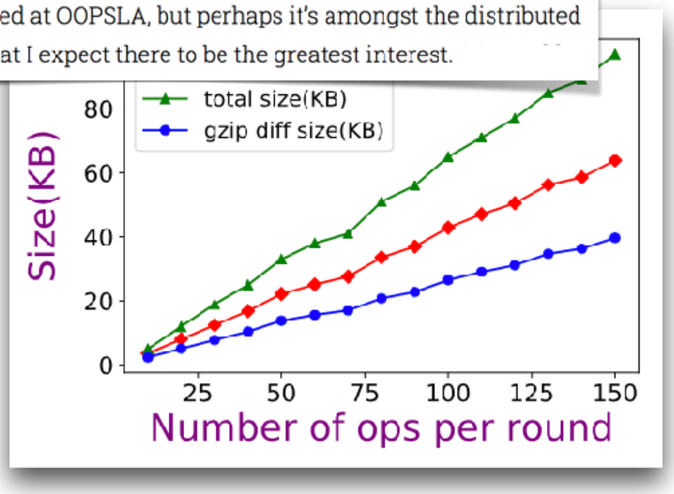
This paper was published at OOPSLA, but perhaps it's amongst the distributed systems community that I expect there to be the greatest interest.

Mergeable Replicated Data Types

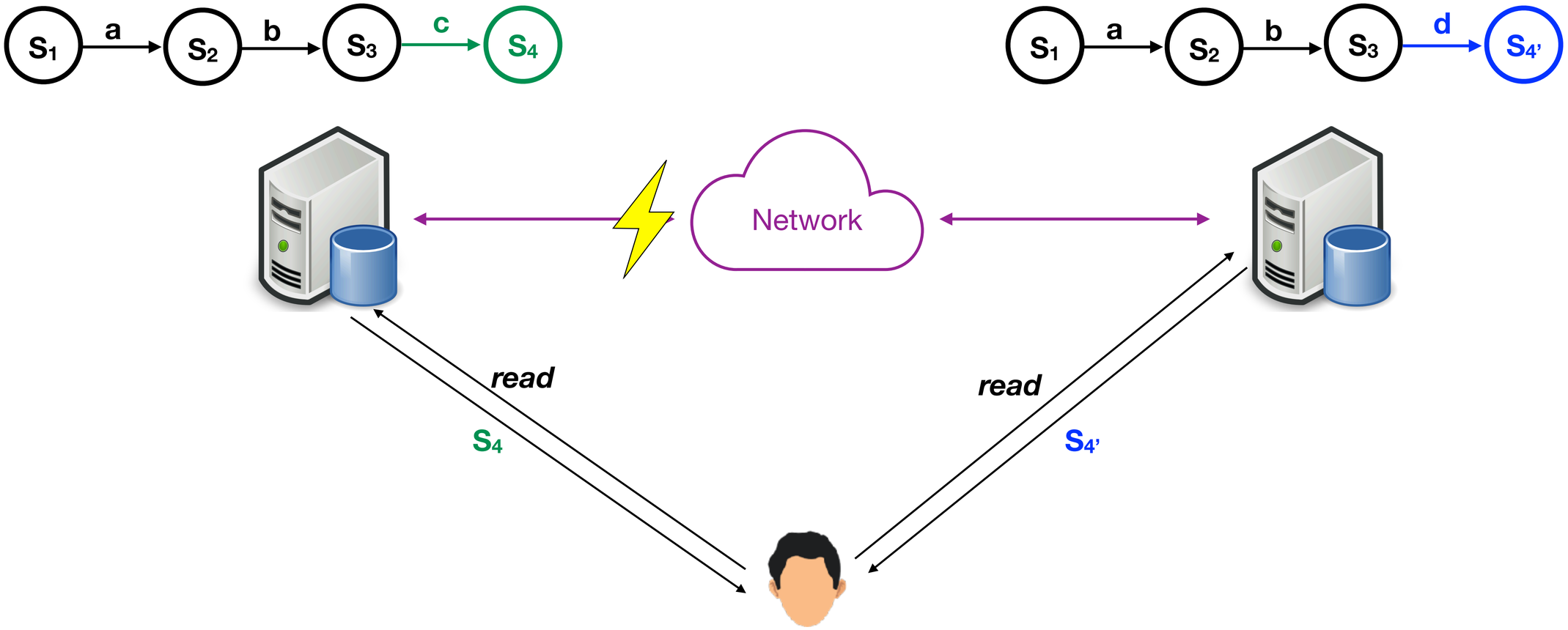
GOWTHAM KAKI, Purdue University, USA
SWARN PRIYA, Purdue University, USA
KC SIVARAMAKRISHNAN, IIT Madras, India
SURESH JAGANNATHAN, Purdue University, USA

[OOPSLA 2019]

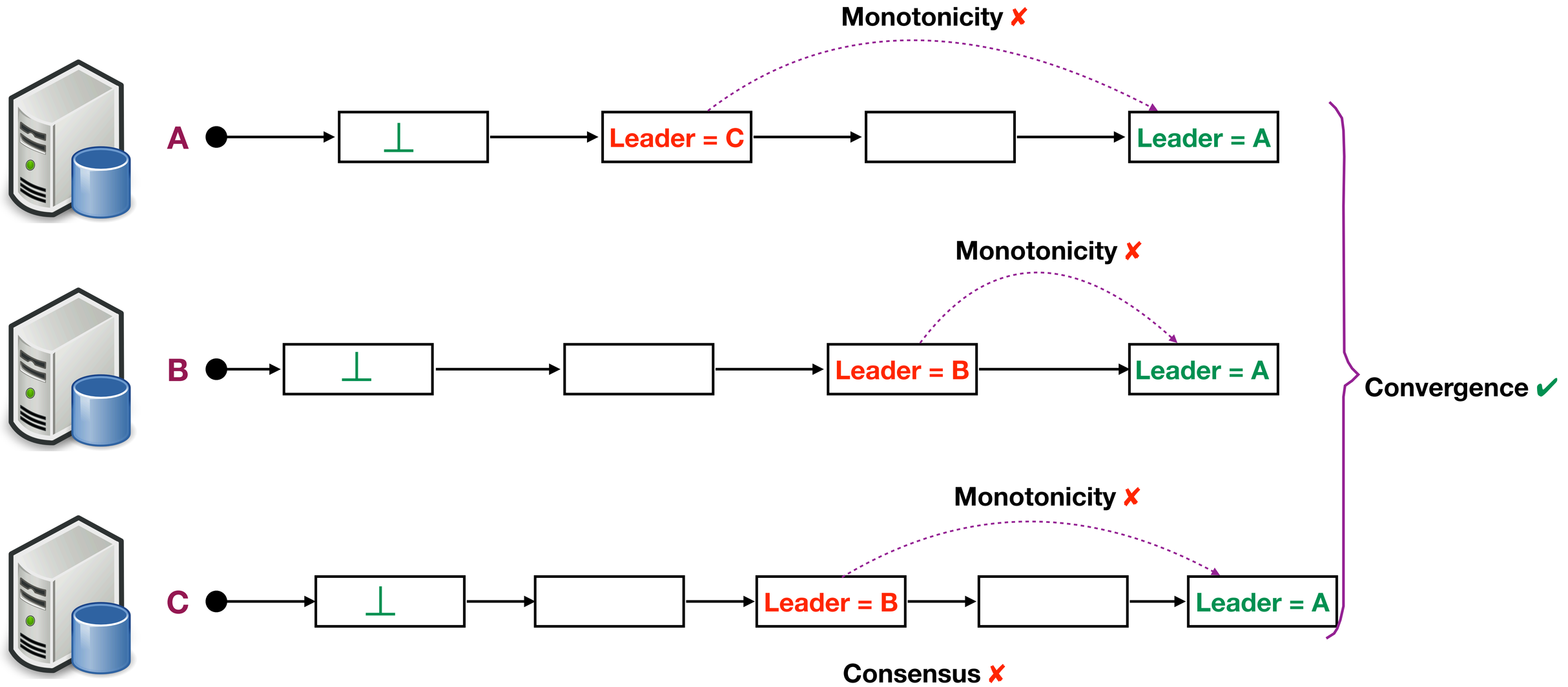




Convergence ≠ Consensus



Example: Leader Election



Convergence is half-way to consensus

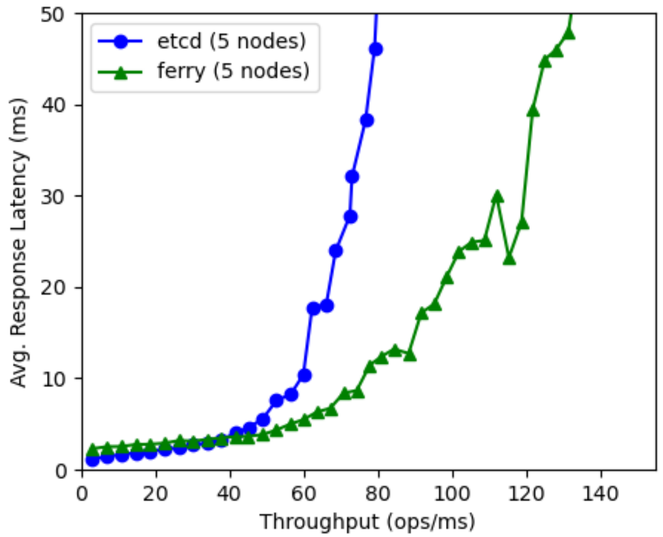
Convergence + Monotonicity ⇒ Consensus

Bolt-On Strong Consistency: Specification, Implementation, and Verification

NICHOLAS V. LEWCHENKO, University of Colorado Boulder, USA
GOWTHAM KAKI, University of Colorado Boulder, USA
BOR-YUH EVAN CHANG*, University of Colorado Boulder, USA and Amazon, USA

[OOPSLA 2025]

- Super-V: A Haskell DSL to implement and formally verify distributed protocols.



- Automated formal verification of Paxos and Raft Implementations with flexible quorum optimization.
- Raft implementation in Super-V performs similarly or better than `etcd-raft`

Conclusion

- Distributed consensus is expensive and hard to get right.
- Convergence is weaker than consensus, but cheap and easy to implement. It can substitute consensus in surprisingly many use cases.
  - Mergeable Replicated Data Types (MRDTs)
  - Quark: https://github.com/cuplv/Quark
- When consensus is unavoidable, it can be implemented *on top of* convergence. This approach results in efficient implementations that can be formally verified with little effort.
  - Super-V: https://github.com/cuplv/super-v