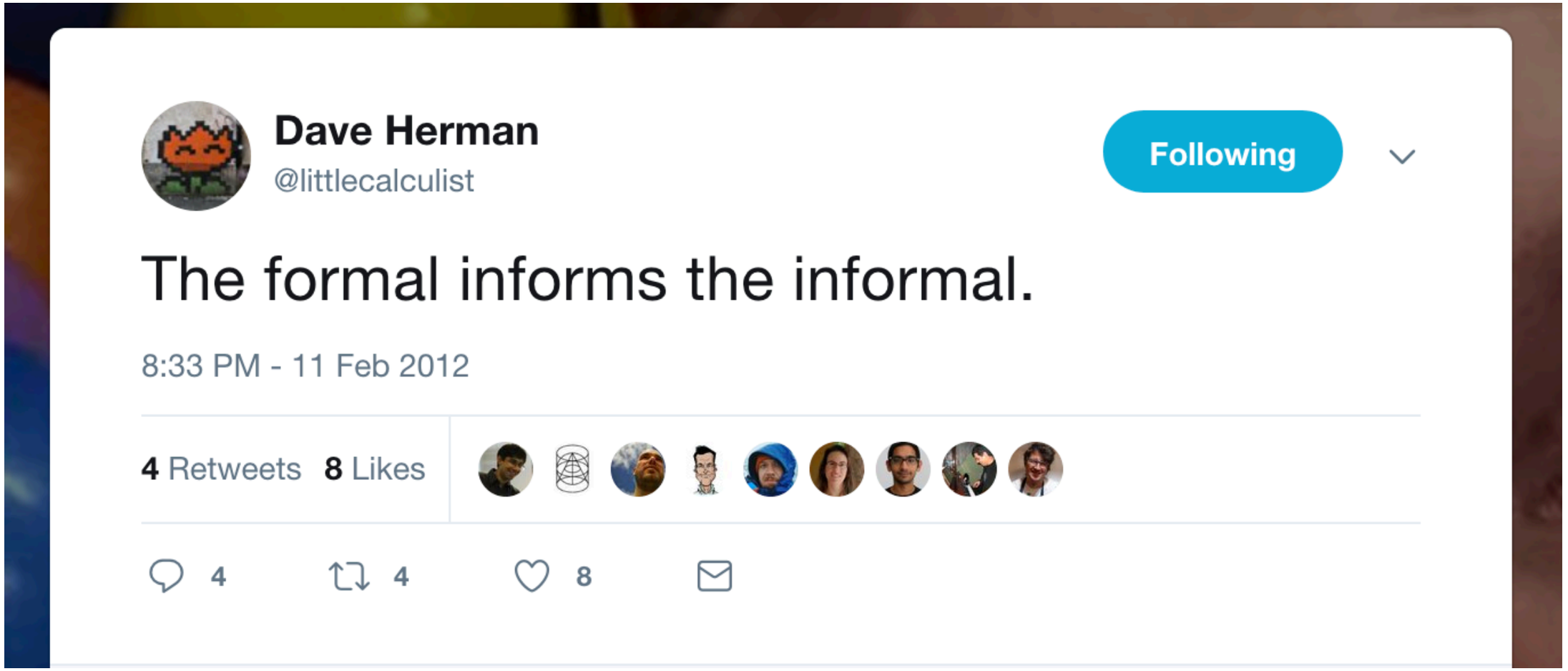
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services

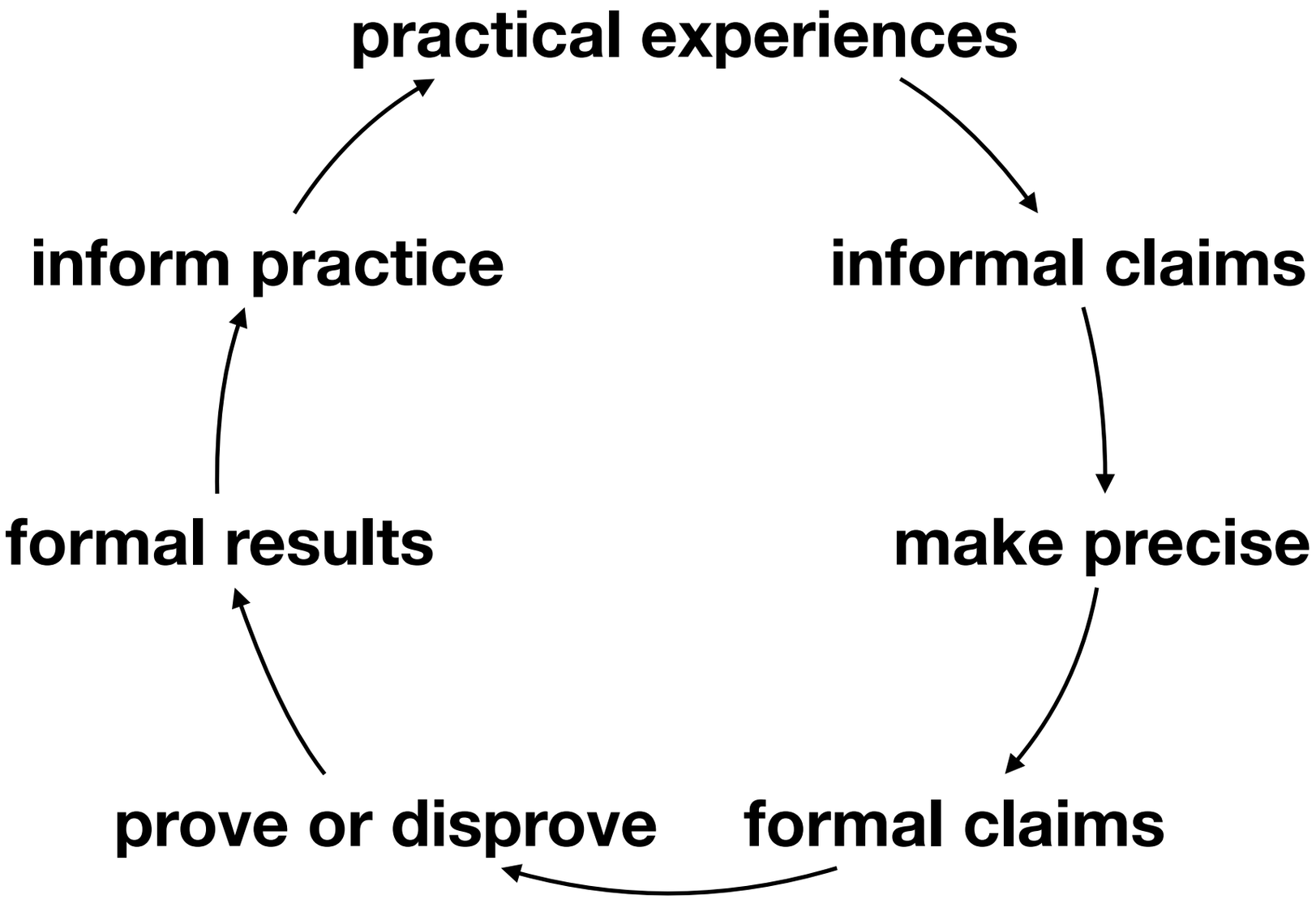
CMPS290S, Fall 2018

Lindsey Kuper

October 1, 2018

first, some context




practical experiences
informal claims
make precise
formal claims
prove or disprove
formal results
inform practice

what’s with this paper?

- no attempt to be self-contained
- doesn’t define all its terms
- no related work
- skimpy citations (and half are self-citations)
- non-peer-reviewed venue (SIGACT News)

and yet!

- enormously influential
  - ~1700 citations
  - Lynch's third most cited work *(which is saying something)*
- partly responsible for popularizing the term "CAP theorem"
- great example of "the formal informs the informal"

Nancy Lynch

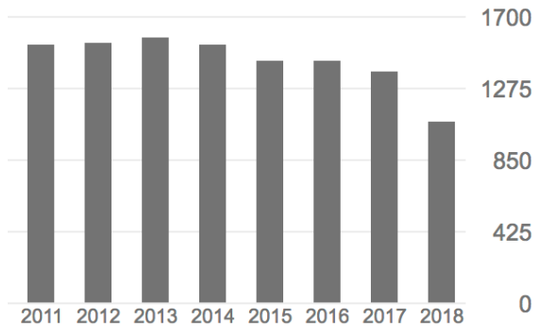
FOLLOW

Professor of EECS, MIT
Verified email at csail.mit.edu

Distributed computing theor… Modeling and verification Wireless networks Biological algorithms

Cited by	All	Since 2013
Citations	34172	8453
h-index	74	32
i10-index	255	114

VIEW ALL



TITLE	CITED BY	YEAR
Distributed algorithms NA Lynch Elsevier	5622	1996
Impossibility of distributed consensus with one faulty process MJ Fischer, NA Lynch, MS Paterson Journal of the ACM (JACM) 32 (2), 374-382	4647	1985
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services S Gilbert, N Lynch Acm Sigact News 33 (2), 51-59	1700	2002
An introduction to input/output automata NA Lynch, MR Tuttle Laboratory for Computer Science, Massachusetts Institute of Technology	1423	1988
Consensus in the presence of partial synchrony C Dwork, N Lynch, L Stockmeyer Journal of the ACM (JACM) 35 (2), 288-323	1359	1988
Hierarchical correctness proofs for distributed algorithms NA Lynch, MR Tuttle Proceedings of the sixth annual ACM Symposium on Principles of distributed …	1065	1987
Hybrid i/o automata N Lynch, R Segala, F Vaandrager, HB Weinberg International Hybrid Systems Workshop, 496-510	750 *	1995
Probabilistic simulations for probabilistic processes R Segala, N Lynch Nordic Journal of Computing 2 (2), 250-273	512	1995
Forward and backward simulations N Lynch, F Vaandrager Information and Computation 121 (2), 214-233	501	1995
A comparison of polynomial time reducibilities RE Ladner, NA Lynch, AL Selman Theoretical Computer Science 1 (2), 103-123	490	1975
A lower bound for the time to assure interactive consistency MJ Fischer, NA Lynch GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUTER SCIENCE	482	1981
Hybrid i/o automata	453	2003

Nancy Lynch

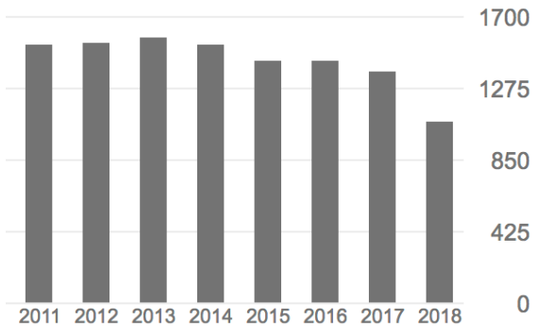
Professor of EECS, MIT
Verified email at csail.mit.edu

Distributed computing theor… Modeling and verification Wireless networks Biological algorithms

FOLLOW

Cited by VIEW ALL

	All	Since 2013
Citations	34172	8453
h-index	74	32
i10-index	255	114



TITLE	CITED BY	YEAR
Distributed algorithms NA Lynch Elsevier	5622	1996
Impossibility of distributed consensus with one faulty process MJ Fischer, NA Lynch, MS Paterson Journal of the ACM (JACM) 32 (2), 374-382	4647	1985
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services S Gilbert, N Lynch Acm Sigact News 33 (2), 51-59	1700	2002
An introduction to input/output automata NA Lynch, MR Tuttle Laboratory for Computer Science, Massachusetts Institute of Technology	1423	1988
Consensus in the presence of partial synchrony C Dwork, N Lynch, L Stockmeyer Journal of the ACM (JACM) 35 (2), 288-323	1359	1988
Hierarchical correctness proofs for distributed algorithms NA Lynch, MR Tuttle Proceedings of the sixth annual ACM Symposium on Principles of distributed …	1065	1987
Hybrid i/o automata N Lynch, R Segala, F Vaandrager, HB Weinberg International Hybrid Systems Workshop, 496-510	750 *	1995
Probabilistic simulations for probabilistic processes R Segala, N Lynch Nordic Journal of Computing 2 (2), 250-273	512	1995
Forward and backward simulations N Lynch, F Vaandrager Information and Computation 121 (2), 214-233	501	1995
A comparison of polynomial time reducibilities RE Ladner, NA Lynch, AL Selman Theoretical Computer Science 1 (2), 103-123	490	1975
A lower bound for the time to assure interactive consistency MJ Fischer, NA Lynch GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUTER SCIENCE	482	1981
Hybrid i/o automata	453	2003

Wyatt Lloyd et al.,
Don’t Settle for Eventual: Scalable Causal Consistency... (SOSP ‘11)

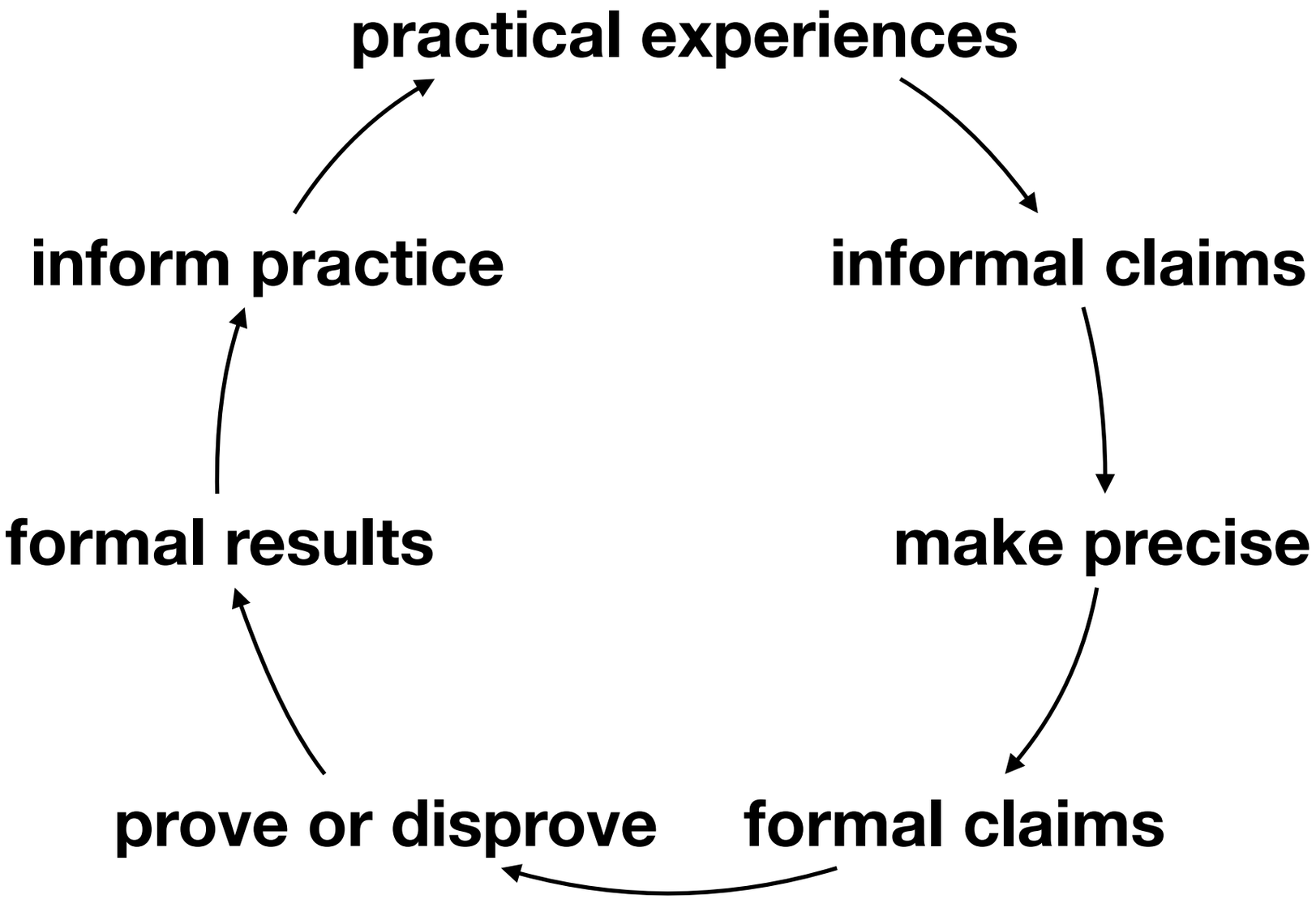
Distributed data stores are a fundamental building block of modern Internet services. Ideally, these data stores would be strongly consistent, always available for reads and writes, and able to continue operating during network partitions. The CAP Theorem, unfortunately, proves it impossible to create a system that achieves all three [13, 23]. Instead, modern web services have chosen overwhelmingly to embrace availability and partition tolerance at the cost of strong consistency [16, 20, 30]. This is perhaps not surpris-

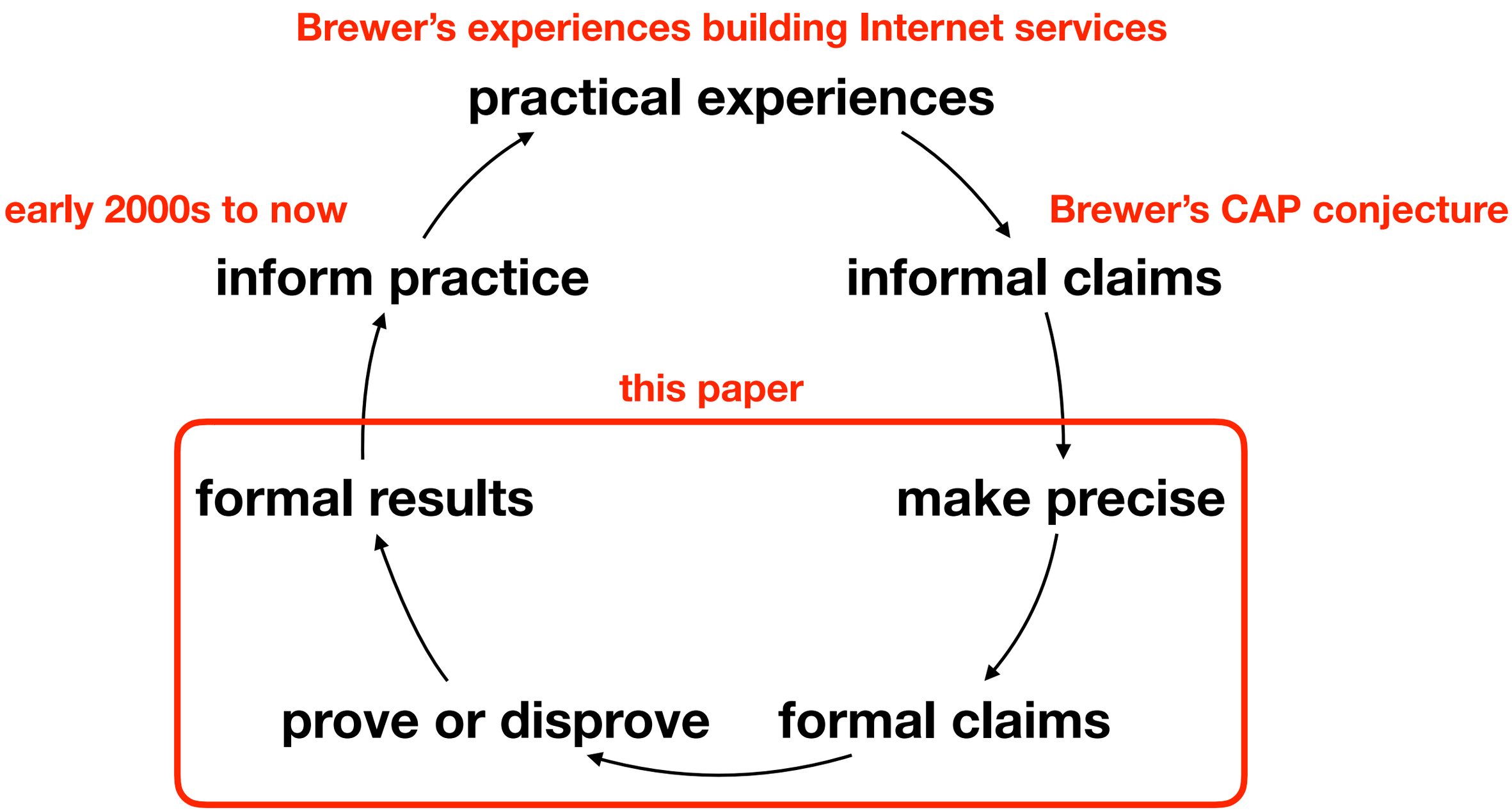
Mohsen Lesani, Christian J. Bell, and Adam Chlipala,
Chapar: Certified Causally Consistent Distributed Key-Value Stores (POPL ‘16)

On the flip side, maintaining strong consistency across replicas [30] can limit parallelism [35] and availability. When availability is a must, the CAP theorem [19] formulates a fundamental trade-off between strong consistency and partition tolerance, and PACELC [3] formulates a trade-off between strong consistency and latency [5]. In reaction to these constraints, modern storage

Gowtham Kaki et al.,
Safe Replication through Bounded Concurrency Verification (OOPSLA ‘18)

Replicated data types (RDTs) are a common abstraction used to program sophisticated distributed applications intended to operate in geo-replicated environments [Burckhardt et al. 2014]. These environments often expose a storage layer in which different replicas may hold different states of the same RDT instance. This occurs because RDT operations may not be atomically and uniformly applied across all replicas, as a consequence of network partitions and failures, weak ordering and delivery guarantees, etc. [Brewer 2000; Gilbert and Lynch 2002] Consequently, replicas may hold


practical experiences
informal claims
make precise
formal claims
prove or disprove
formal results
inform practice


Brewer's experiences building Internet services
practical experiences
early 2000s to now
Brewer's CAP conjecture
inform practice
informal claims
this paper
formal results
make precise
prove or disprove
formal claims

2012 follow-up work

- Brewer's CAP Twelve Years Later (Wednesday's reading!)
- Gilbert and Lynch's own, less well-known follow-up

Perspectives on the CAP Theorem

Seth Gilbert
National University of Singapore

Nancy A. Lynch
Massachusetts Institute of Technology

Abstract

Almost twelve years ago, in 2000, Eric Brewer introduced the idea that there is a fundamental trade-off between *consistency*, *availability*, and *partition tolerance*. This trade-off, which has become known as the *CAP Theorem*, has been widely discussed ever since. In this paper, we review the CAP Theorem and situate it within the broader context of distributed computing theory. We then discuss the practical implications of the CAP Theorem, and explore some general techniques for coping with the inherent trade-offs that it implies.

1 Introduction

Almost twelve years ago, in 2000, Eric Brewer introduced the idea that there is a fundamental trade-off between *consistency*, *availability*, and *partition tolerance*. This trade-off, which has become known as the *CAP Theorem*, has been widely discussed ever since.

Theoretical context. Our first goal in this paper is to situate the CAP Theorem in the broader context of distributed computing theory. Some of the interest in the CAP Theorem, perhaps, derives from the fact that it illustrates a more general trade-off that appears everywhere (e.g., [4,7,15,17,23]) in the study of distributed computing: the impossibility of guaranteeing both *safety* and *liveness* in an *unreliable distributed system*:

Safety: Informally, an algorithm is *safe* if nothing bad ever happens. A quiet, uneventful room is perfectly safe. Consistency (as defined in the CAP Theorem) is a classic safety property: every response sent to a client is correct.

Liveness: By contrast, an algorithm is *live* if eventually something good happens. In a busy pub, there may be some good things happening, and there may be some bad things happening—but overall, it is quite lively. Availability is a

Brewer's conjecture

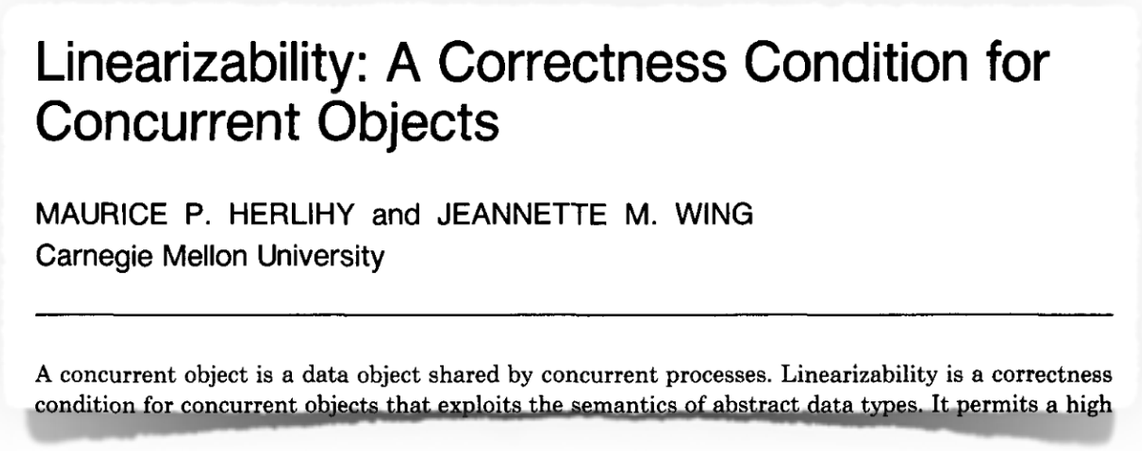
Impossible for a web service to simultaneously guarantee:

- consistency
- availability
- partition tolerance

The paper makes this claim more precise…

(atomic) consistency

- == Herlihy and Wing's *linearizability* (Friday's reading!)



Linearizability: A Correctness Condition for Concurrent Objects

MAURICE P. HERLIHY and JEANNETTE M. WING
Carnegie Mellon University

A concurrent object is a data object shared by concurrent processes. Linearizability is a correctness condition for concurrent objects that exploits the semantics of abstract data types. It permits a high

- as if all the operations run on a single server [Gilbert and Lynch 2012]

- "each server returns the *right* response to each request" [Gilbert and Lynch 2012]

availability

- “each request [must] eventually receive a response” [Gilbert and Lynch 2012]

- a “request” can be either a read or a write

partition tolerance

- “messages may be delayed and, sometimes, lost forever” [Gilbert and Lynch 2012]

- property of the underlying network, not of the service

the paper’s Theorem 1

“It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:

- Availability
- Atomic consistency

in all fair executions (including those in which messages are lost).”

the paper's Theorem 1

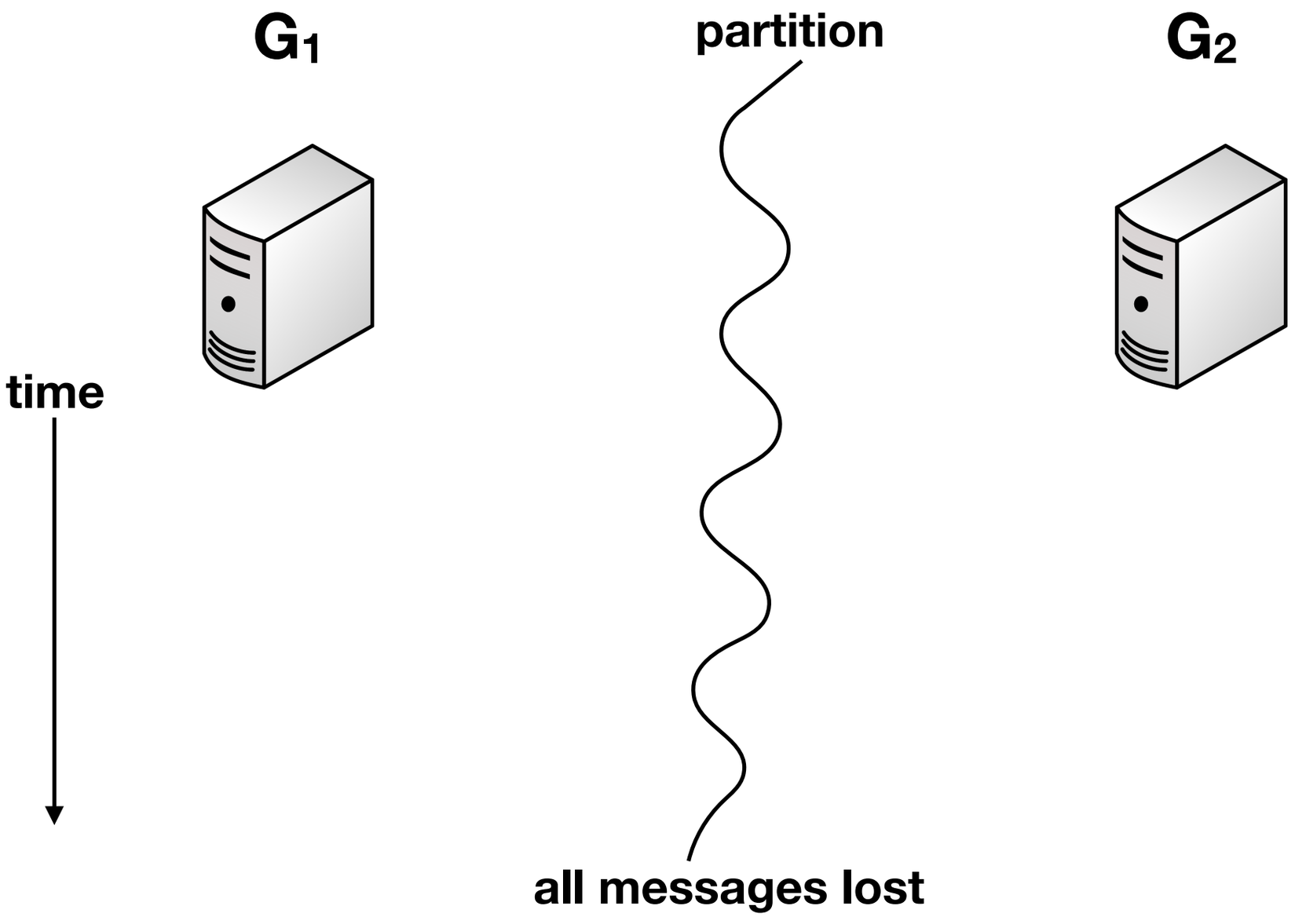
"It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:

- Availability
- Atomic consistency

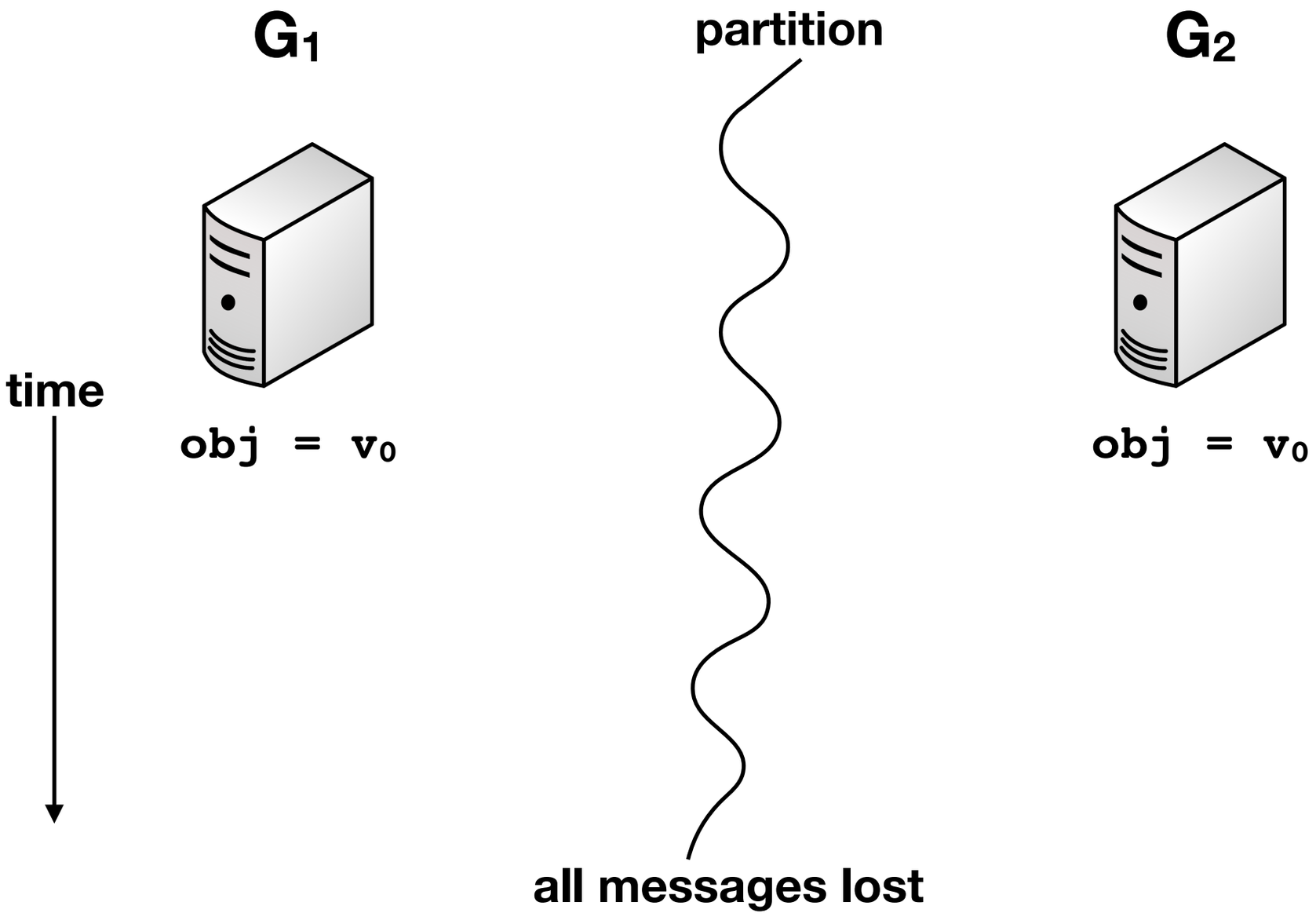
in all fair executions (including those in which messages are lost)."

(which terms did the paper not define? I see a few…)

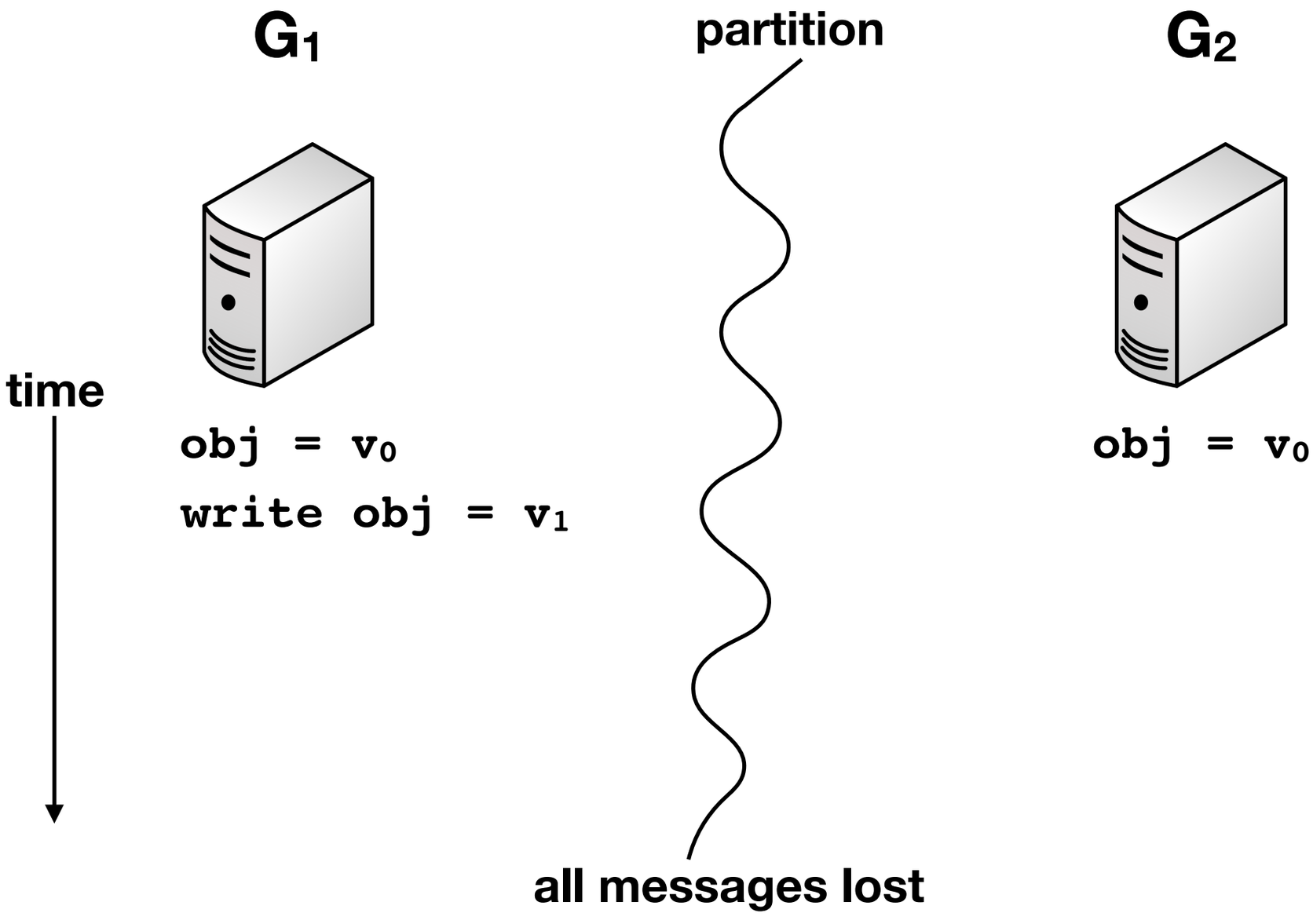
proof sketch



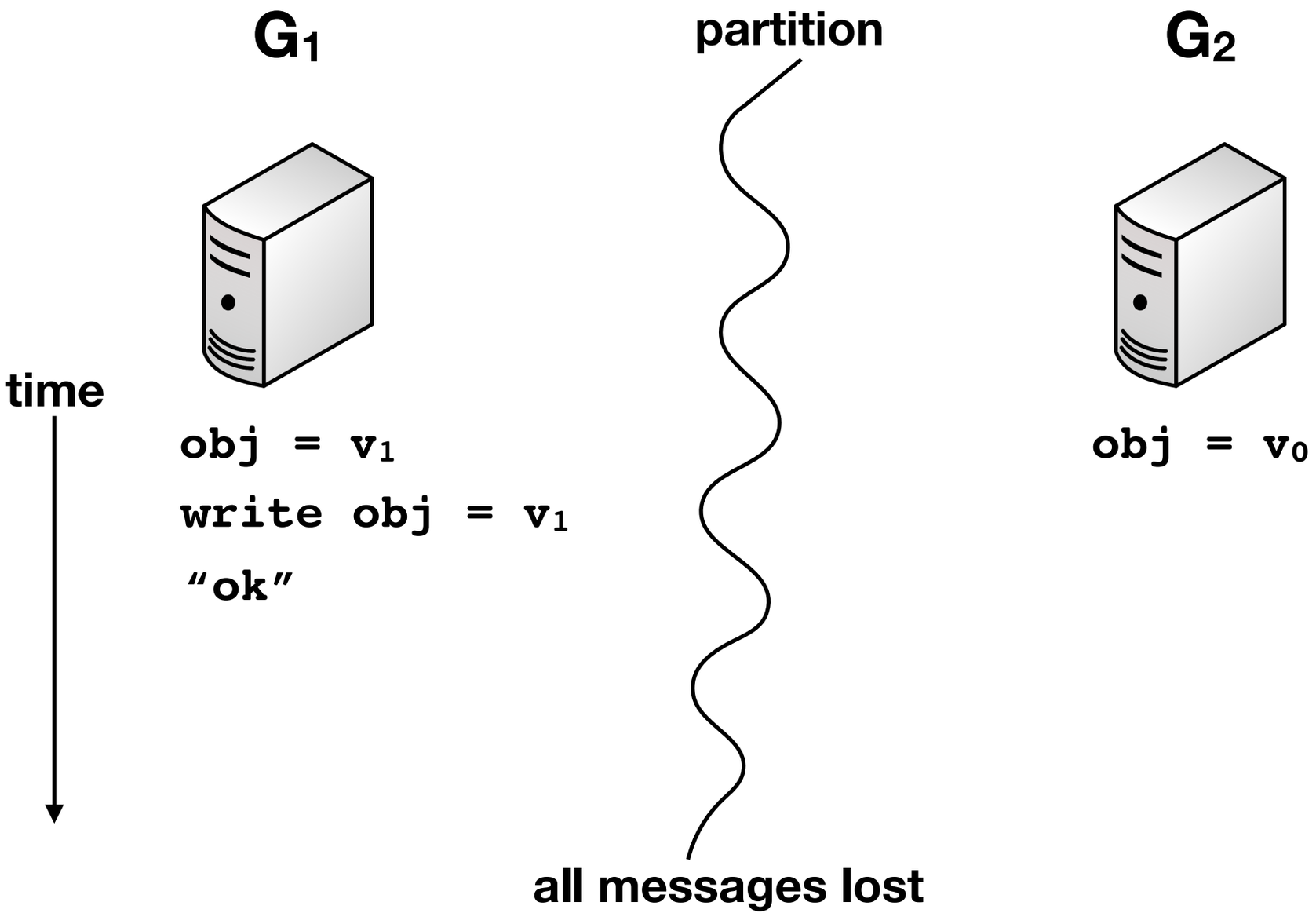
proof sketch



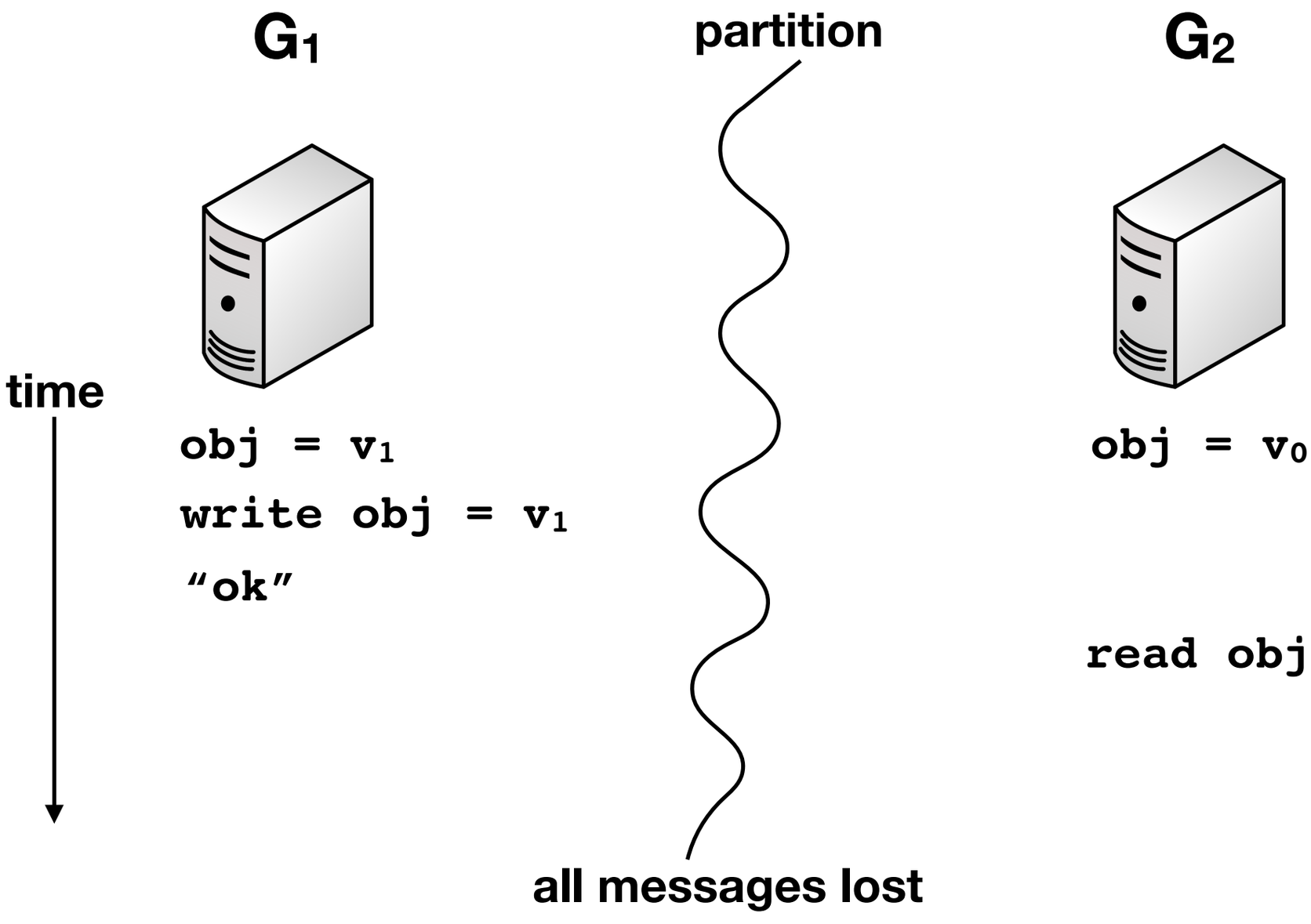
proof sketch



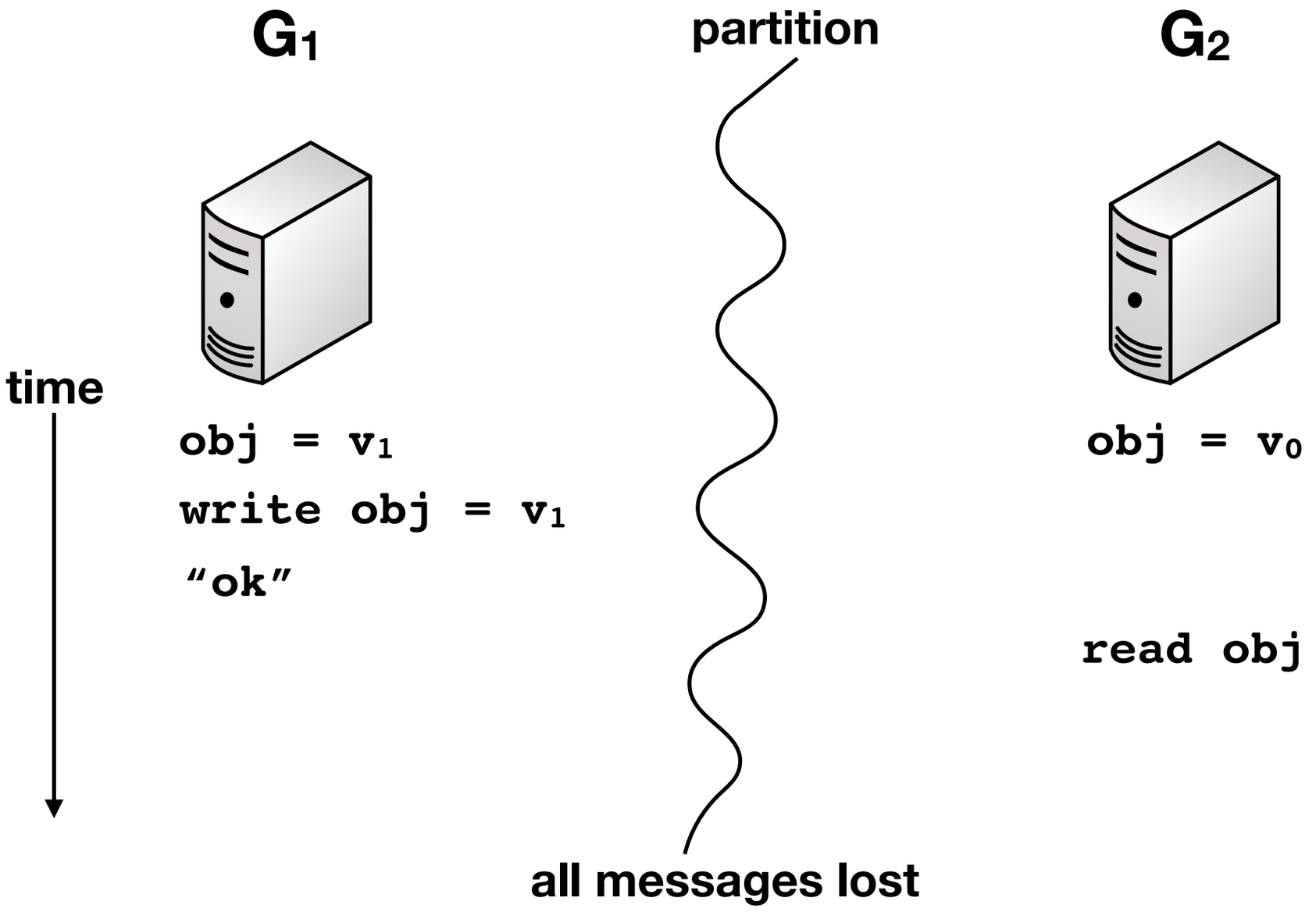
proof sketch



proof sketch



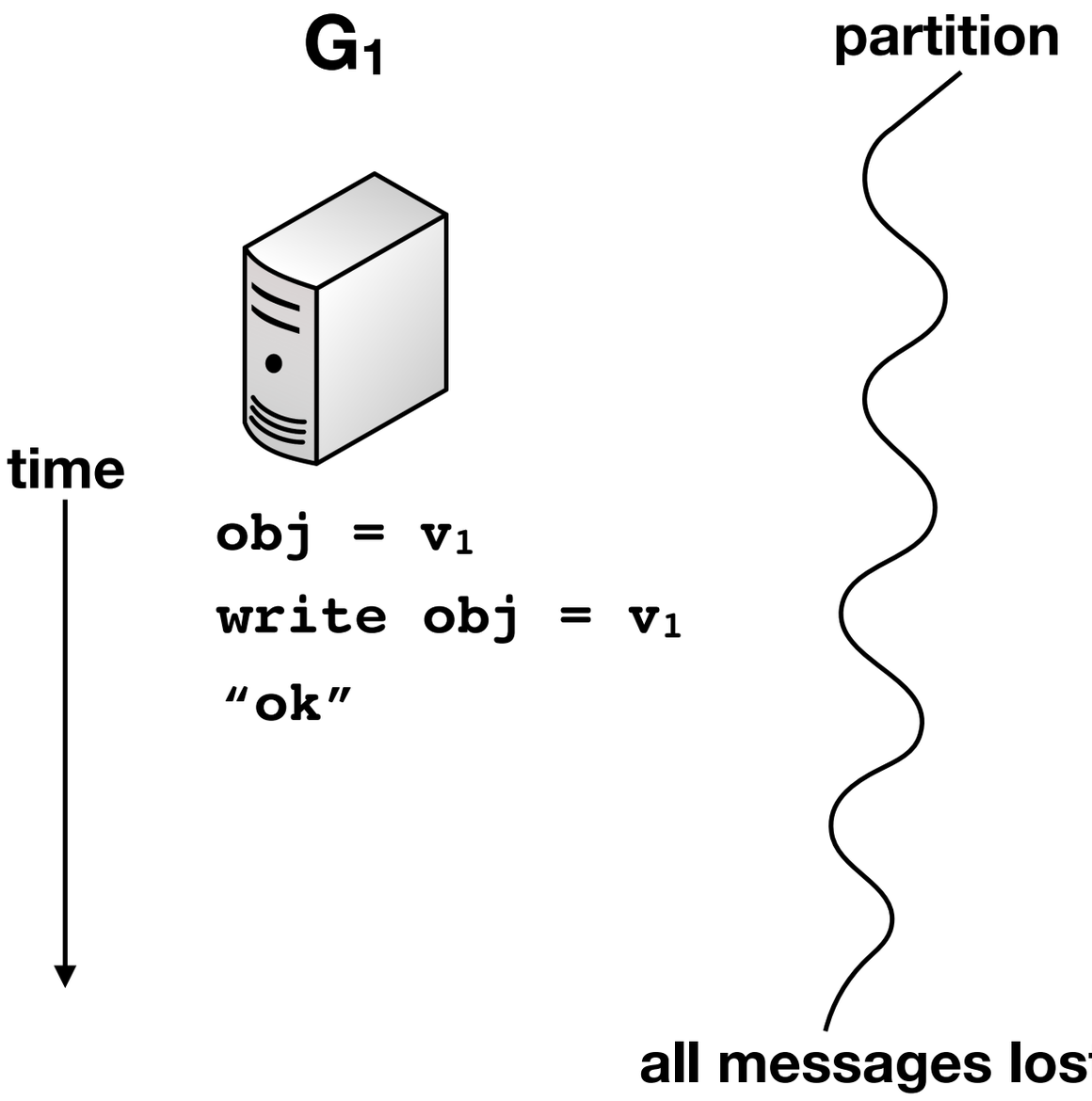
proof sketch



two options:

- wait for partition to heal (sacrificing availability)
- return current value (sacrificing consistency)

proof sketch



two options:

- wait for partition to heal (sacrificing availability)
- return current value (sacrificing consistency)

assumptions

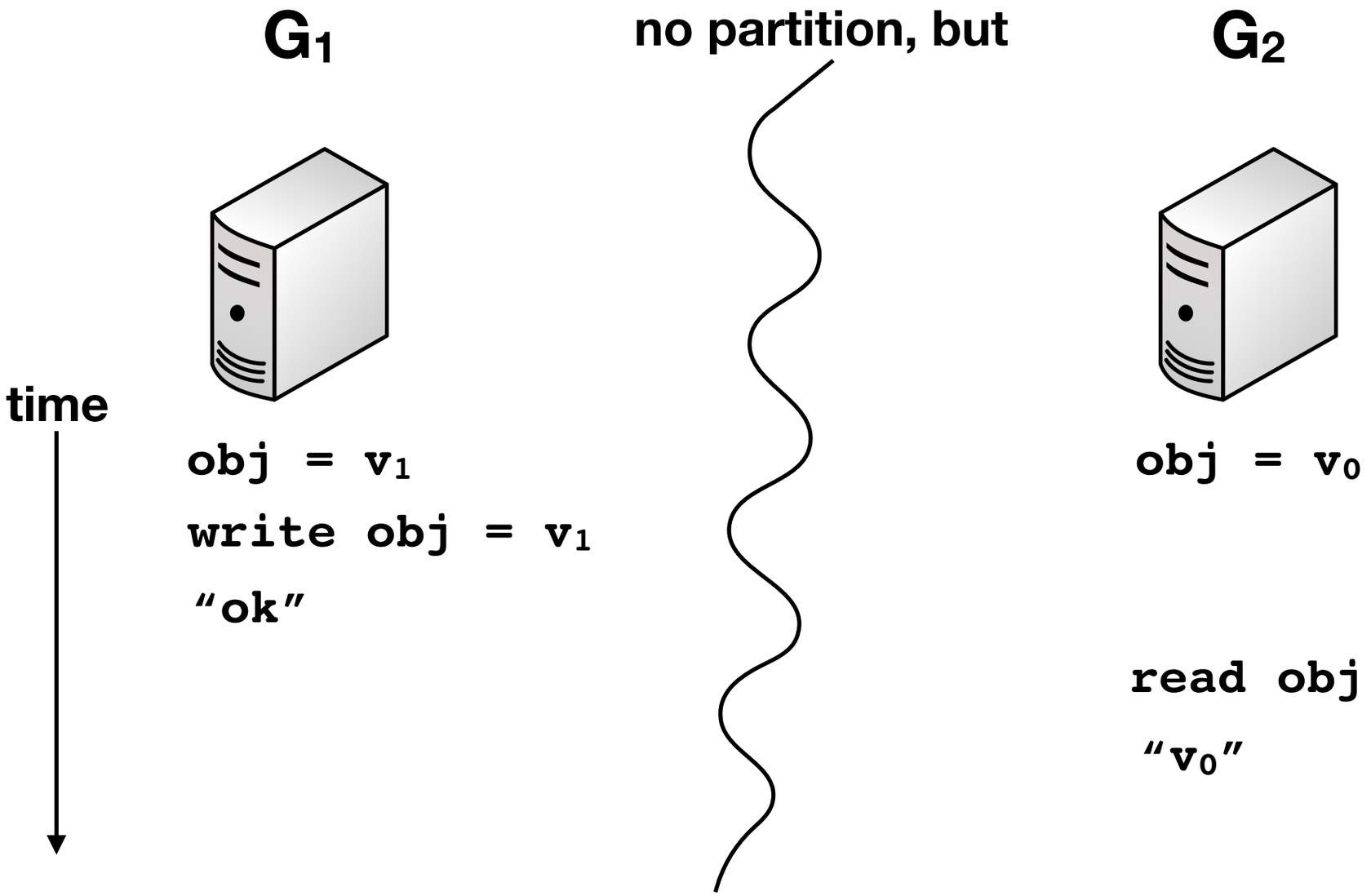
1. all messages between G_1 and G_2 are lost

2. no other client requests occur in G_1 or G_2

why are these assumptions OK to make?

why is assumption 1 actually unnecessary (Corollary 1.1)?

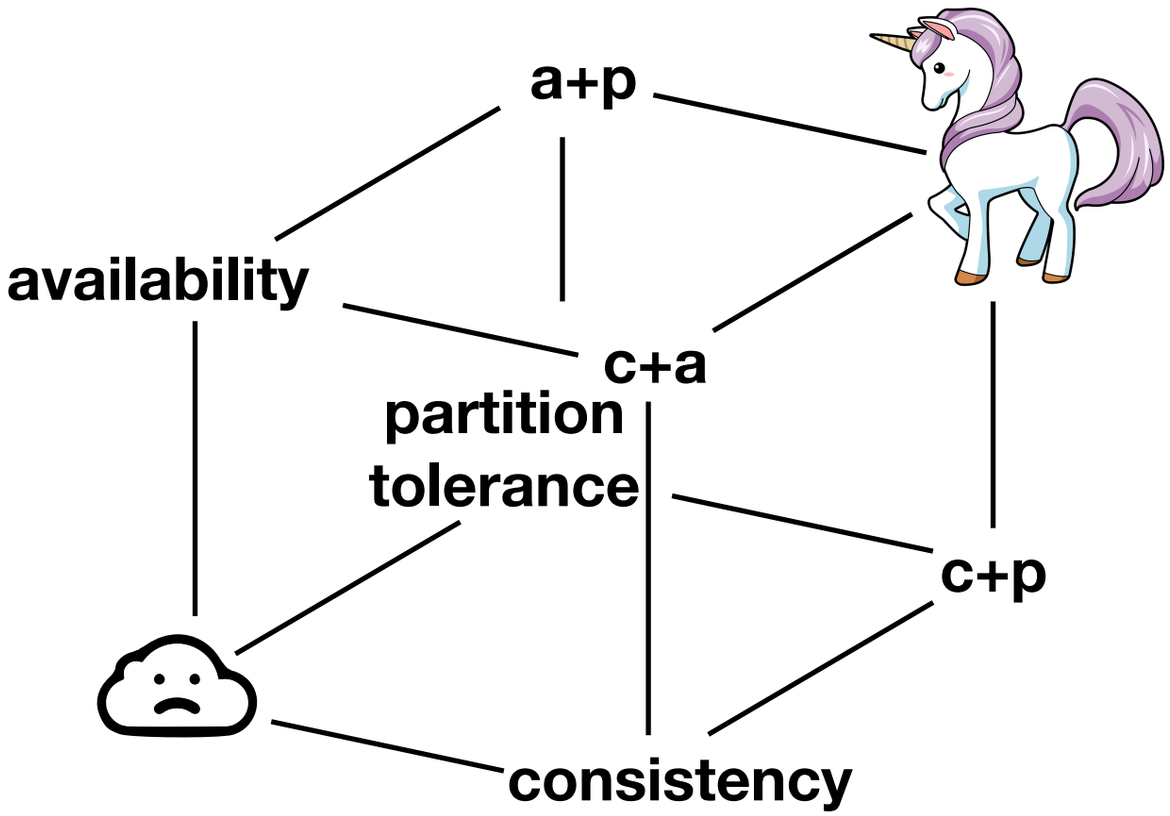
the paper's Corollary 1.1



two options:

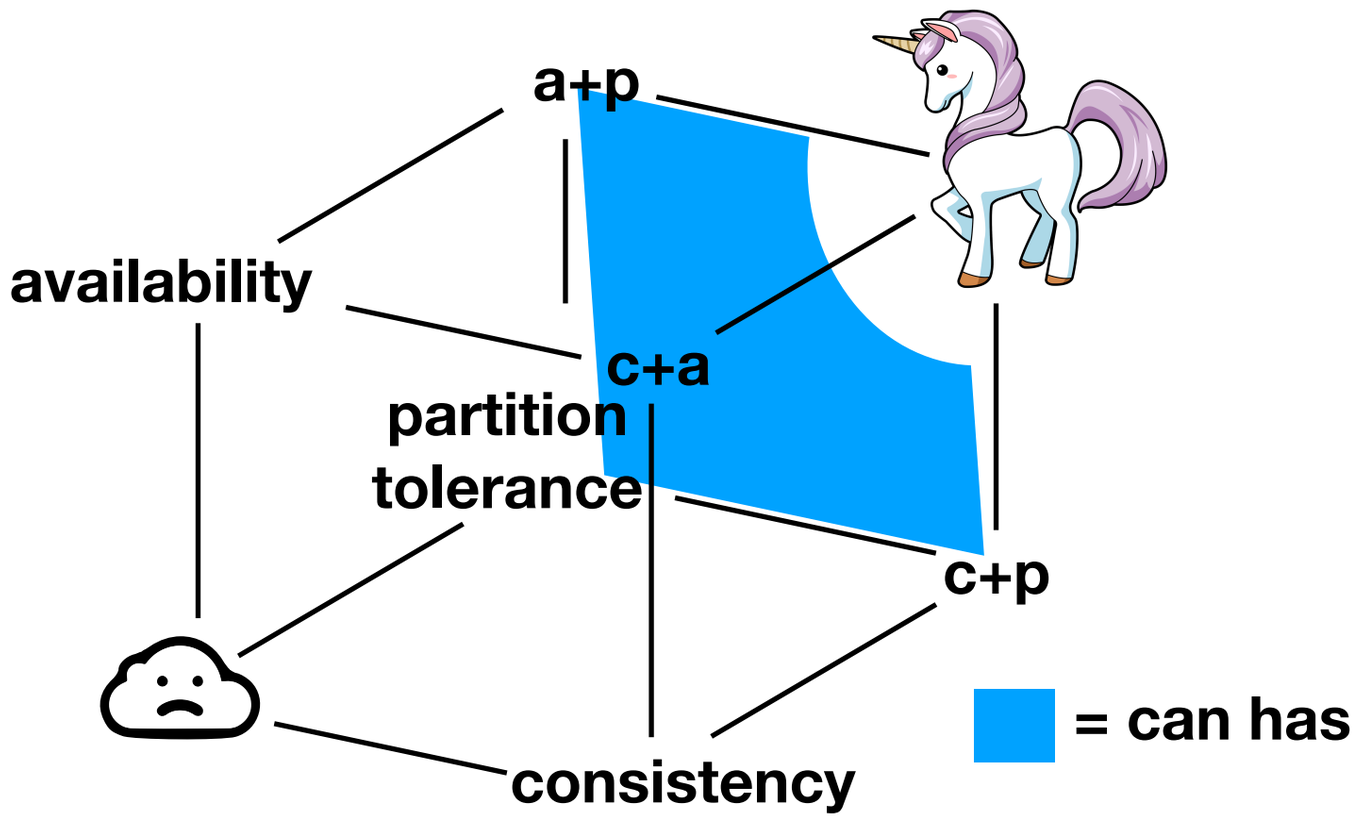
- wait for potential update from G_1 (sacrificing availability)
- return current value (sacrificing consistency)

pick two?



- the original paper would have you believe so…
- but in practice, partitions occur (so you can't really pick c+a)
- and moreover, it's not actually a binary decision, anyway
- see Wednesday's reading

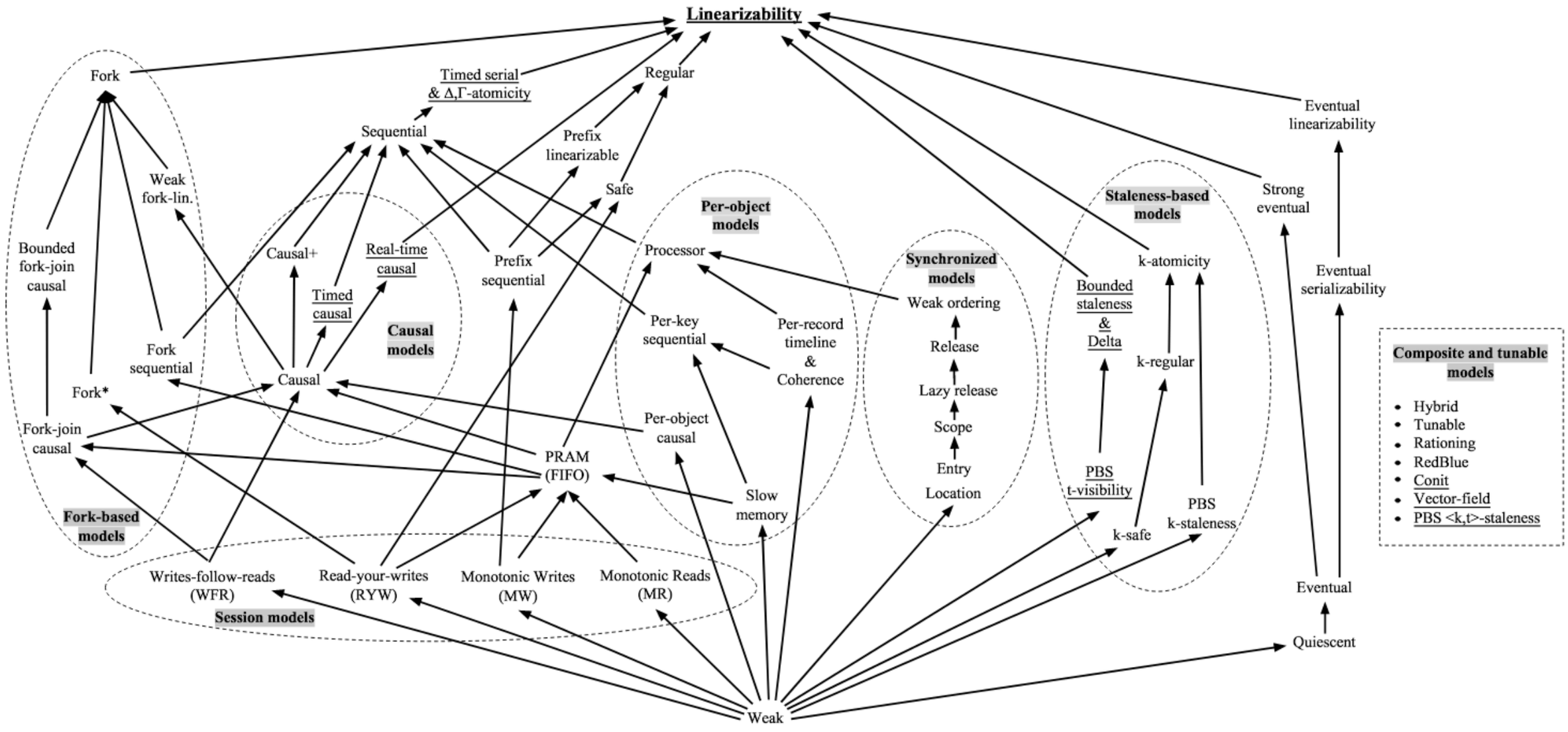
pick two?



- the original paper would have you believe so…
- but in practice, partitions occur (so you can’t really pick c+a)
- and moreover, it’s not actually a binary decision, anyway
- see Wednesday’s reading

weaker consistency conditions

"if some responses are wrong, *in what way* are they wrong?"



[Paolo Viotti and Marko Vukolić, Consistency in Non-Transactional Distributed Storage Systems]

theoretical context

CAP is just one instance of the impossibility of having both

- *safety* (“nothing bad happens”)
- *liveness* (“something good eventually happens”)

in an *unreliable distributed system*

— Gilbert and Lynch’s Perspectives on the CAP Theorem (2012)

theoretical context

CAP is just one instance of the impossibility of having both

- *safety* (“nothing bad happens”)
  i.e., consistency
- *liveness* (“something good eventually happens”)
  i.e., availability

in an *unreliable distributed system*
i.e., one that experiences partitions

— Gilbert and Lynch’s Perspectives on the CAP Theorem (2012)