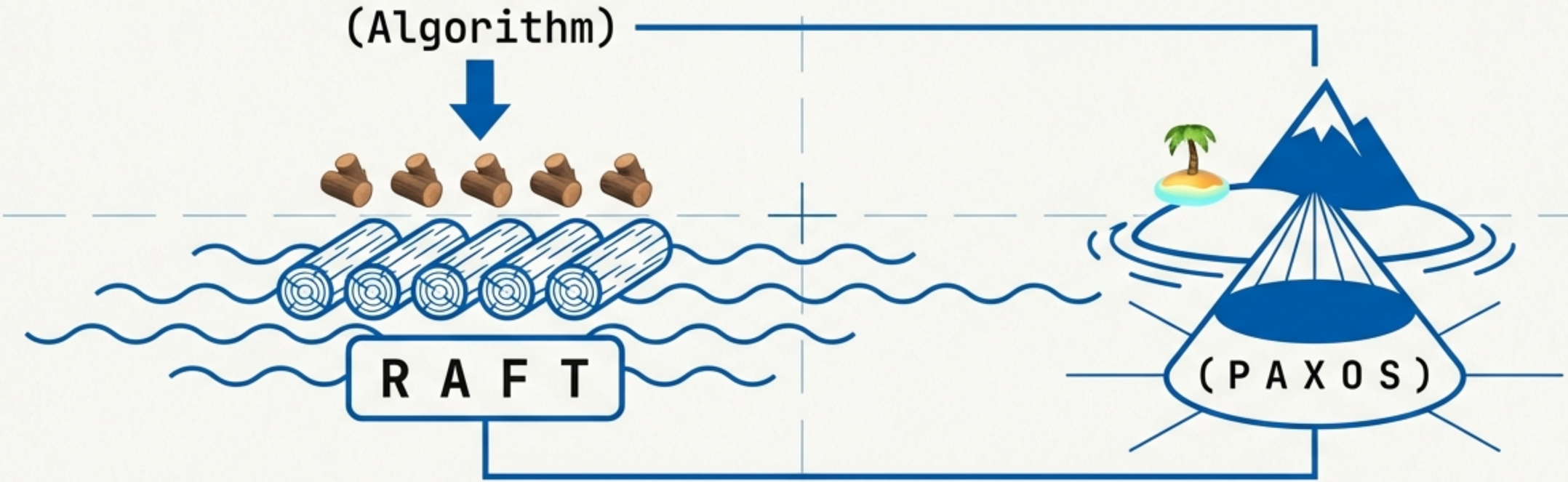
DESIGNING FOR UNDERSTANDABILITY

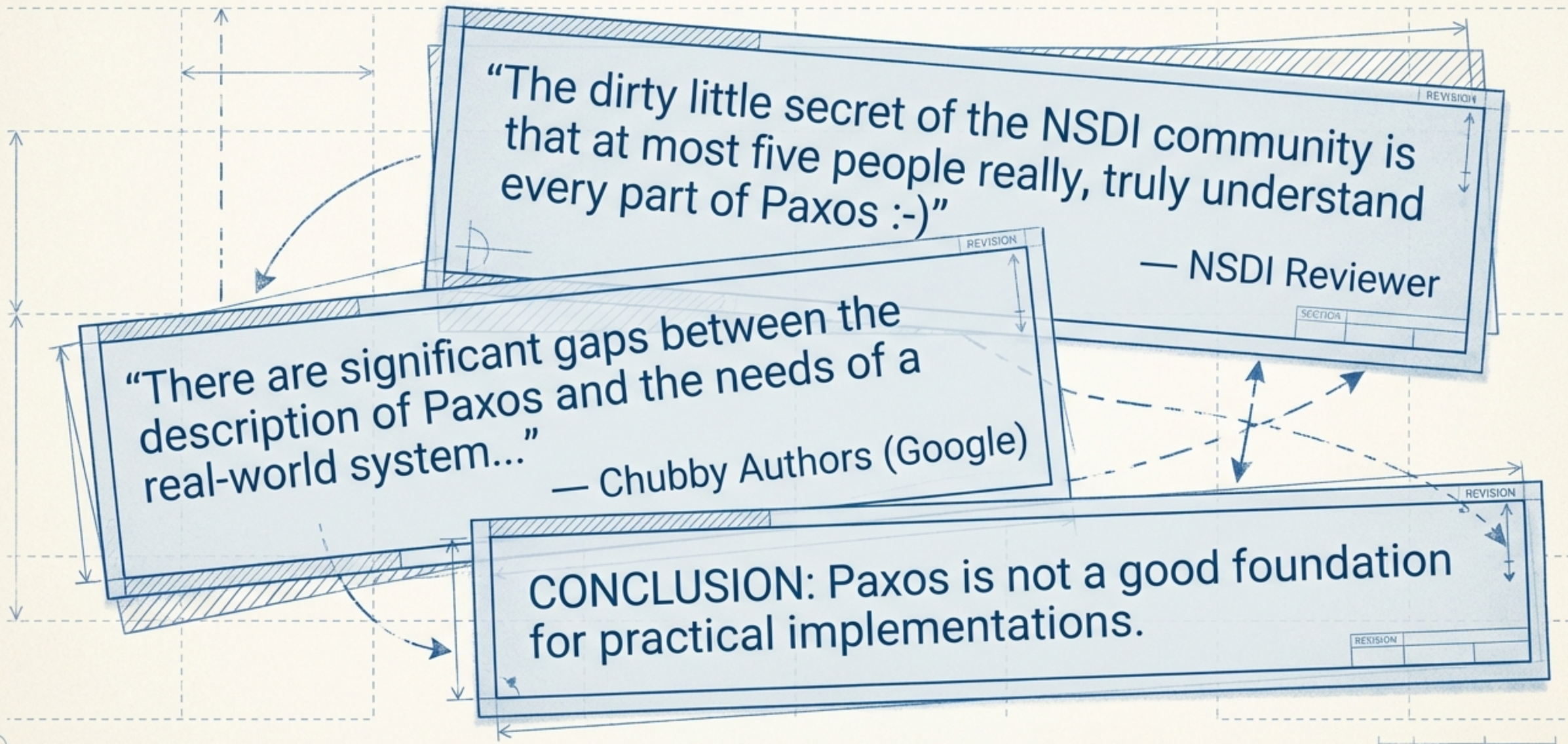
The Raft Consensus Algorithm



Diego Ongaro & John Ousterhout
Stanford University

R.A.F.T. → **R**eplicated **A**nd **F**ault **T**olerant

THE PROBLEM: PAXOS IS IMPENETRABLE.

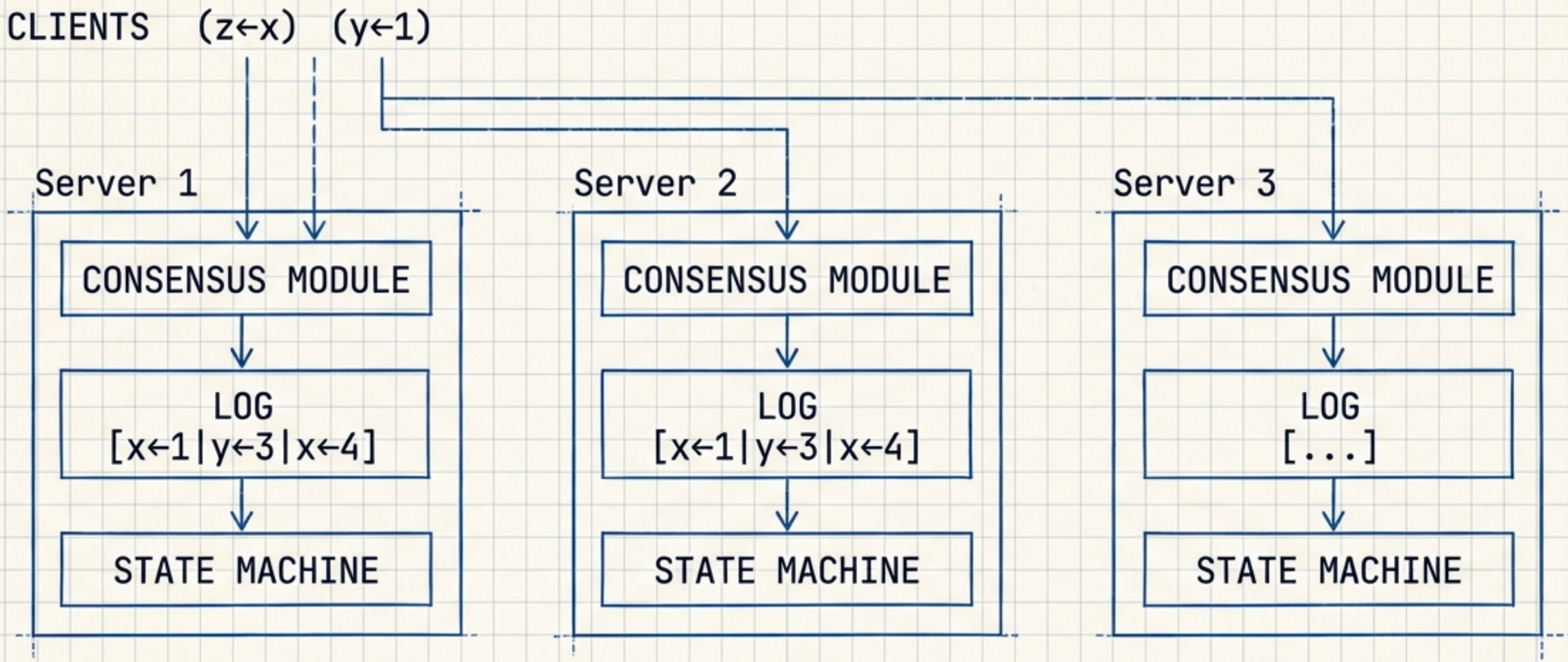


THESIS: ALGORITHMS SHOULD BE DESIGNED FOR...

- ( ) Correctness
- ( ) Efficiency
- ( ) Conciseness
- (•) **UNDERSTANDABILITY**

The Raft Challenge: Is there a different consensus algorithm that is fundamentally **easier to understand**?

CONTEXT: THE REPLICATED STATE MACHINE (RSM)



1. Consensus Module replicates Log.
2. State Machine executes commands in log order.

THE STRATEGY: DECOMPOSITION

To achieve understandability, Raft separates the problem into:

1. **LEADER ELECTION**
   Select one server to act as leader. Detect crashes.

---

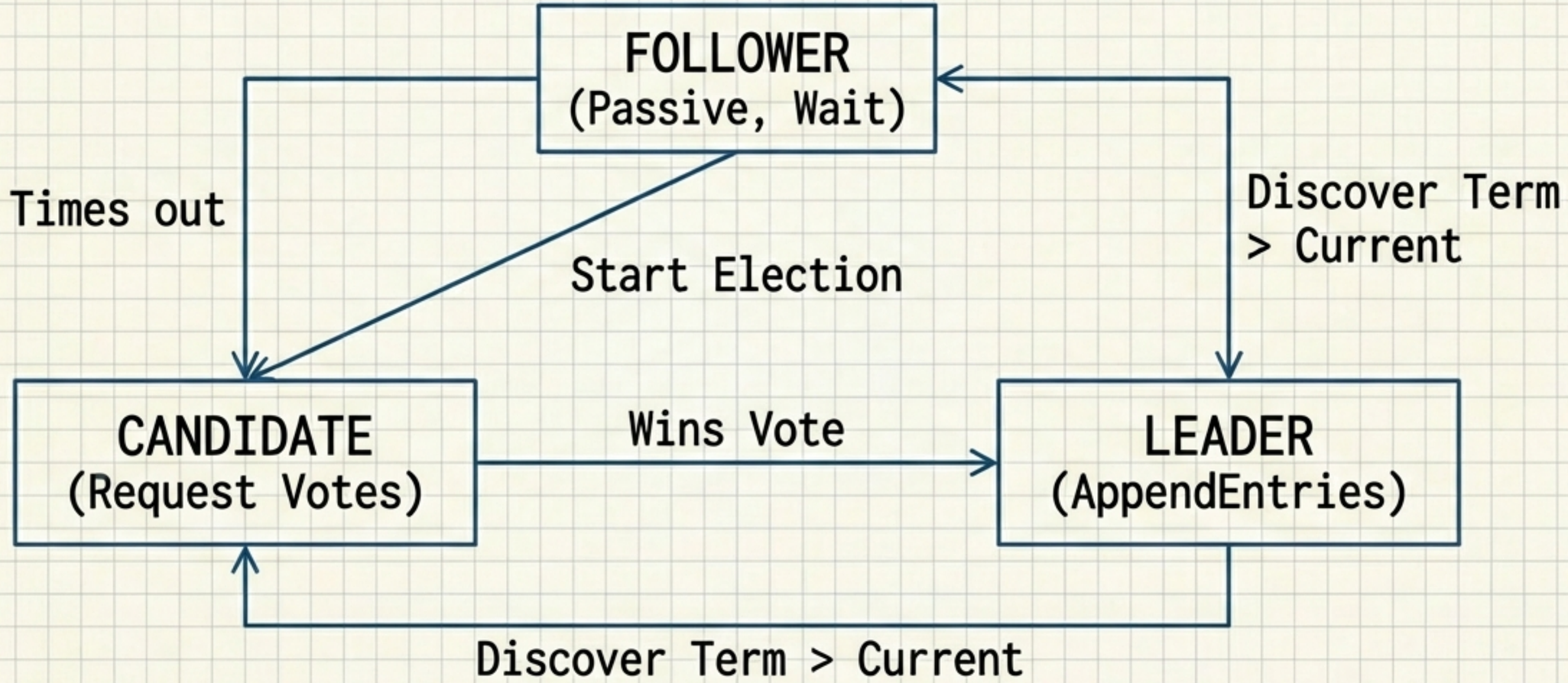
2. **LOG REPLICATION**
   Leader accepts commands, appends to log, replicates to others.

---

3. **SAFETY**
   Keep logs consistent. Only up-to-date servers can be leader.

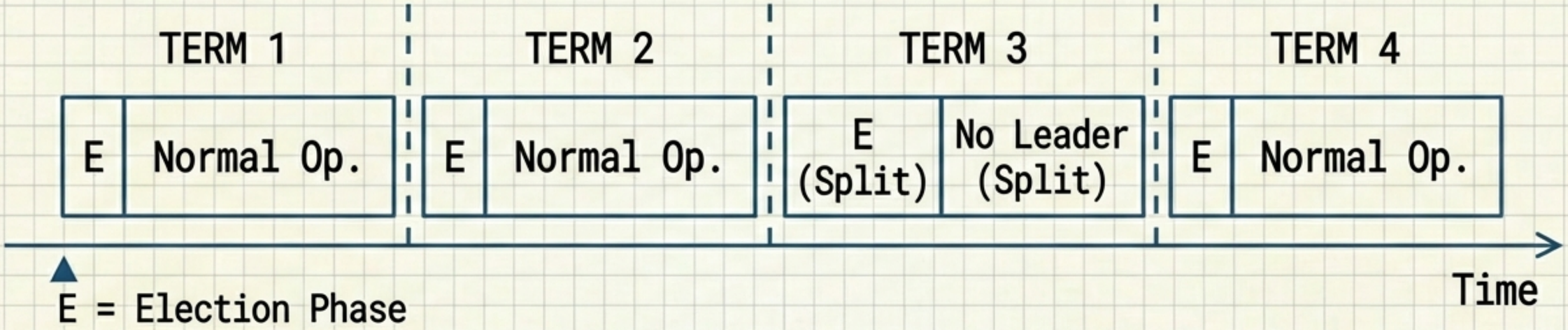
DESIGN GOAL: Minimize state space. Eliminate special cases.

MECHANISM 1: SERVER STATES



MECHANISM 2: TERMS AS LOGICAL TIME

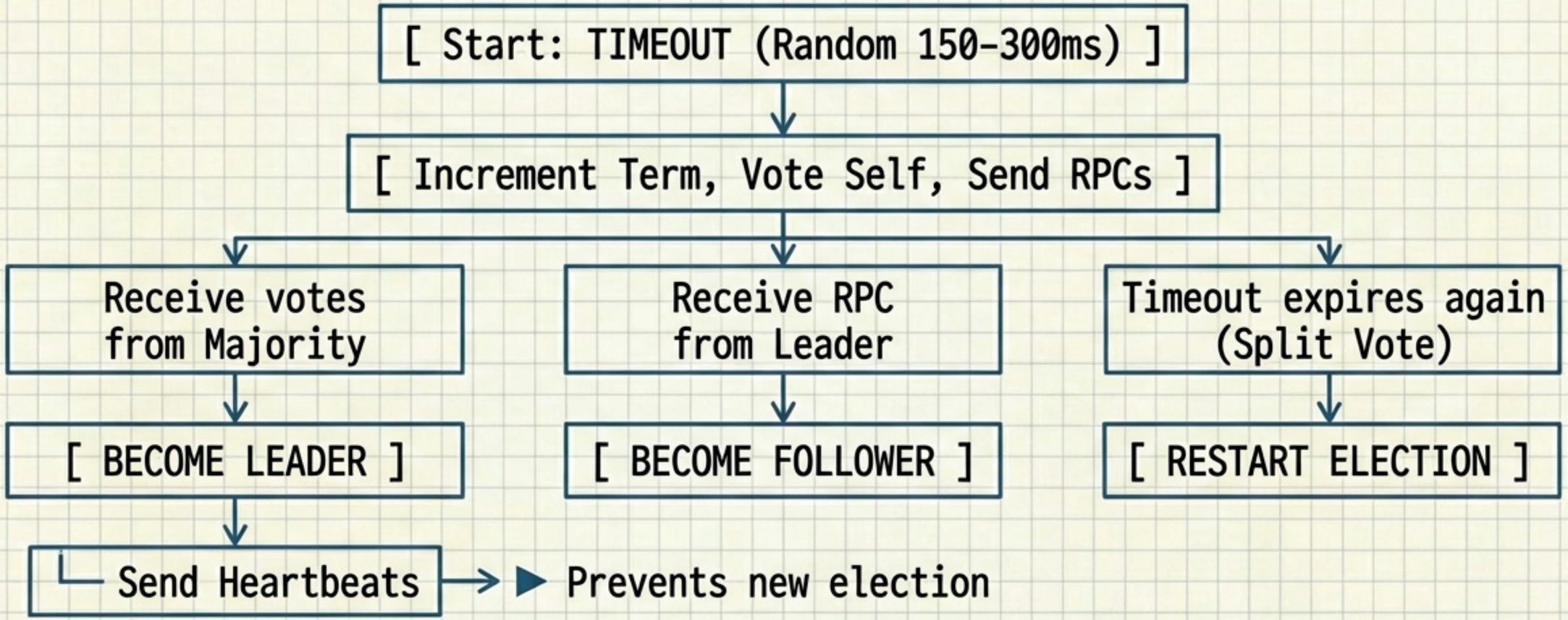
Time is divided into numbered Terms of arbitrary length.



KEY RULES:

- Terms act as a logical clock to detect obsolete information.
- Terms are exchanged in every RPC.
- **If (RPC Term > Current Term):** Update term, downgrade to Follower.

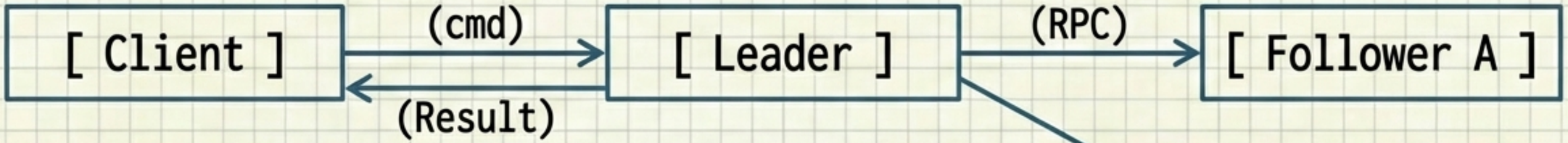
MECHANISM 3: LEADER ELECTION PROCESS



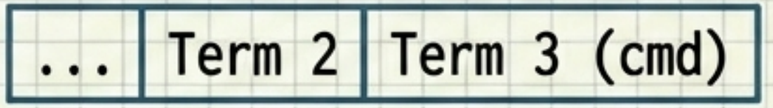
"Randomized timeouts ensure collisions are rare and resolved quickly."

NORMAL OPERATION (LOG REPLICATION)

1. **CLIENT** sends command to Leader.



2. **LEADER** appends command to local Log.



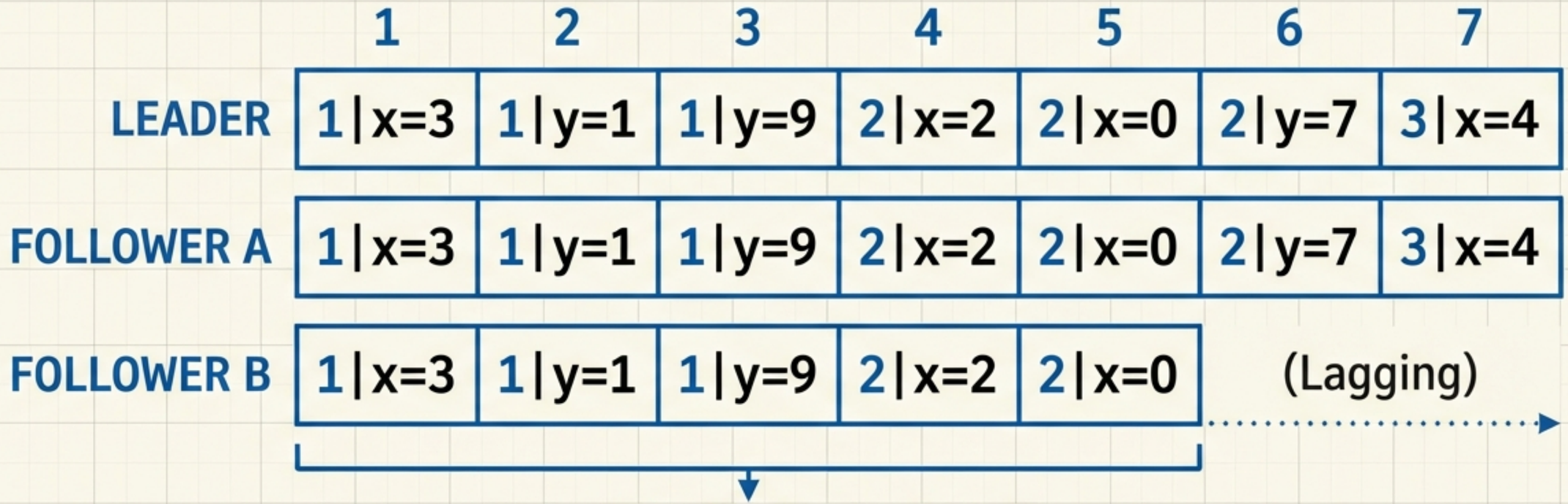
3. **LEADER** sends AppendEntries RPCs to Followers.

4. **ENTRY COMMITTED** (Once replicated on Majority).

5. **LEADER** replies to Client & notifies Followers.

DEEP DIVE: THE LOG STRUCTURE

Log Entry = [Term Number | Command]



COMMITTED ENTRIES

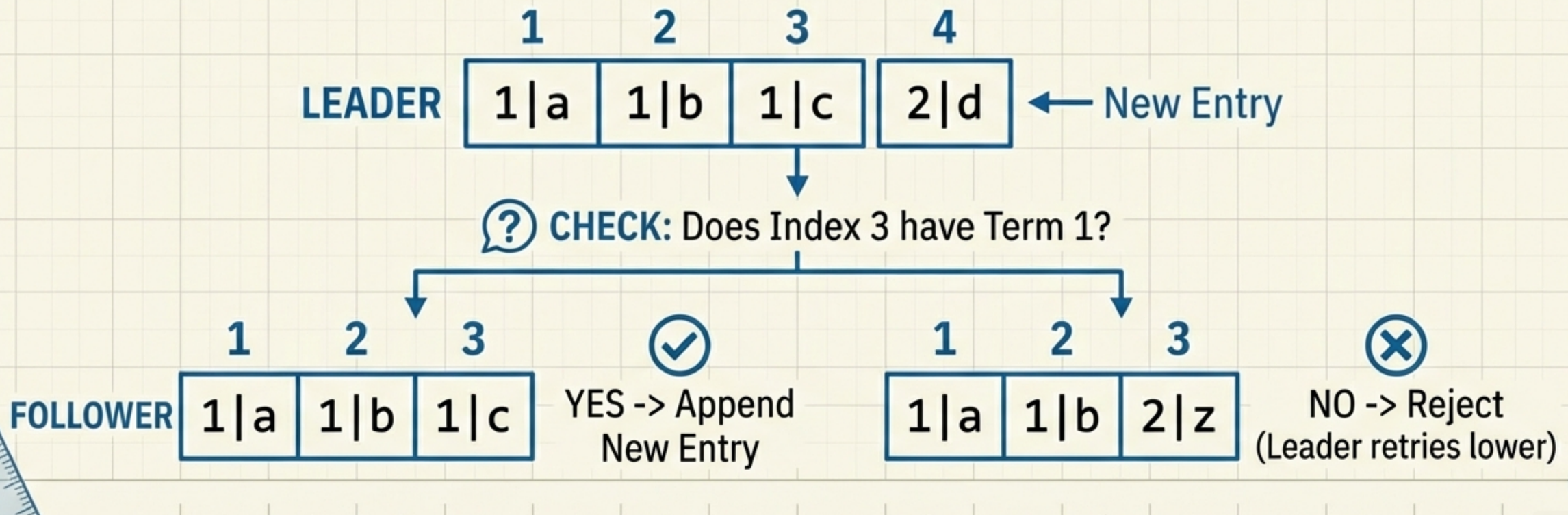
(Stored on majority -> Safe to execute)

DEEP DIVE: LOG CONSISTENCY (INDUCTION STEP)

THE LOG MATCHING PROPERTY:

If two logs agree at Index N, they agree on all 1..N.

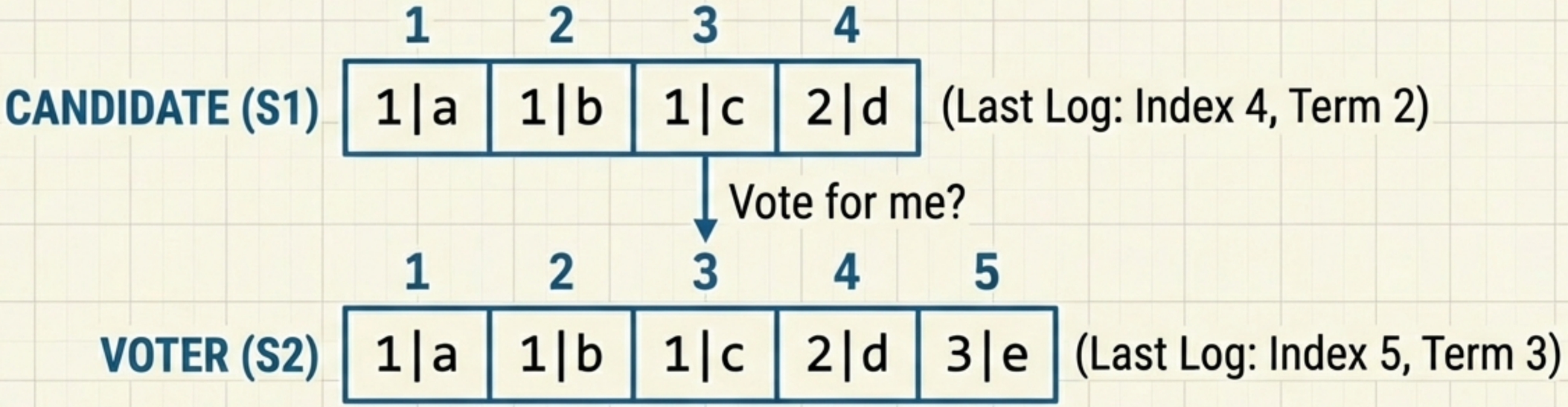
LEADER SENDING APPEND_ENTRIES(PrevIndex=3, PrevTerm=1):



SAFETY CONSTRAINT: LEADER COMPLETENESS

Rule: A candidate must contain all committed entries to get elected.

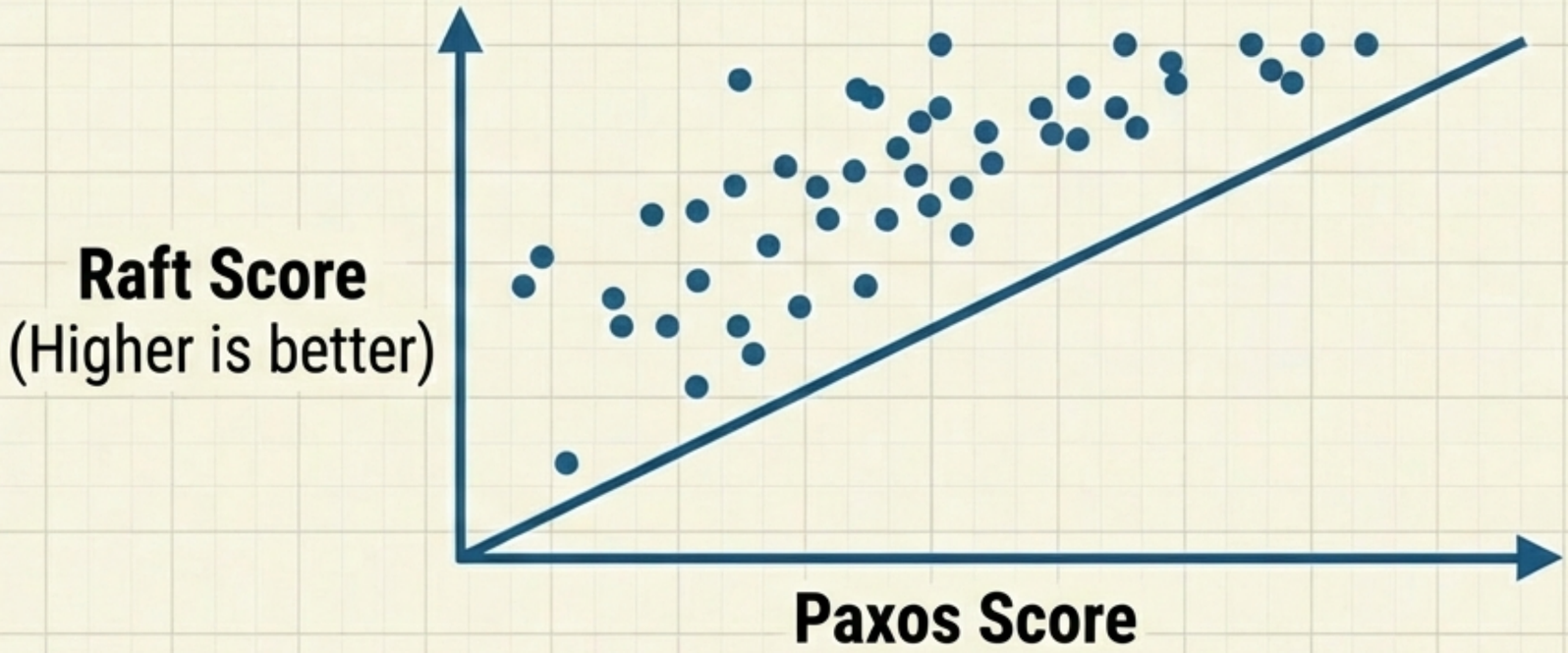
SCENARIO: Candidate S1 requests vote from Voter S2.



RESULT: ⊗ VOTE DENIED

My log is more up-to-date than yours.

EVALUATION: USER STUDY RESULTS



- 43 Grad Students (Berkeley/Stanford)
- Points above diagonal = Student scored higher on Raft

CONCLUSION: Raft is empirically easier to understand.

IMPACT: FROM REJECTION TO STANDARD

THE STRUGGLE (Academic)

- Difficult to publish.
- Rejected from 3 major conferences.
- Reviewers: “Understandability is too subjective.”

THE SUCCESS (Industry)

- Published USENIX ATC 2014.
- 83+ Open Source Implementations.
- Deployed in:
  - Kubernetes
  - Etcd
  - CockroachDB
  - Docker Swarm

SUMMARY

1. LEADER ELECTION

Randomized timeouts solve collisions.

2. LOG REPLICATION

Strong leader pushes logs to followers.

3. SAFETY

Leader Completeness Property guarantees no data loss.

“Understandability enables robust systems. Complexity is not a requirement for correctness.”