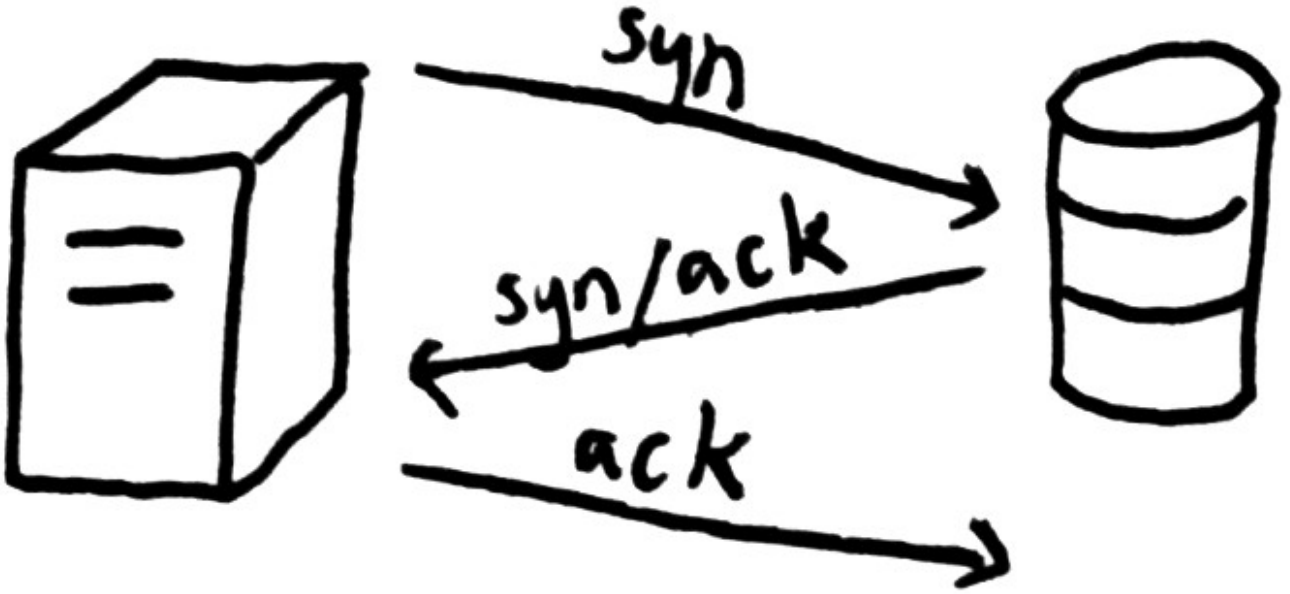
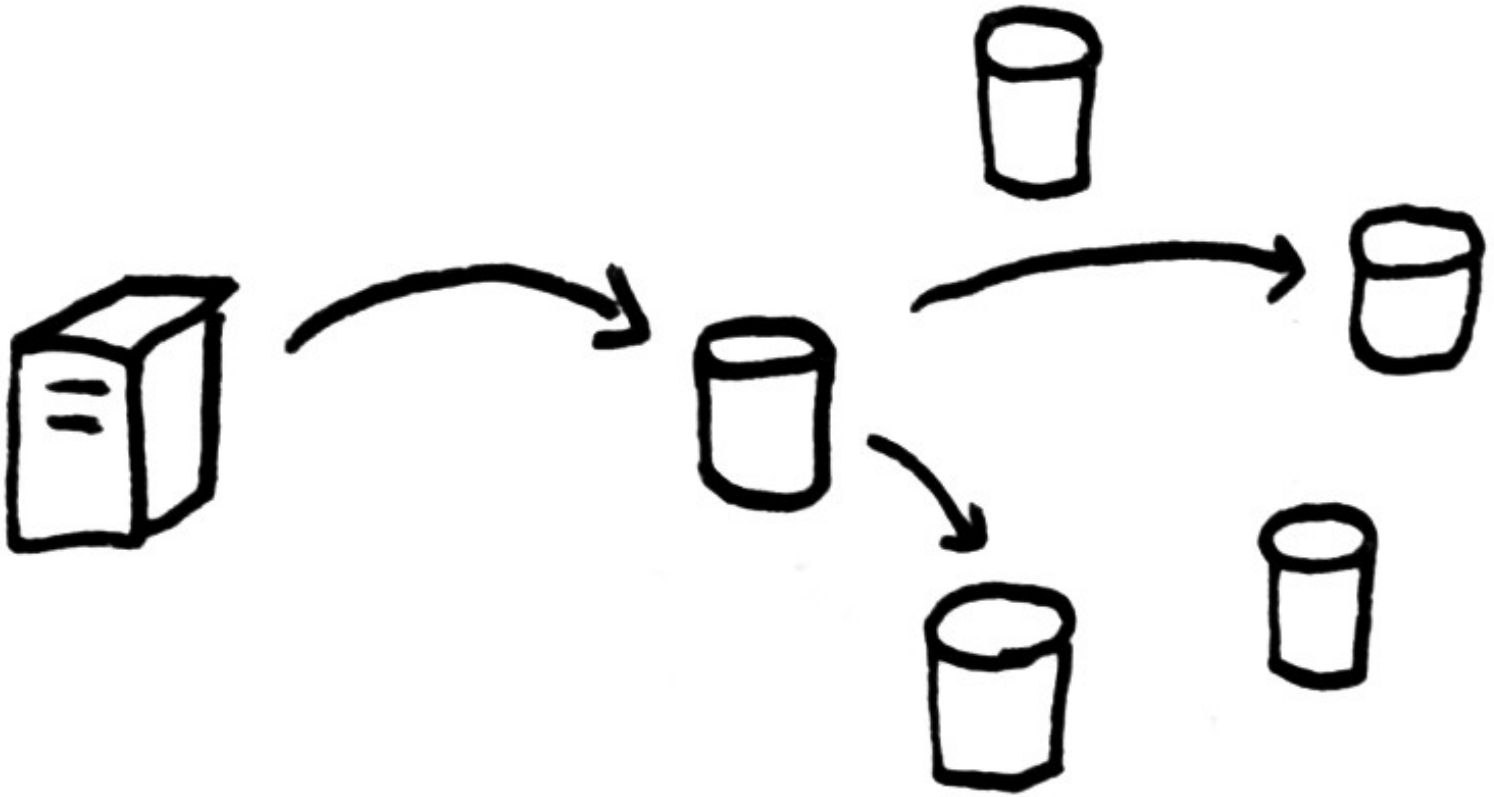
HEY,

I just met you

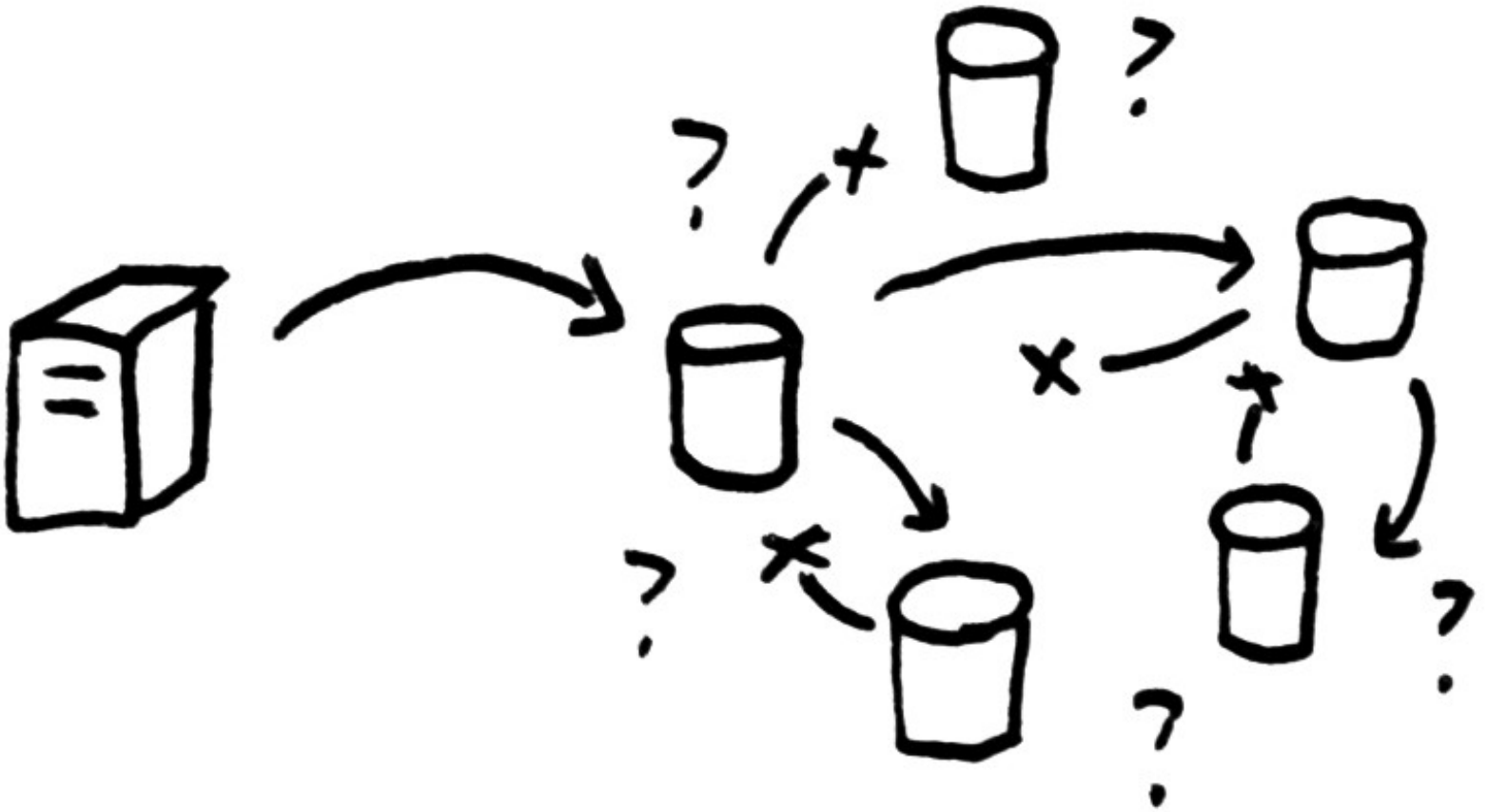


Our network's flaky

But here's my data



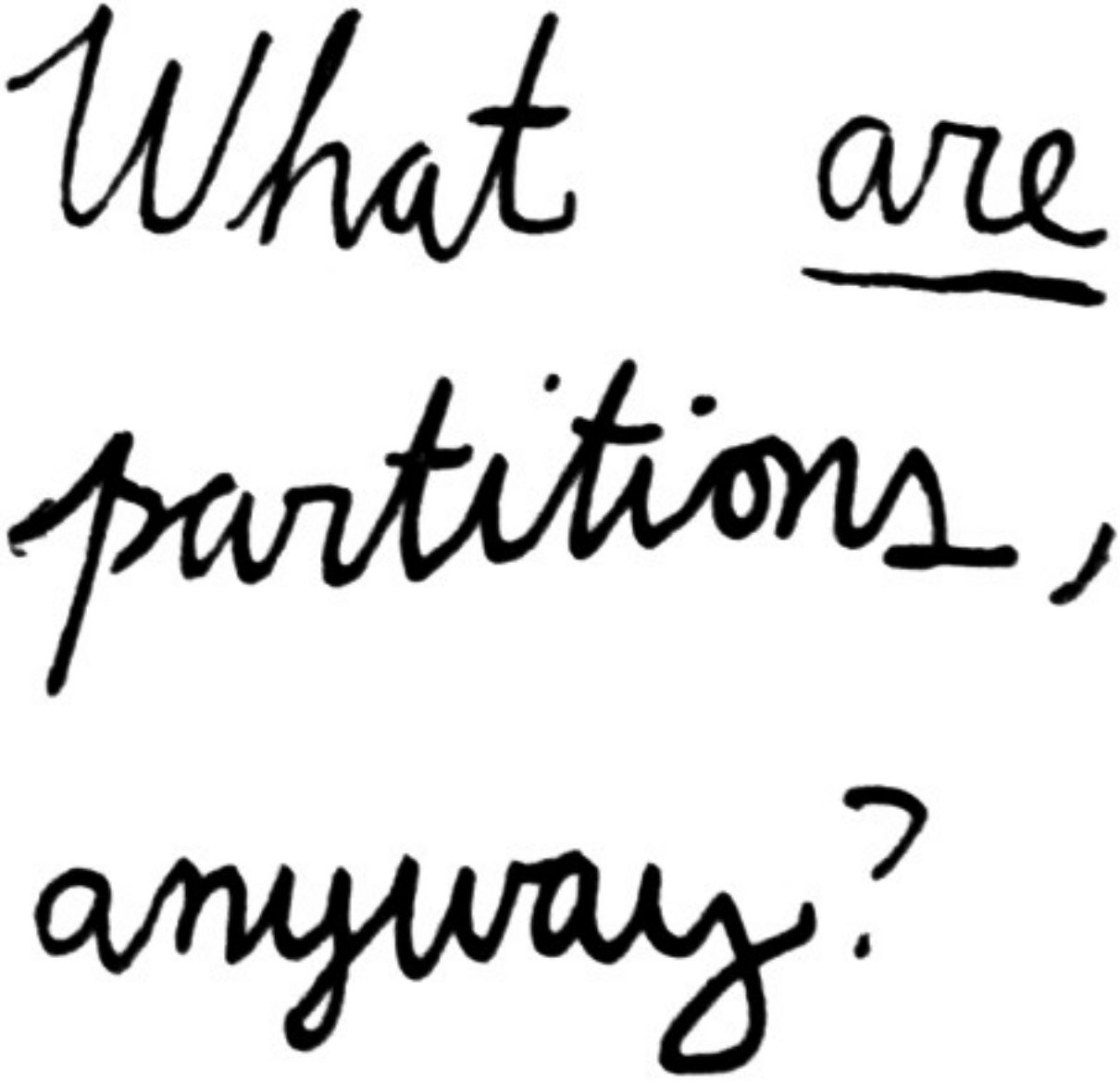
So store it maybe?



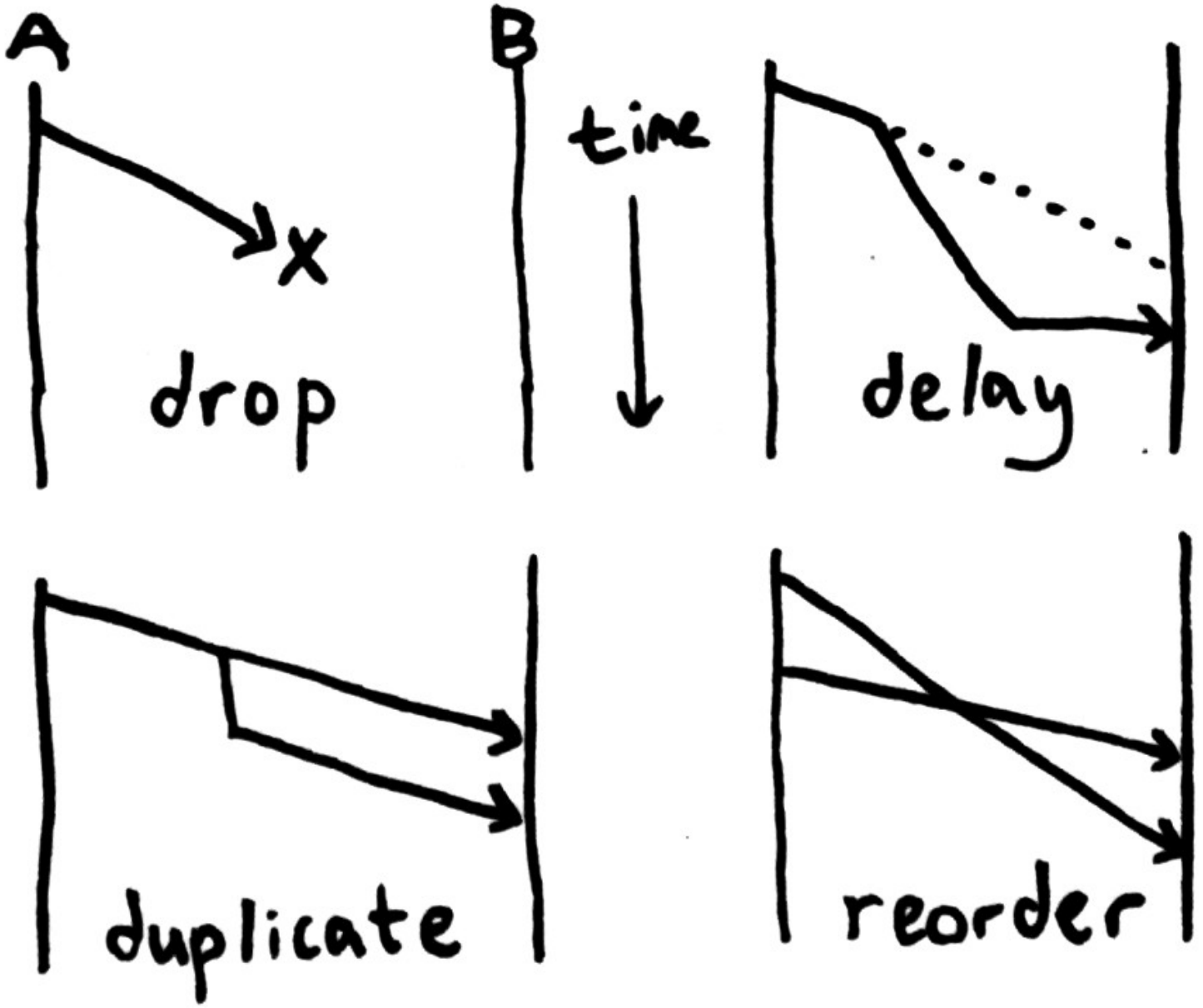
@aphyr

(Kyle Kingsbury)


PARTITIONS
★for★
EVERYONE!


What are
partitions,
anyway?

Formally:

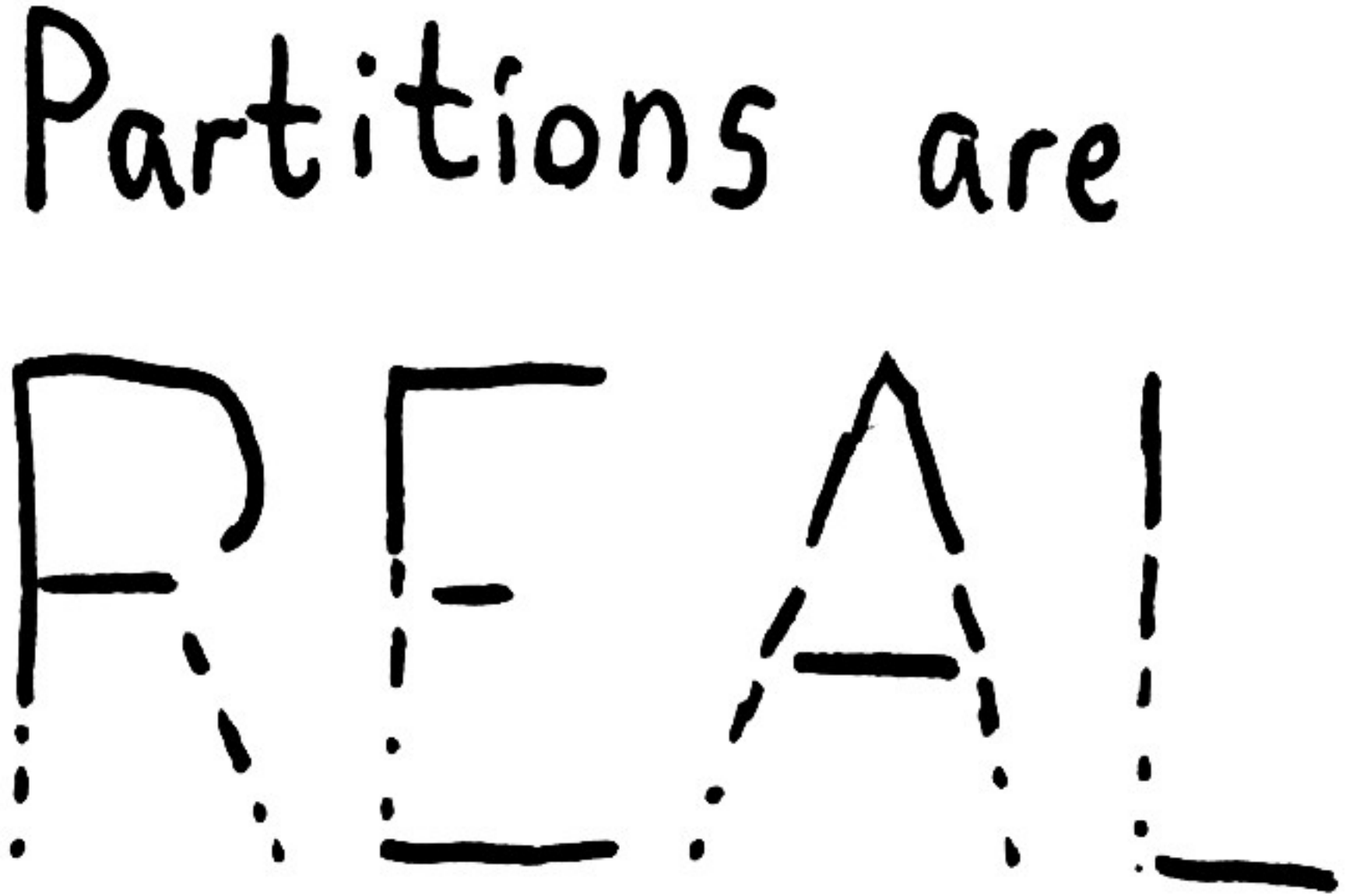


IP networks do all four.

TCP means no dupes, reorders

- Unless you retry!

Delays are indistinguishible from drops, after timeout


Partitions are
REAL

Microsoft: 40.8 link failures/day

- 5 minutes — one week
- Redundancy improves packet loss by only 43%

Google: in 1 cluster...

- 5 racks see 50% pkt loss
- 8 net maint/yr, 30 min loss in 4
- 3 router failures/yr

Yahoo Sherpa

Amazon Dynamo

Google Chubby

Causes

- GC pauses
- Network maint
- Segfaults & crashes
- Faulty NICs
- Bridge loops

... continved

- Spanning tree
- Hosted networks
- The cloud
- WAN links & Backhoes

Cloudflare

Pagerduty

Fog Creek

Netflix

Github

City Cloud

Searchbox.io

AWS

Freistel IT

Twilio

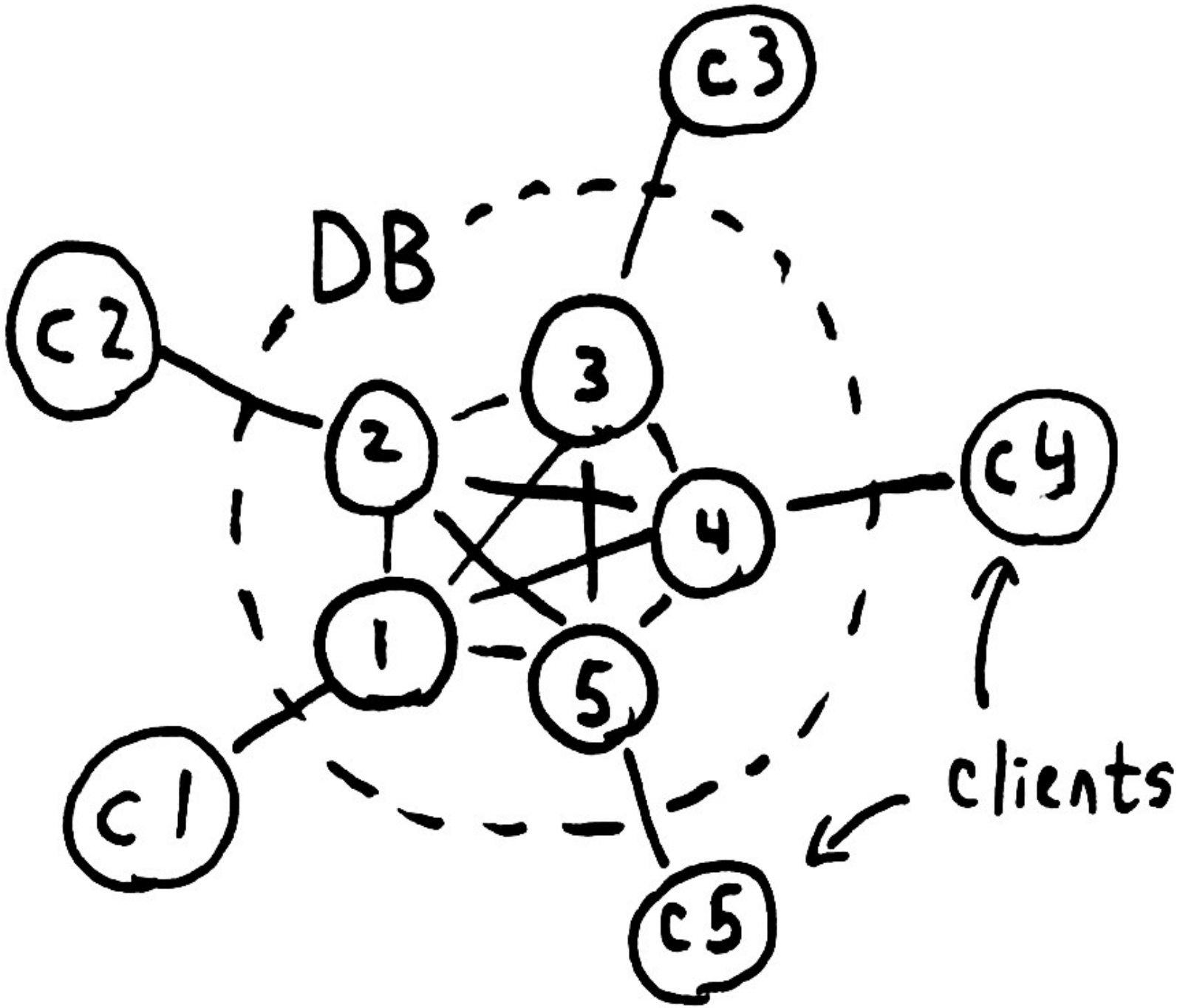
PLANET

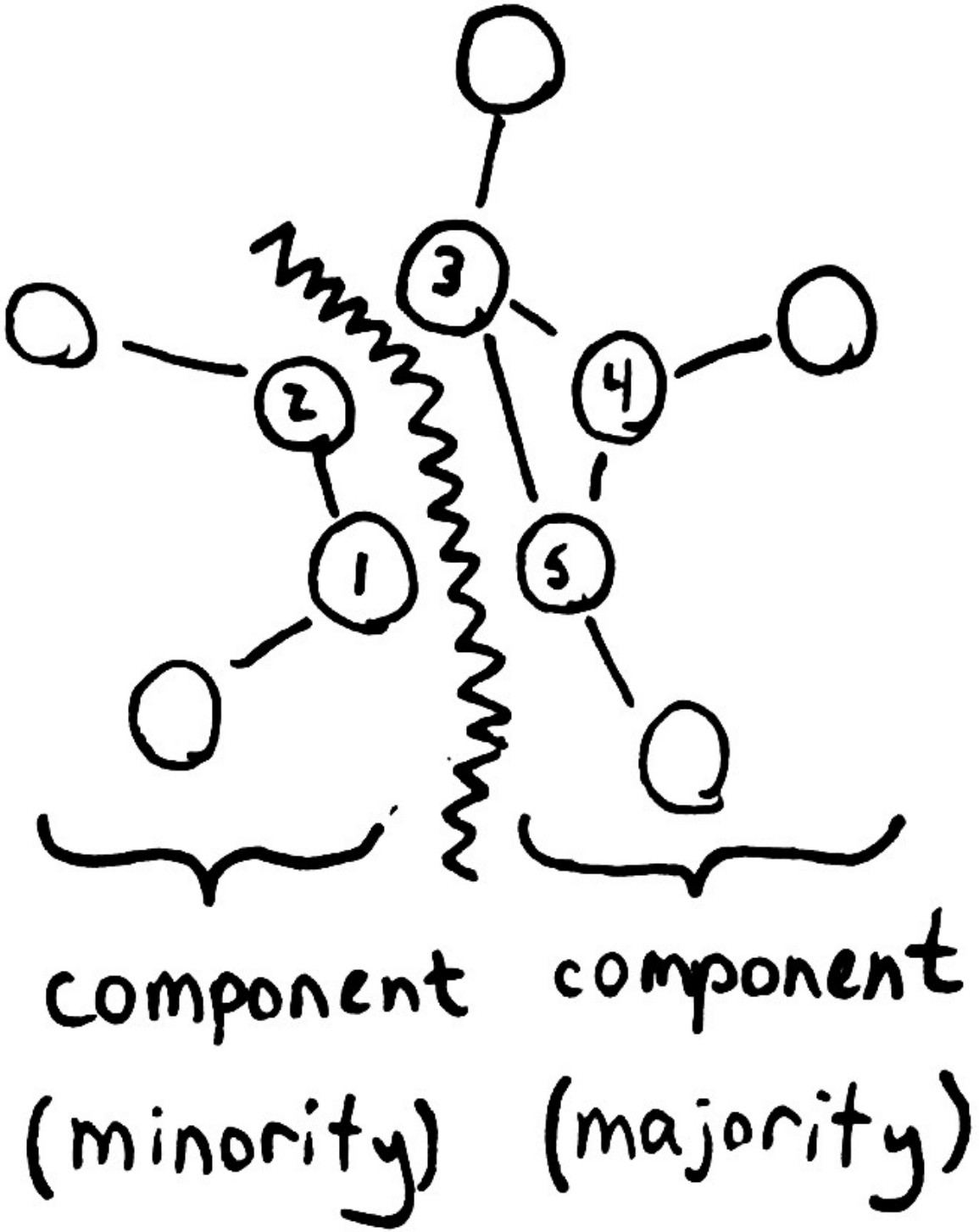
TERAH

(you are not here)

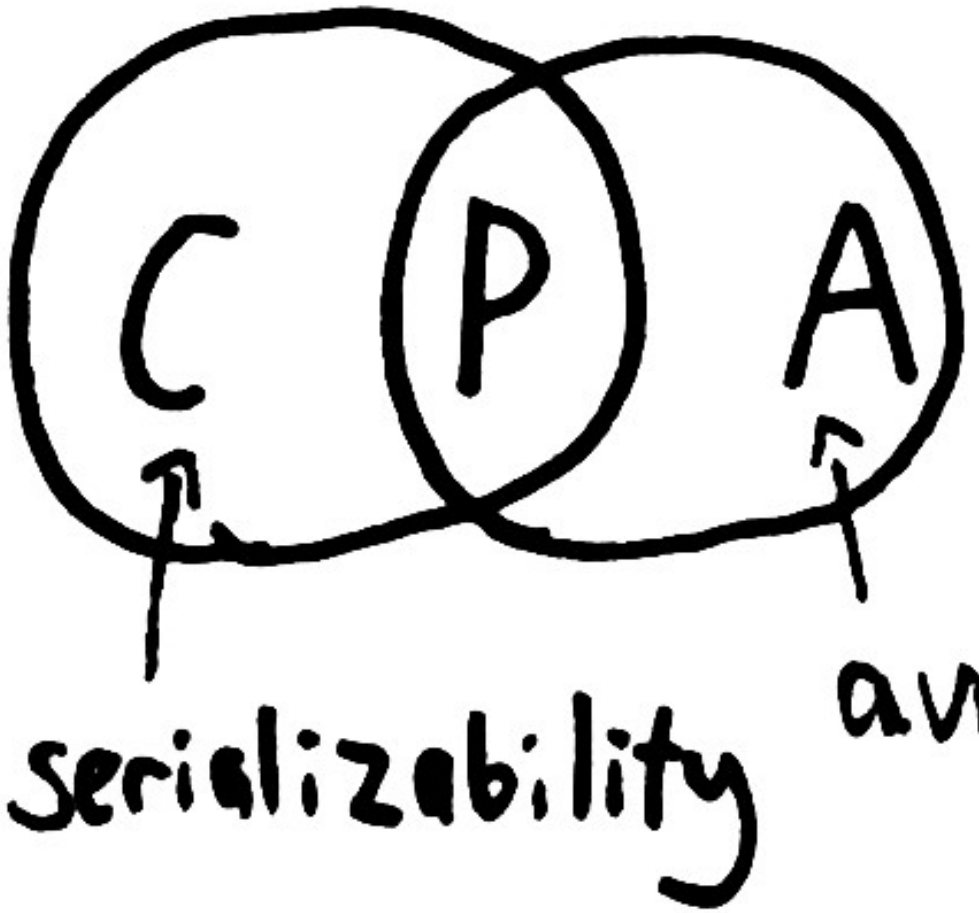
Distributed Systems tend to fail part... rtially

Jepsen is an exploration of those failure modes


c3
DB
c2
3
2
c4
4
1
5
clients
c1
c5


3
2
4
1
5
component
(minority)
component
(majority)

In broad terms



Both AP and CP systems can remain available in this case

needs a majority or primary node - forces re-election

doesn't care about partitions — business as normal

Single-server Postgres

- CP
- SPOF
- consistent
- clients may not agree

Redis (w/failover)

- Split-brain
- Drops all data on one component
- Not even close to consistent.

MongoDB

- All consistency levels led to lost data
- Weak defaults
- Majority was a bug, fixed

Rick

- LWW means dropping writes
- Even with PR/PW = ALL
- Even with lock service
- Use CRDTs

That was Then,

This is 

MEOW

That was Then,

This is 

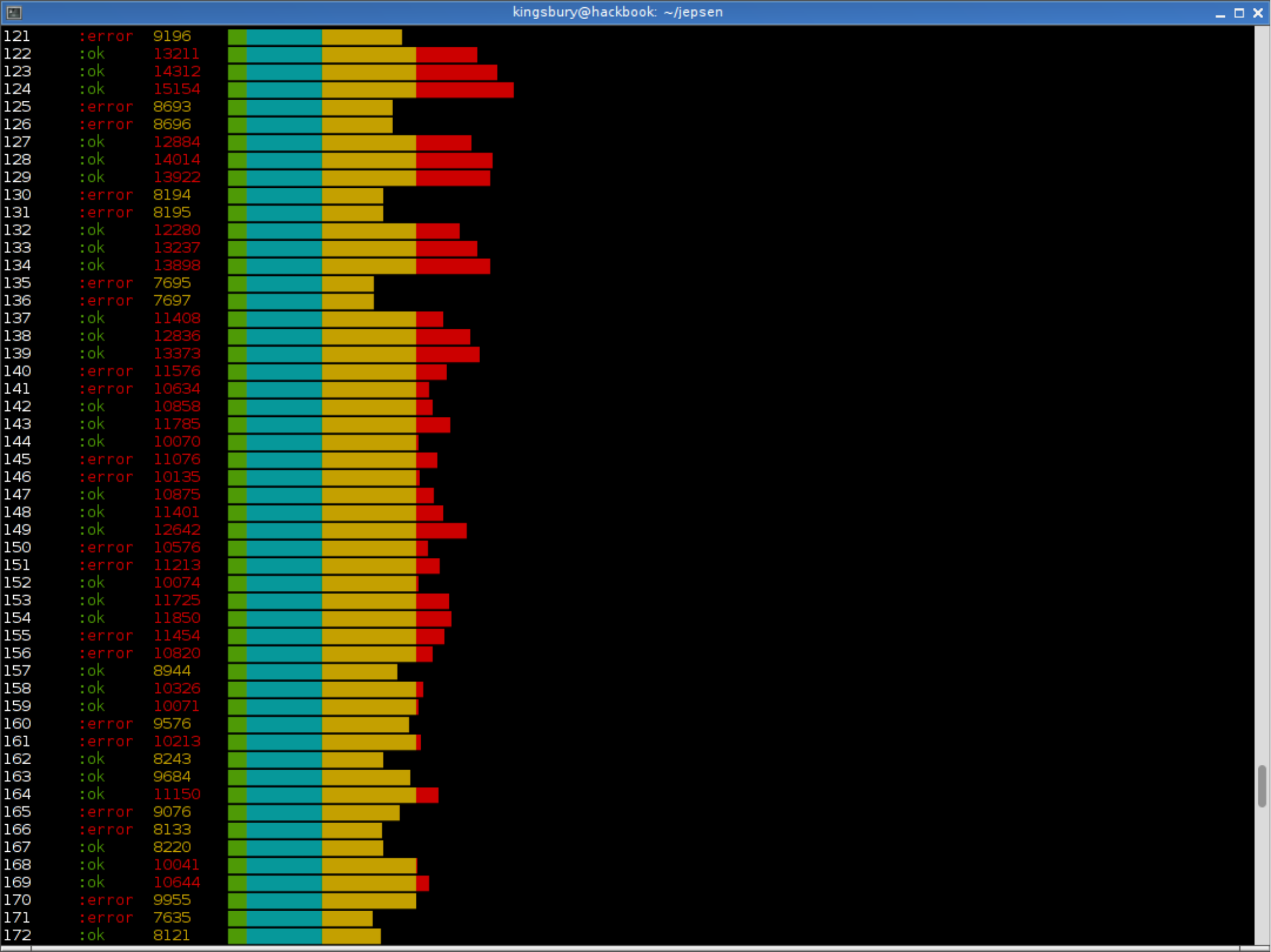
now


Zookeeper
NuoDB
Kafka
Cassandra

- Paxos-alike protocol
- Linearizable writes
- Majority w/ leaders

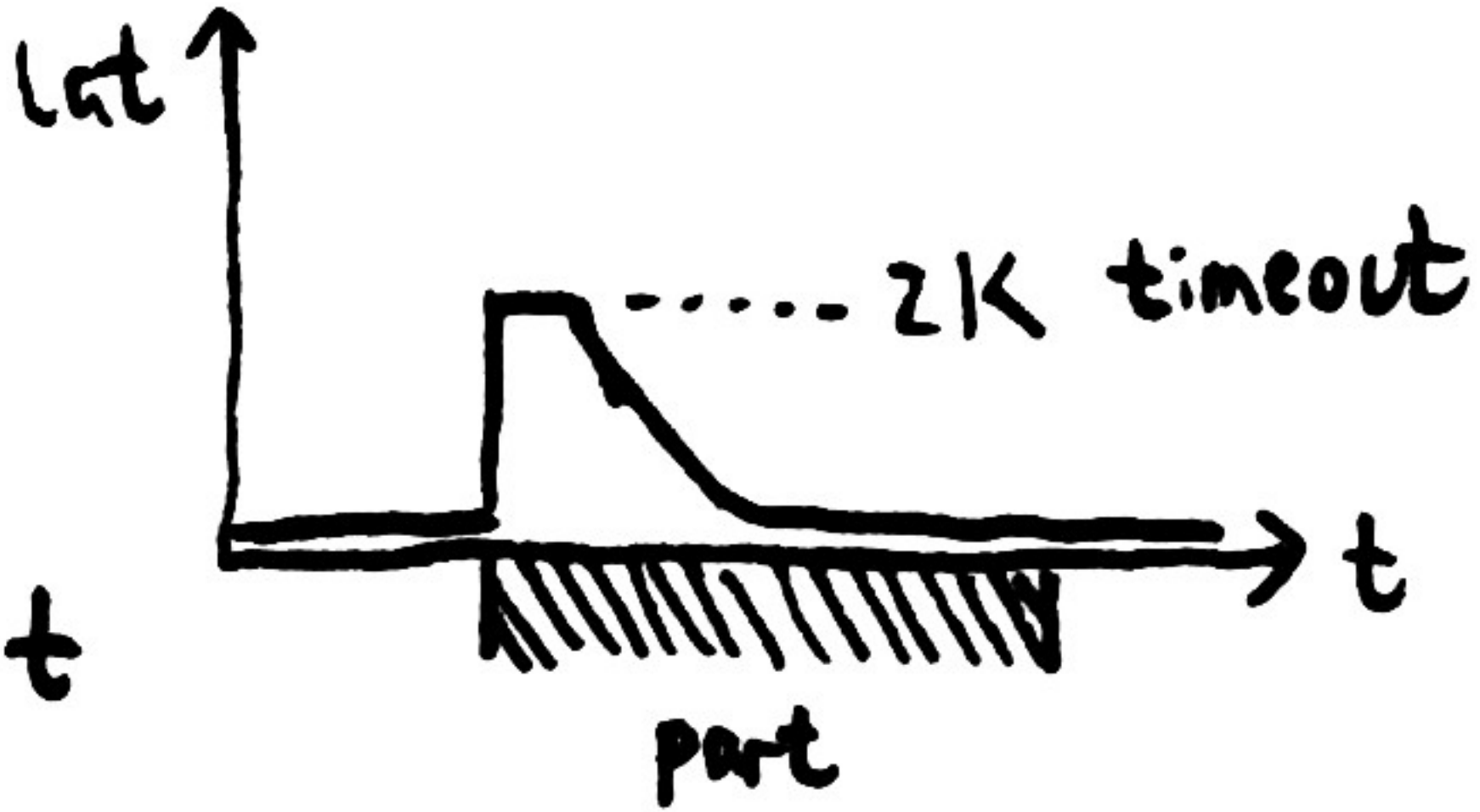
- Detects partition quickly
- Writes immediately fail
- Majority component continues after election
- Only partly available

Boooring.


kingsbury@hackbook: ~/jepsen
121 :error 9196
122 :ok 13211
123 :ok 14312
124 :ok 15154
125 :error 8693
126 :error 8696
127 :ok 12884
128 :ok 14014
129 :ok 13922
130 :error 8194
131 :error 8195
132 :ok 12280
133 :ok 13237
134 :ok 13898
135 :error 7695
136 :error 7697
137 :ok 11408
138 :ok 12836
139 :ok 13373
140 :error 11576
141 :error 10634
142 :ok 10858
143 :ok 11785
144 :ok 10070
145 :error 11076
146 :error 10135
147 :ok 10875
148 :ok 11401
149 :ok 12642
150 :error 10576
151 :error 11213
152 :ok 10074
153 :ok 11725
154 :ok 11850
155 :error 11454
156 :error 10820
157 :ok 8944
158 :ok 10326
159 :ok 10071
160 :error 9576
161 :error 10213
162 :ok 8243
163 :ok 9684
164 :ok 11150
165 :error 9076
166 :error 8133
167 :ok 8220
168 :ok 10041
169 :ok 10644
170 :error 9955
171 :error 7635
172 :ok 8121

```
49465 [pool-11-thread-20-SendThread(n1:2181)] INFO org.apache.zookeeper.ClientCnxn - Socket connection established to n1/10.10.3.242:2
181, initiating session
49469 [pool-11-thread-20-SendThread(n1:2181)] INFO org.apache.zookeeper.ClientCnxn - Unable to read additional data from server sessio
nid 0x0, likely server has closed socket, closing socket connection and attempting reconnect
49548 [pool-13-thread-39] INFO org.apache.zookeeper.ZooKeeper - Session: 0x0 closed
49548 [pool-13-thread-36-EventThread] INFO org.apache.zookeeper.ClientCnxn - EventThread shut down
49548 [pool-13-thread-39] INFO org.apache.zookeeper.ZooKeeper - Initiating client connection, connectString=n2:2181 sessionTimeout=200
0 watcher=org.apache.curator.ConnectionState@110d66fa
49549 [pool-13-thread-39-SendThread(n2:2181)] INFO org.apache.zookeeper.ClientCnxn - Opening socket connection to server n2/10.10.3.52
:2181. Will not attempt to authenticate using SASL (unknown error)
49550 [pool-13-thread-39-SendThread(n2:2181)] INFO org.apache.zookeeper.ClientCnxn - Socket connection established to n2/10.10.3.52:21
81, initiating session
49550 [pool-13-thread-39-SendThread(n2:2181)] INFO org.apache.zookeeper.ClientCnxn - Unable to read additional data from server sessio
nid 0x0, likely server has closed socket, closing socket connection and attempting reconnect
300     :error  11696
301     :error  9640
302     :ok     66
303     :ok     67
304     :ok     50
305     :error  11196
306     :error  9140
307     :ok     44
308     :ok     63
309     :ok     50
310     :error  10696
311     :error  9970
312     :ok     17
313     :ok     46
314     :ok     13
315     :error  10195
50777 [pool-13-thread-39-SendThread(n2:2181)] INFO org.apache.zookeeper.ClientCnxn - Opening socket connection to server n2/10.10.3.52
:2181. Will not attempt to authenticate using SASL (unknown error)
50778 [pool-13-thread-39-SendThread(n2:2181)] INFO org.apache.zookeeper.ClientCnxn - Socket connection established to n2/10.10.3.52:21
81, initiating session
50788 [pool-13-thread-39-SendThread(n2:2181)] INFO org.apache.zookeeper.ClientCnxn - Unable to read additional data from server sessio
nid 0x0, likely server has closed socket, closing socket connection and attempting reconnect
316     :error  11514
317     :ok     50
318     :ok     50
319     :ok     77
320     :error  9586
321     :error  8969
322     :ok     45
323     :ok     75
324     :ok     51
51374 [pool-11-thread-42] WARN org.apache.curator.ConnectionState - Connection attempt unsuccessful after 2026 (greater than max timeo
ut of 2000). Resetting connection and trying again with a new connection.
51397 [pool-11-thread-20-SendThread(n1:2181)] INFO org.apache.zookeeper.ClientCnxn - Opening socket connection to server n1/10.10.3.24
2:2181. Will not attempt to authenticate using SASL (unknown error)
51397 [pool-11-thread-20-SendThread(n1:2181)] INFO org.apache.zookeeper.ClientCnxn - Socket connection established to n1/10.10.3.242:2
181, initiating session
51412 [pool-13-thread-22] WARN org.apache.curator.ConnectionState - Connection attempt unsuccessful after 2037 (greater than max timeo
```

Latency



ZooKeeper

78% acknowledged

0% false positives

0.1% false negatives

ZK strategies

- Use it.
- Use curator, etc.
- Consider linearizability carefully! (writes only)

NUODB

Brewer's CAP Conjecture is False

and a purported "proof" at

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.1495&rep=rep1&type=pdf

The CAP conjecture, I am convinced, is false and can be proven false.

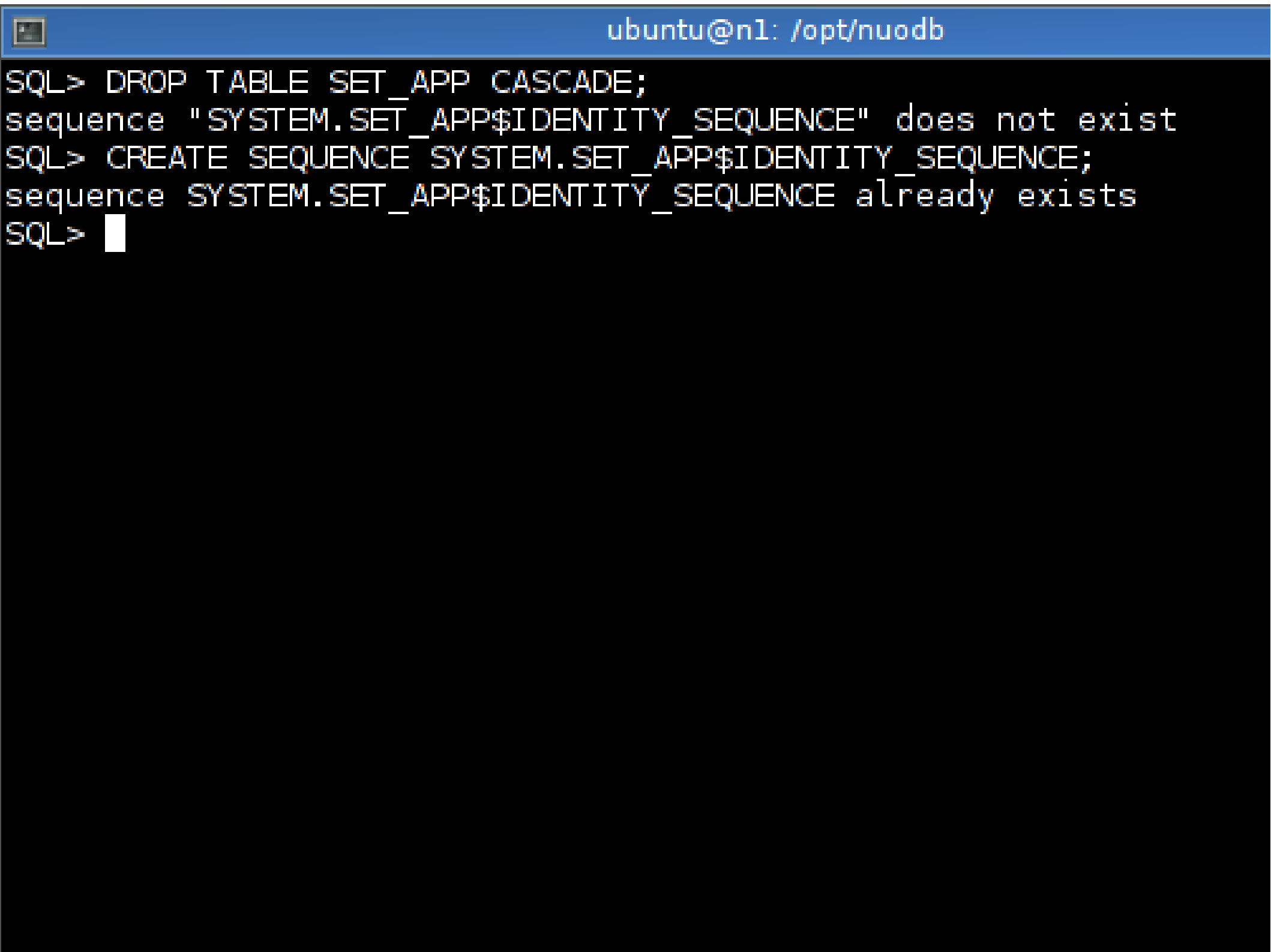
The CAP conjecture has been a theoretical millstone around the neck of all ACID systems. Good riddance.

This is the first wooden stake for the heart of the noSQL movement. There are more coming.

--
Jim Starkey
Founder, NimbusDB, Inc.
978 526-1376

If the CAP theorem means that all surviving nodes must be able to continue processing without communication after a network failure, then NUODB is not partition resistant. If partition resistance includes the possibility for a surviving subset of the chorus to sing on, then NUODB refutes the CAP theorem.

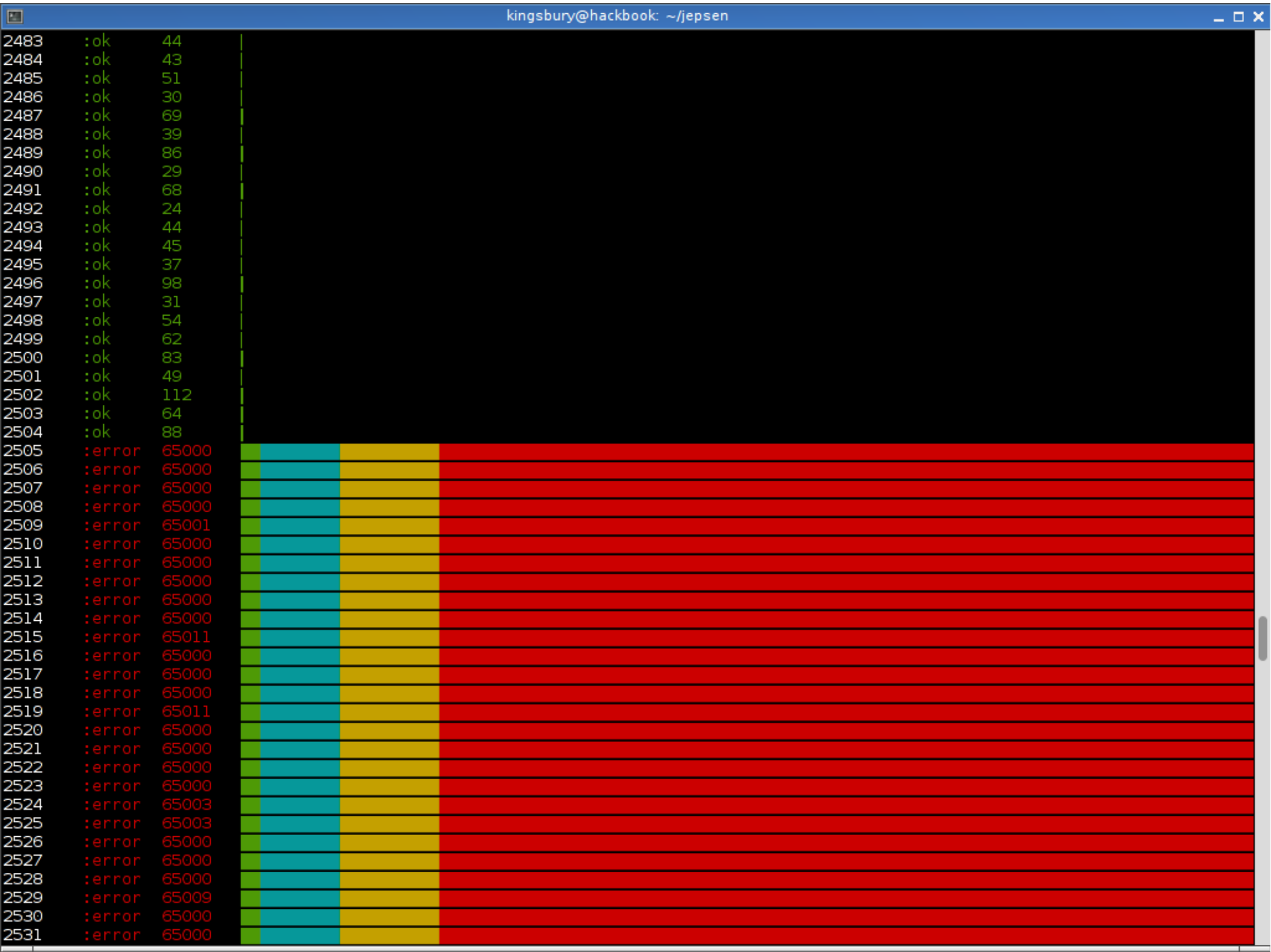
- Design is complex
- Monotonic leaders for ensembles over each row, table, db, ...
- Claims to be C, A, ∄P?

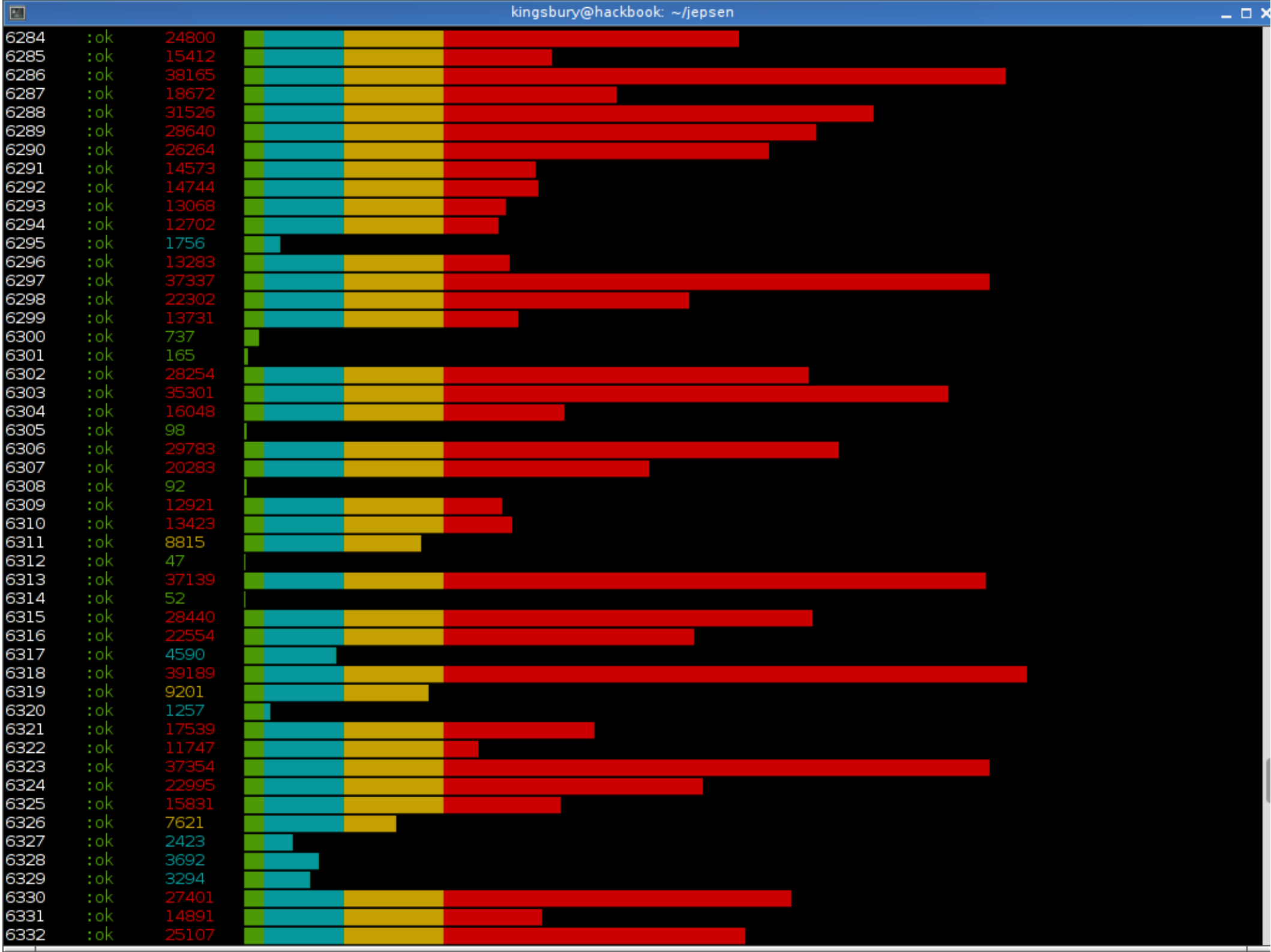

ubuntu@n1: /opt/nuodb
SQL> DROP TABLE SET_APP CASCADE;
sequence "SYSTEM.SET_APP$IDENTITY_SEQUENCE" does not exist
SQL> CREATE SEQUENCE SYSTEM.SET_APP$IDENTITY_SEQUENCE;
sequence SYSTEM.SET_APP$IDENTITY_SEQUENCE already exists
SQL>

- Race conditions in join
- Can self-join nodes
- Race conditions in DB create
- Basically impossible to start reliably
- Does not bring crashed node storage back online

"Duplicate value in unique index SEQUENCES..PRIMARY_KEY

- End of stream reached
- Broken pipe,
- Conn reset
- indefinite latency


kingsbury@hackbook: ~/jepsen
2483 :ok 44
2484 :ok 43
2485 :ok 51
2486 :ok 30
2487 :ok 69
2488 :ok 39
2489 :ok 86
2490 :ok 29
2491 :ok 68
2492 :ok 24
2493 :ok 44
2494 :ok 45
2495 :ok 37
2496 :ok 98
2497 :ok 31
2498 :ok 54
2499 :ok 62
2500 :ok 83
2501 :ok 49
2502 :ok 112
2503 :ok 64
2504 :ok 88
2505 :error 65000
2506 :error 65000
2507 :error 65000
2508 :error 65000
2509 :error 65001
2510 :error 65000
2511 :error 65000
2512 :error 65000
2513 :error 65000
2514 :error 65000
2515 :error 65011
2516 :error 65000
2517 :error 65000
2518 :error 65000
2519 :error 65011
2520 :error 65000
2521 :error 65000
2522 :error 65000
2523 :error 65000
2524 :error 65003
2525 :error 65003
2526 :error 65000
2527 :error 65000
2528 :error 65000
2529 :error 65009
2530 :error 65000
2531 :error 65000

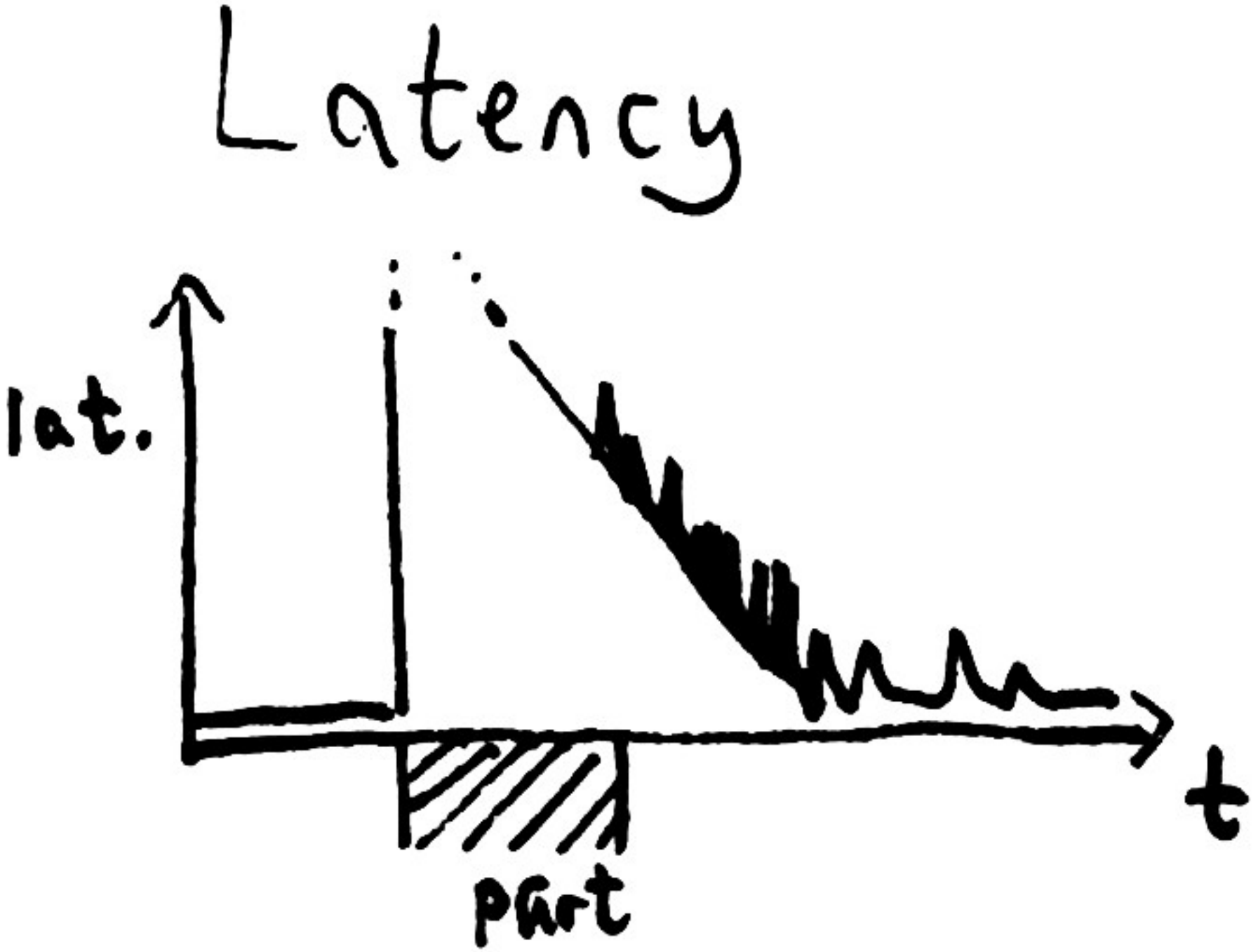

kingsbury@hackbook: ~/jepsen
6284 :ok 24800
6285 :ok 15412
6286 :ok 38165
6287 :ok 18672
6288 :ok 31526
6289 :ok 28640
6290 :ok 26264
6291 :ok 14573
6292 :ok 14744
6293 :ok 13068
6294 :ok 12702
6295 :ok 1756
6296 :ok 13283
6297 :ok 37337
6298 :ok 22302
6299 :ok 13731
6300 :ok 737
6301 :ok 165
6302 :ok 28254
6303 :ok 35301
6304 :ok 16048
6305 :ok 98
6306 :ok 29783
6307 :ok 20283
6308 :ok 92
6309 :ok 12921
6310 :ok 13423
6311 :ok 8815
6312 :ok 47
6313 :ok 37139
6314 :ok 52
6315 :ok 28440
6316 :ok 22554
6317 :ok 4590
6318 :ok 39189
6319 :ok 9201
6320 :ok 1257
6321 :ok 17539
6322 :ok 11747
6323 :ok 37354
6324 :ok 22995
6325 :ok 15831
6326 :ok 7621
6327 :ok 2423
6328 :ok 3692
6329 :ok 3294
6330 :ok 27401
6331 :ok 14891
6332 :ok 25107

NuoDB

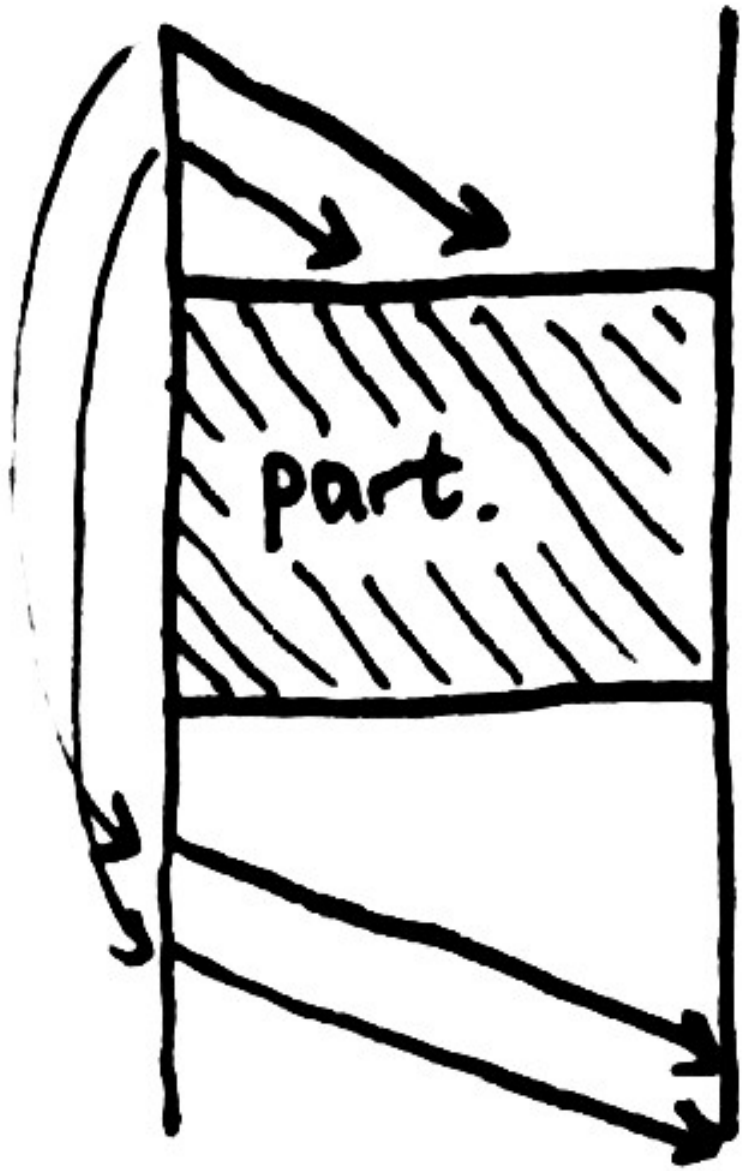
55% acknowledged

0% false positives

0.1% false negatives


Latency
lat.
t
part

"Is a database which takes five minutes to respond really available?"



This is just pushing the problem off for later.

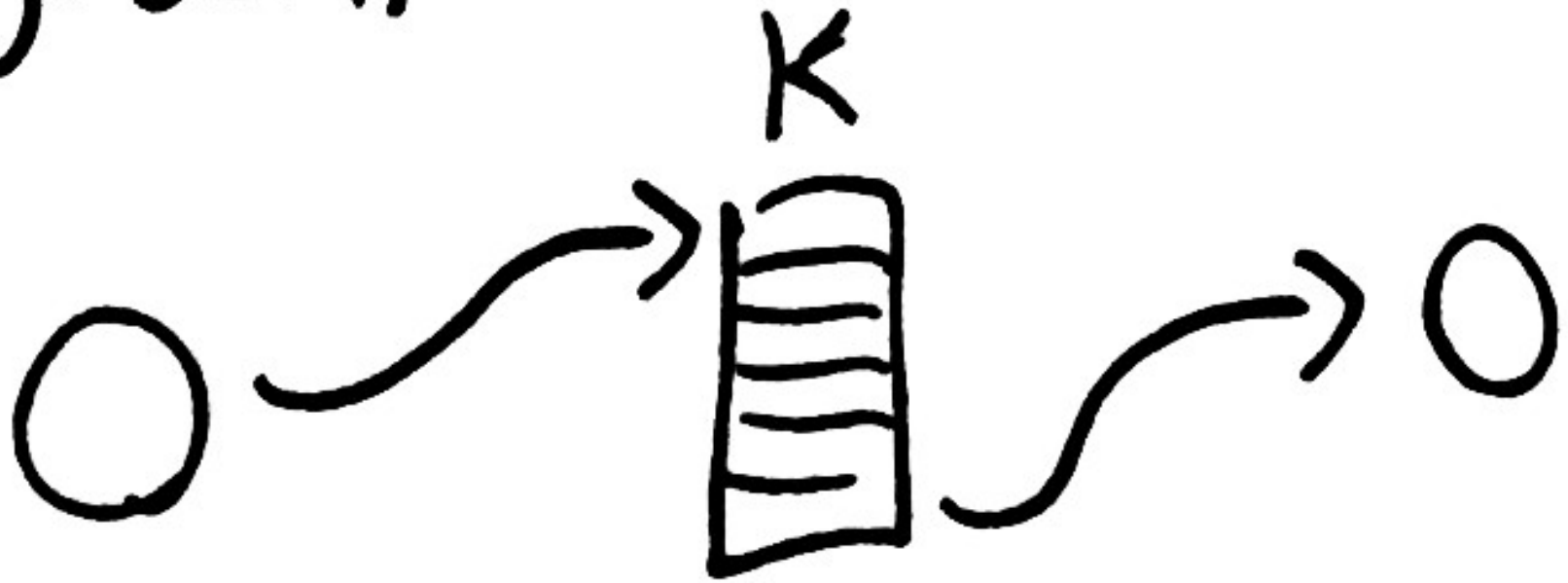
NuoDB is working on
LIVENESS DETECTION
and many of these bugs.
Stay tuned!

NuoDB strategies

- It's still young
- Enable liveness detection
- Wait 'n' see


KAFKA

Kafka is a messaging system!



Provides a sharded immutable log of messages.

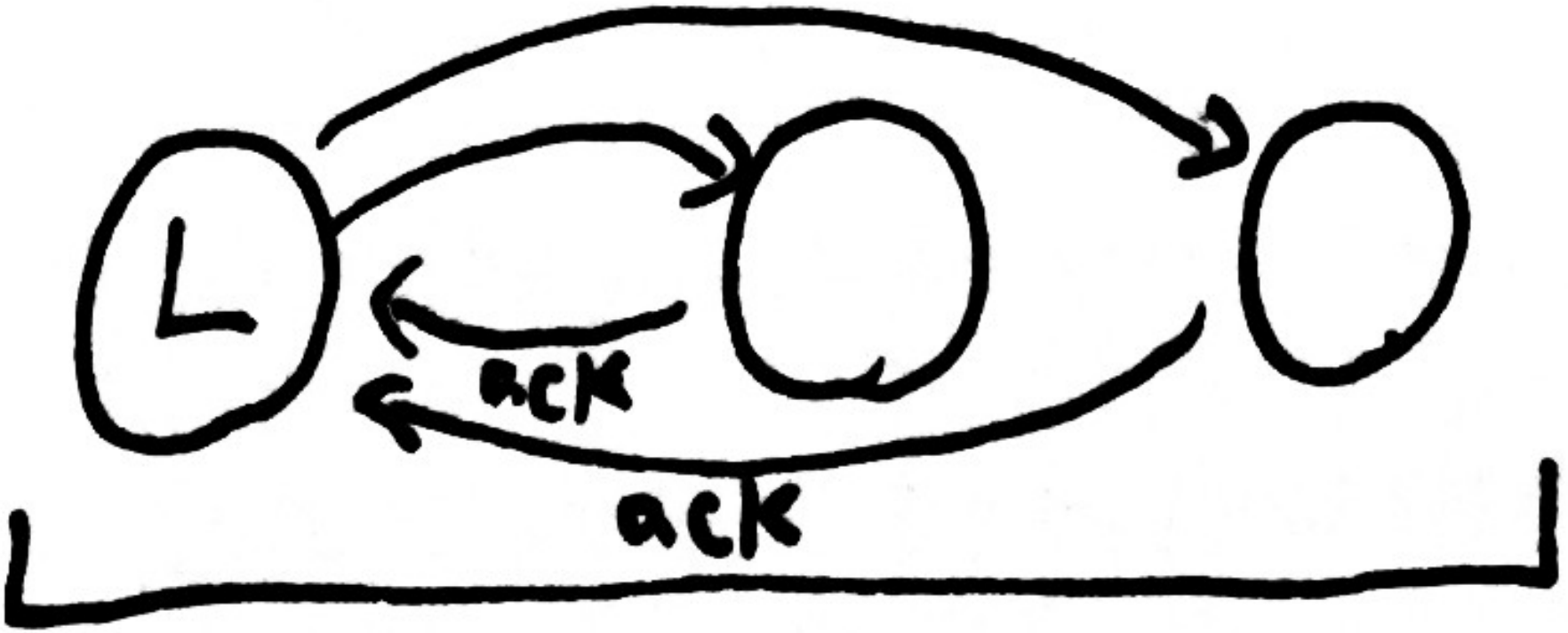
New in 0.8.0:

REPLICATION!

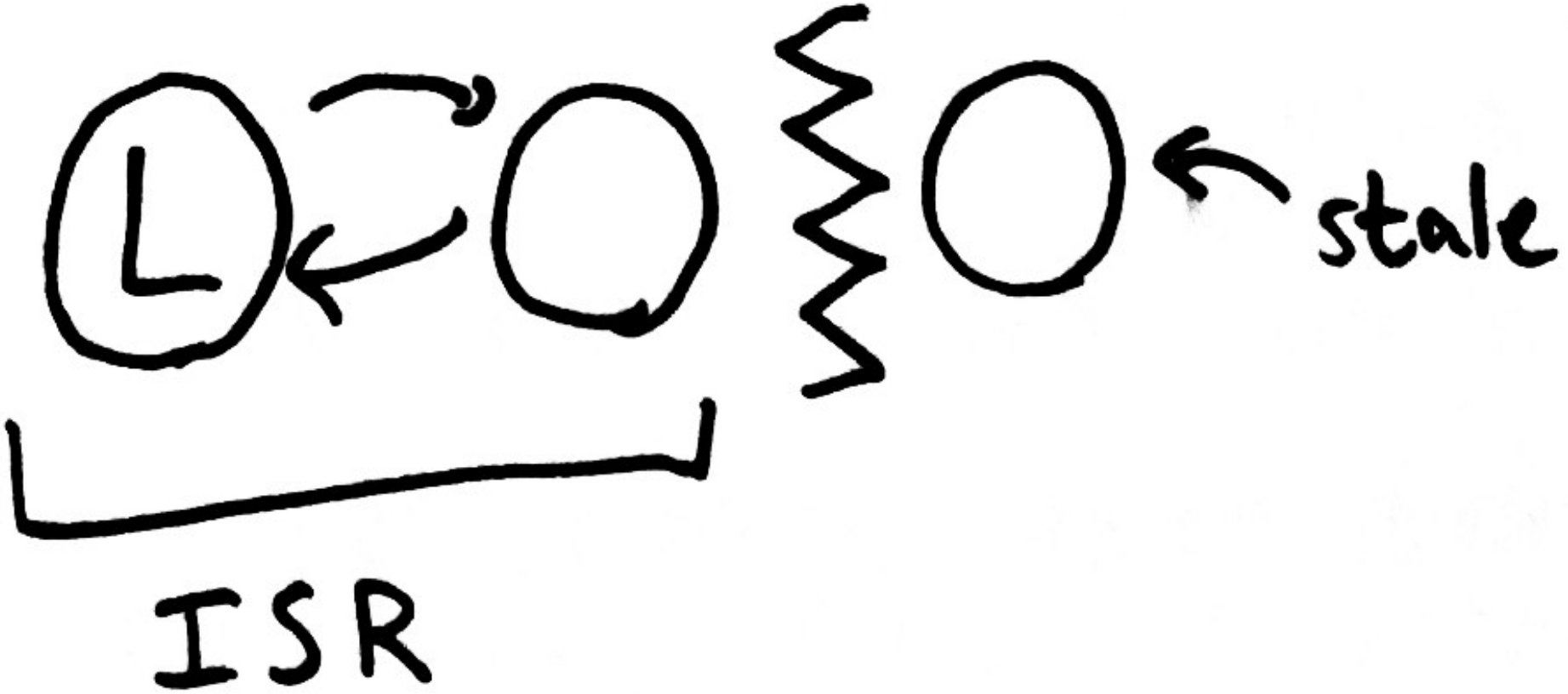
Kafka Replication: Pick CA

- Brokers within a datacenter
  - i.e., network partitioning is rare
- Strong consistency
  - replicas byte-wise identical
- Highly available
  - typical failover time: < 10ms

In a shard:



"in-sync replicas"


L
stale
ISR

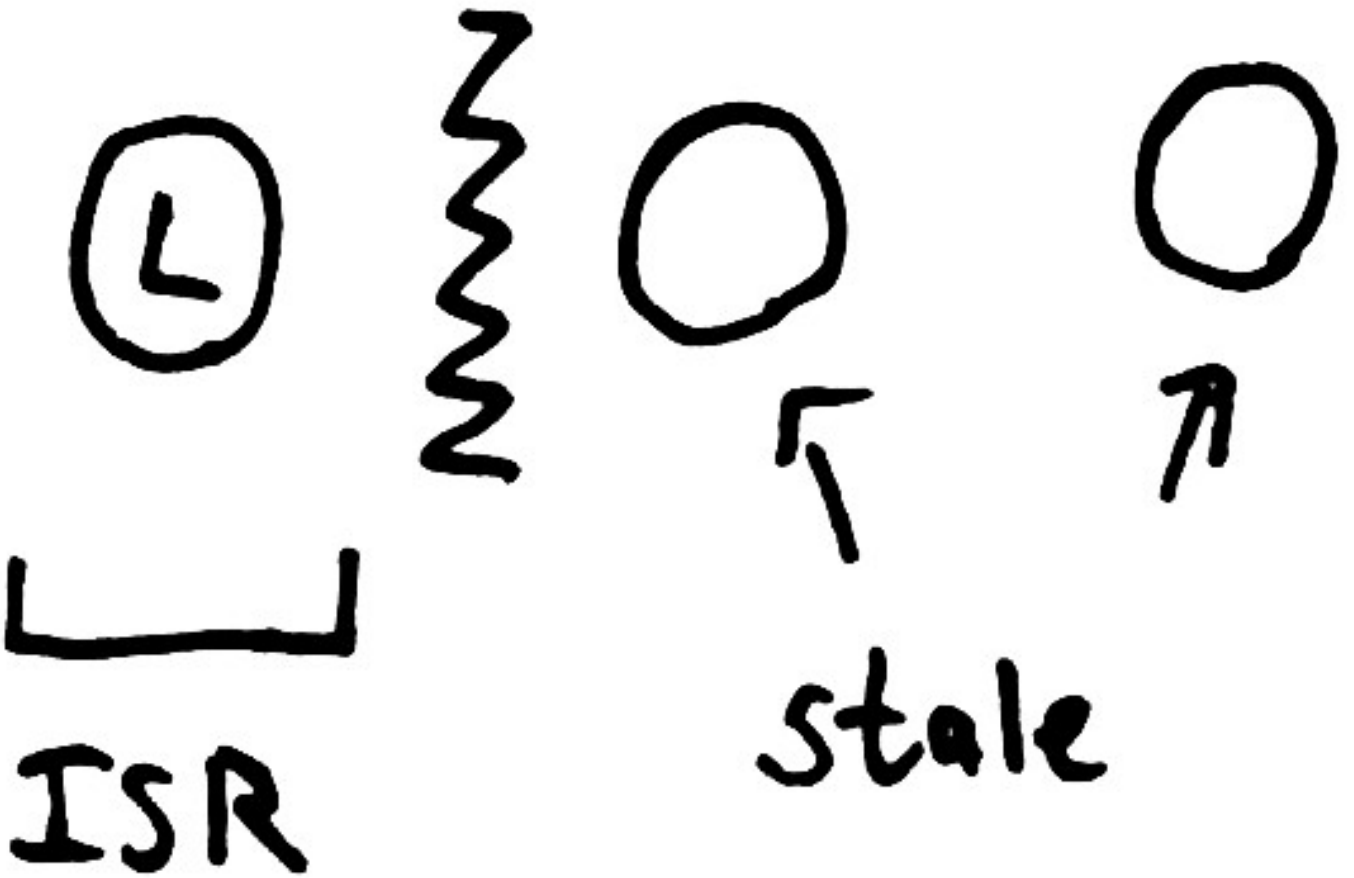
"f replicas means:

⇒ tolerate

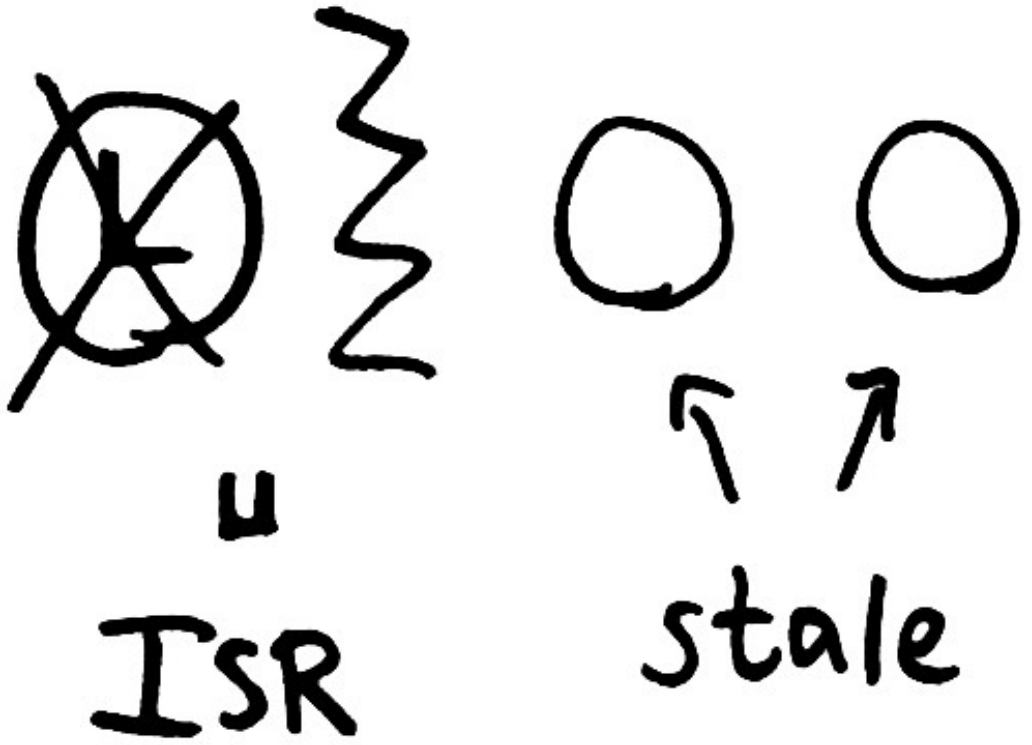
f - 1

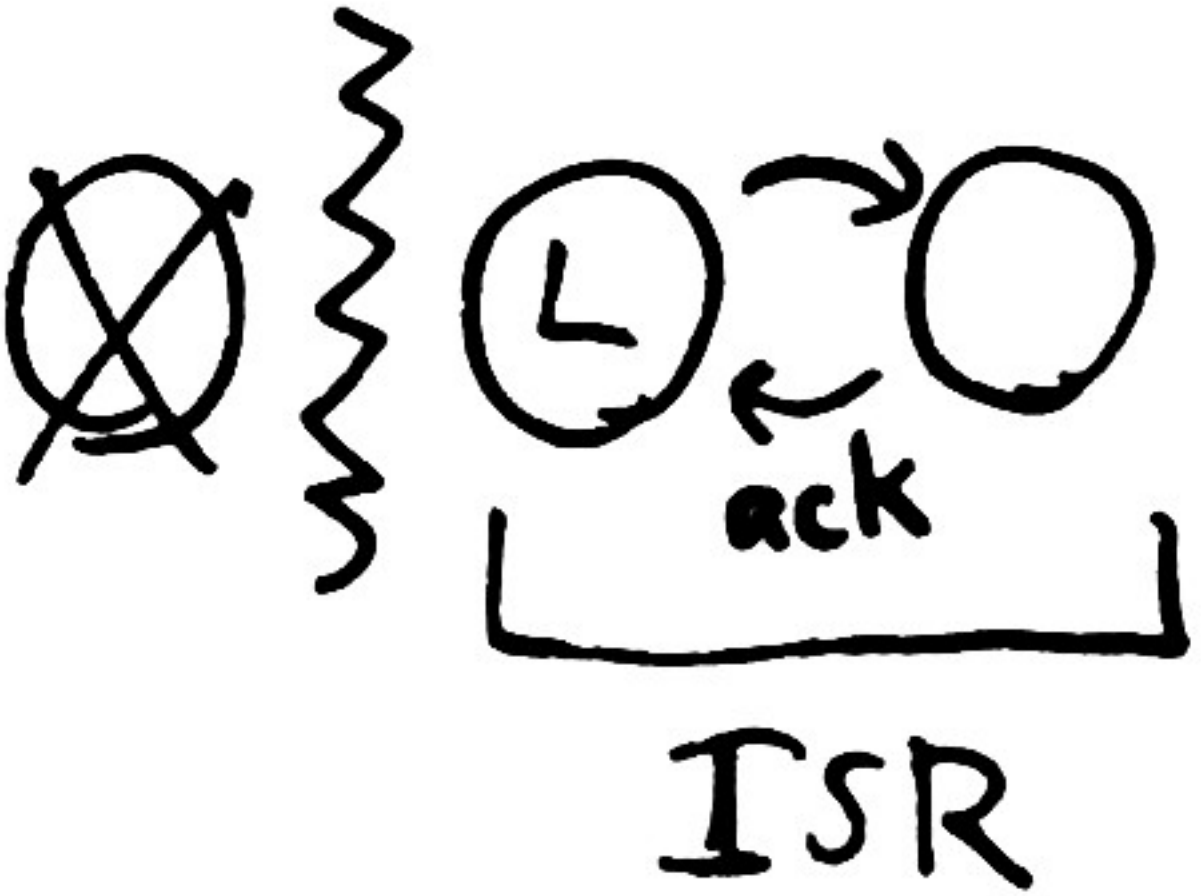
failures"

How?

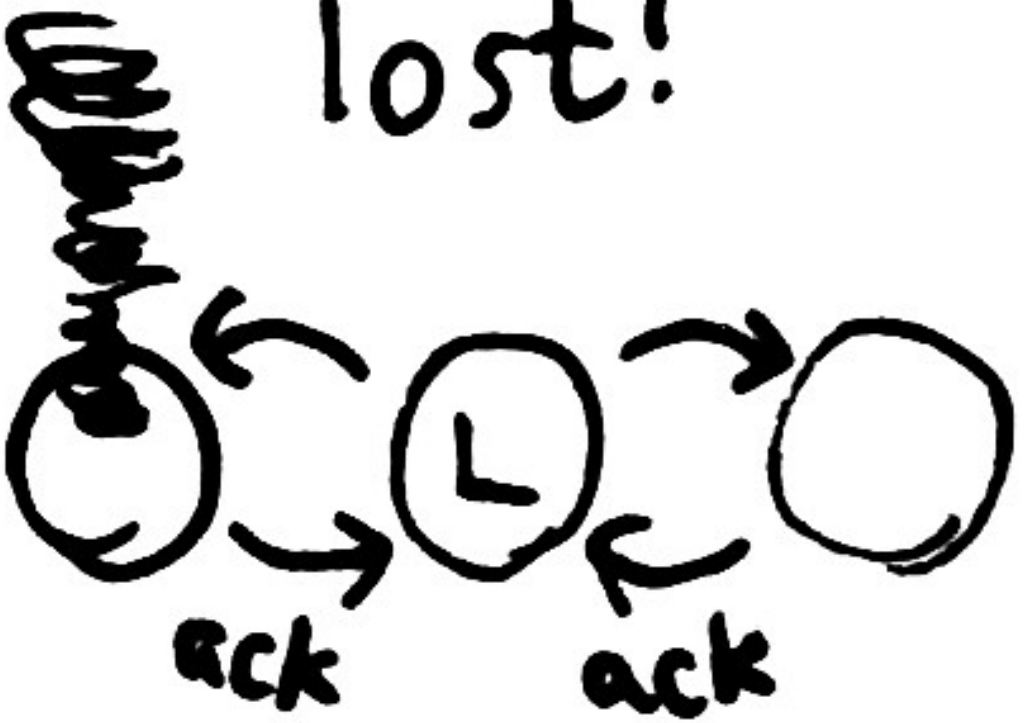


What if leader disappears?

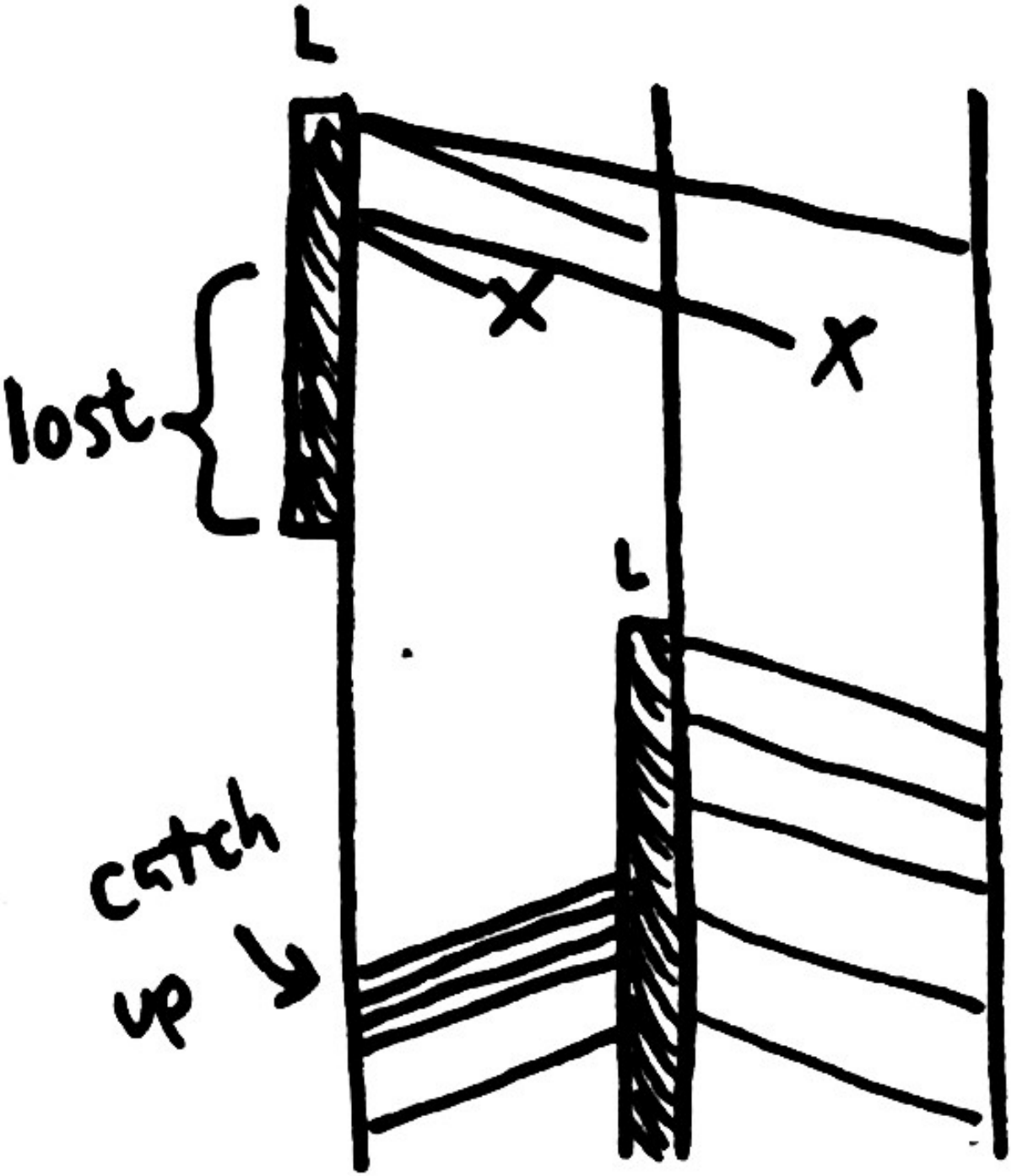



L
ack
ISR

Old leader's commits lost!

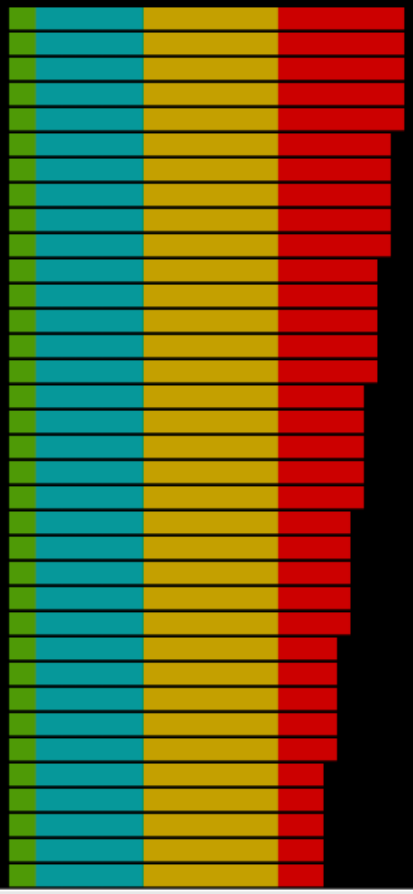


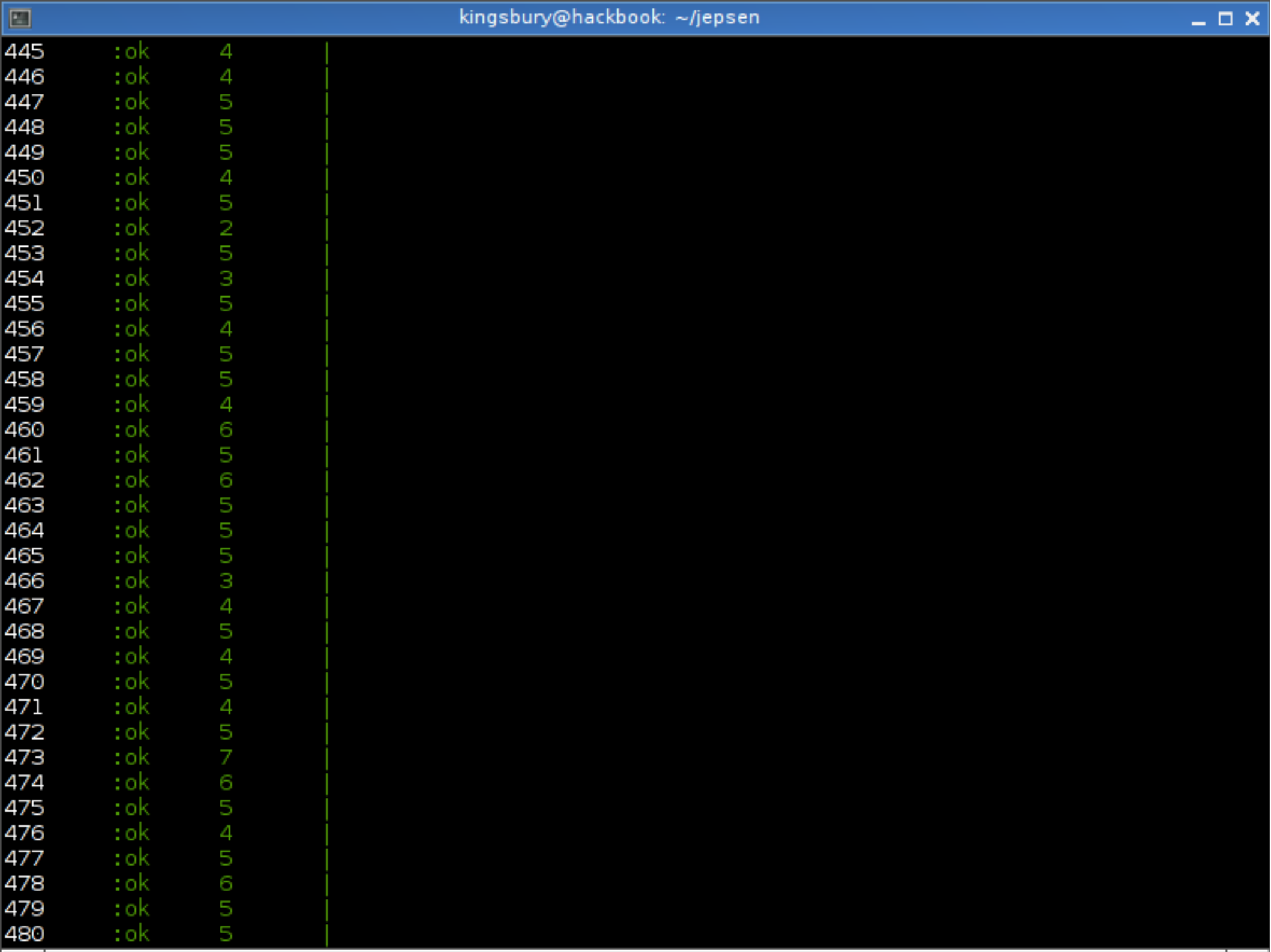
ISR

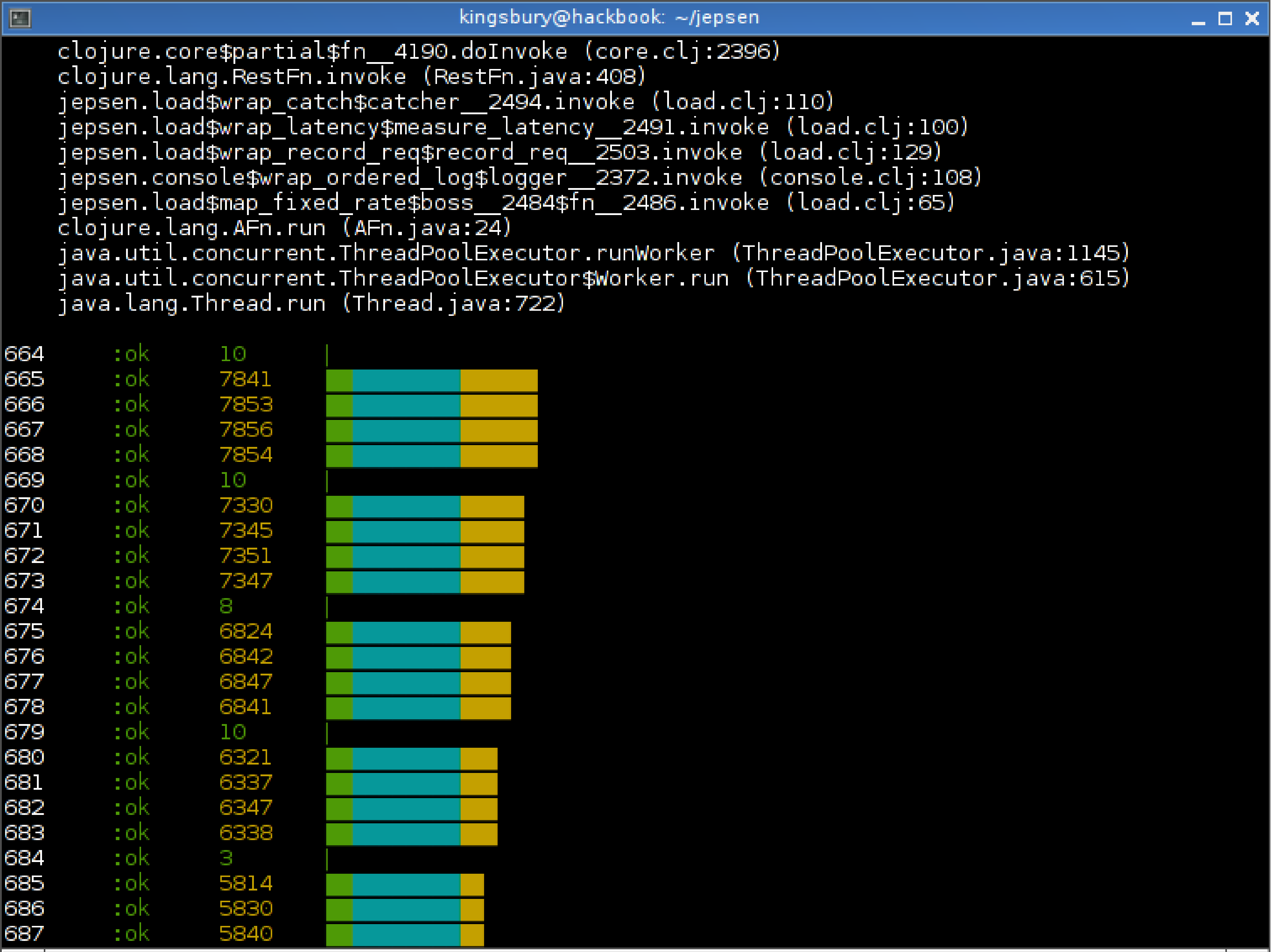

L
lost
X
X
L
catch
up

```
124     :error  10080
Failed to send messages after 3 tries.
kafka.common.FailedToSendMessageException: Failed to send messages after 3 tries.
 at kafka.producer.async.DefaultEventHandler.handle (DefaultEventHandler.scala:90)
    kafka.producer.Producer.send (Producer.scala:74)
    kafka.javaapi.producer.Producer.send (Producer.scala:32)
    clj_kafka.producer$send_message.invoke (producer.clj:19)
    jepsen.kafka$app$reify__3240.add (kafka.clj:64)
    jepsen.set_app$eval2525$fn__2548$G__2515__2551.invoke (set_app.clj:12)
    jepsen.set_app$eval2525$fn__2548$G__2514__2555.invoke (set_app.clj:12)
    clojure.lang.AFn.applyToHelper (AFn.java:163)
    clojure.lang.AFn.applyTo (AFn.java:151)
    clojure.core$apply.invoke (core.clj:619)
    clojure.core$partial$fn__4190.doInvoke (core.clj:2396)
    clojure.lang.RestFn.invoke (RestFn.java:408)
    jepsen.load$wrap_catch$catcher__2494.invoke (load.clj:110)
    jepsen.load$wrap_latency$measure_latency__2491.invoke (load.clj:100)
    jepsen.load$wrap_record_req$record_req__2503.invoke (load.clj:129)
    jepsen.console$wrap_ordered_log$logger__2372.invoke (console.clj:108)
    jepsen.load$map_fixed_rate$boss__2484$fn__2486.invoke (load.clj:65)
    clojure.lang.AFn.run (AFn.java:24)
    java.util.concurrent.ThreadPoolExecutor.runWorker (ThreadPoolExecutor.java:1145)
    java.util.concurrent.ThreadPoolExecutor$Worker.run (ThreadPoolExecutor.java:615)
    java.lang.Thread.run (Thread.java:722)

125     :ok     14668
126     :ok     14658
127     :ok     14659
128     :ok     14671
129     :ok     14662
130     :ok     14166
131     :ok     14157
132     :ok     14157
133     :ok     14165
134     :ok     14161
135     :ok     13660
136     :ok     13655
137     :ok     13657
138     :ok     13659
139     :ok     13655
140     :ok     13163
141     :ok     13152
142     :ok     13154
143     :ok     13152
144     :ok     13155
145     :ok     12660
146     :ok     12650
147     :ok     12654
148     :ok     12650
149     :ok     12651
150     :ok     12156
151     :ok     12148
152     :ok     12150
153     :ok     12151
154     :ok     12152
155     :ok     11653
156     :ok     11646
157     :ok     11648
158     :ok     11647
159     :ok     11650
```




kingsbury@hackbook: ~/jepsen
445 :ok 4 |
446 :ok 4 |
447 :ok 5 |
448 :ok 5 |
449 :ok 5 |
450 :ok 4 |
451 :ok 5 |
452 :ok 2 |
453 :ok 5 |
454 :ok 3 |
455 :ok 5 |
456 :ok 4 |
457 :ok 5 |
458 :ok 5 |
459 :ok 4 |
460 :ok 6 |
461 :ok 5 |
462 :ok 6 |
463 :ok 5 |
464 :ok 5 |
465 :ok 5 |
466 :ok 3 |
467 :ok 4 |
468 :ok 5 |
469 :ok 4 |
470 :ok 5 |
471 :ok 4 |
472 :ok 5 |
473 :ok 7 |
474 :ok 6 |
475 :ok 5 |
476 :ok 4 |
477 :ok 5 |
478 :ok 6 |
479 :ok 5 |
480 :ok 5 |


kingsbury@hackbook: ~/jepsen
clojure.core$partial$fn__4190.doInvoke (core.clj:2396)
clojure.lang.RestFn.invoke (RestFn.java:408)
jepsen.load$wrap_catch$catcher__2494.invoke (load.clj:110)
jepsen.load$wrap_latency$measure_latency__2491.invoke (load.clj:100)
jepsen.load$wrap_record_req$record_req__2503.invoke (load.clj:129)
jepsen.console$wrap_ordered_log$logger__2372.invoke (console.clj:108)
jepsen.loadmap_fixed_rateboss__2484$fn__2486.invoke (load.clj:65)
clojure.lang.AFn.run (AFn.java:24)
java.util.concurrent.ThreadPoolExecutor.runWorker (ThreadPoolExecutor.java:1145)
java.util.concurrent.ThreadPoolExecutor$Worker.run (ThreadPoolExecutor.java:615)
java.lang.Thread.run (Thread.java:722)
664 :ok 10
665 :ok 7841
666 :ok 7853
667 :ok 7856
668 :ok 7854
669 :ok 10
670 :ok 7330
671 :ok 7345
672 :ok 7351
673 :ok 7347
674 :ok 8
675 :ok 6824
676 :ok 6842
677 :ok 6847
678 :ok 6841
679 :ok 10
680 :ok 6321
681 :ok 6337
682 :ok 6347
683 :ok 6338
684 :ok 3
685 :ok 5814
686 :ok 5830
687 :ok 5840

Kafka

100% acknowledged

50% false positives

0% false negatives

2 well-timed failures on a single node mean lost data.

- ISR must shrink
- All ISR nodes must lose ZK connection.
- "All" could be 1 node.

How do you recover tolerance to more failures? Use majority.

- ISR lower bound of N/2+1

Majority quorums are provably optimal for NDC consensus fault-tolerance.

– Peleg & Wool, '93

Kafka is considering stronger bounds on correctness; must balance performance.

Kafka strategies

- 0.8 is still beta

- If present, turn on stronger safety rules

- Roll your own replication?


Cassandra!

Dynamo system

- Partitioned hash ring
- Quorums & handoff
- LWW only, not vclocks

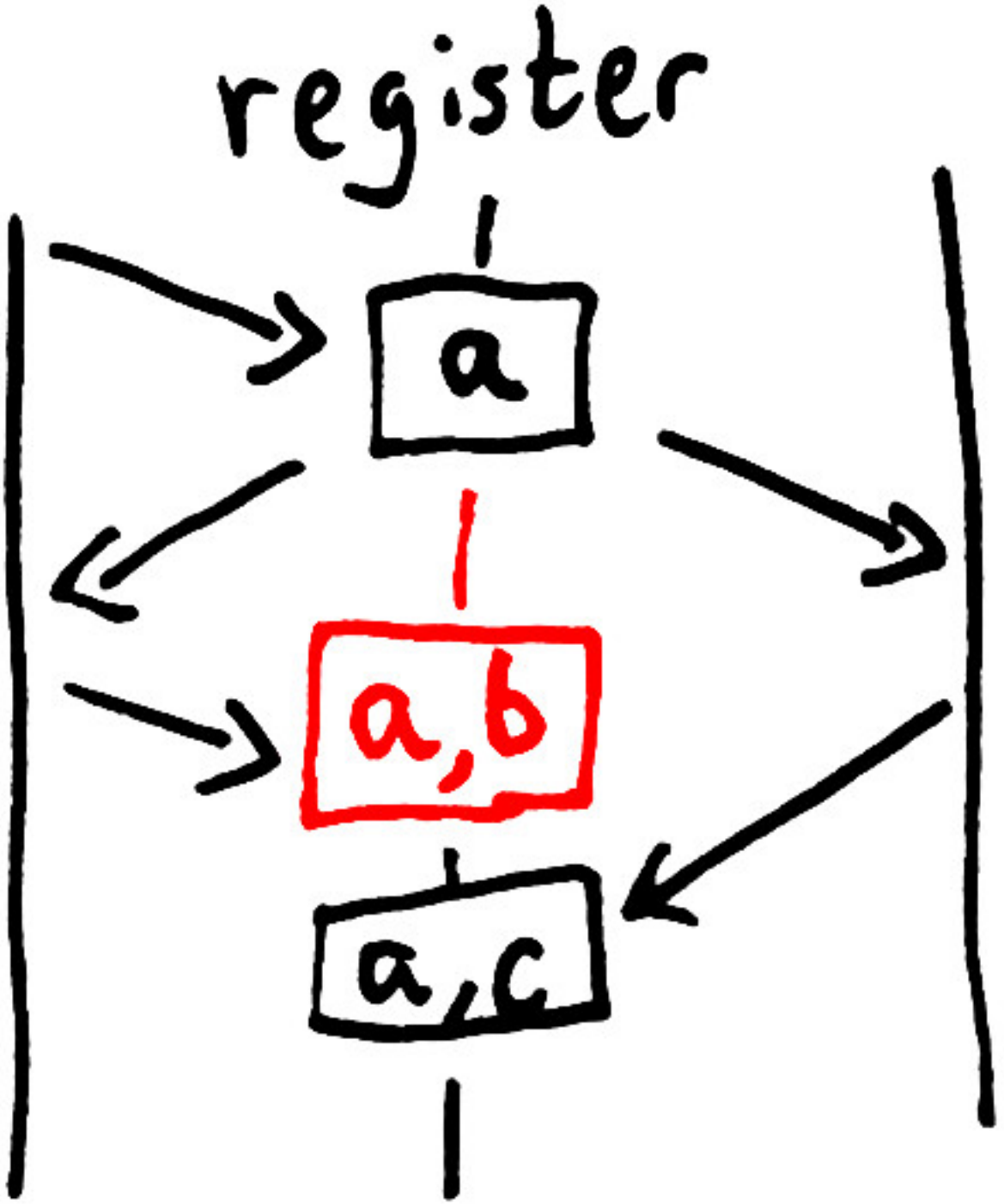
"But Last Write Wins is bad, right?"

Well Sally, it depends!

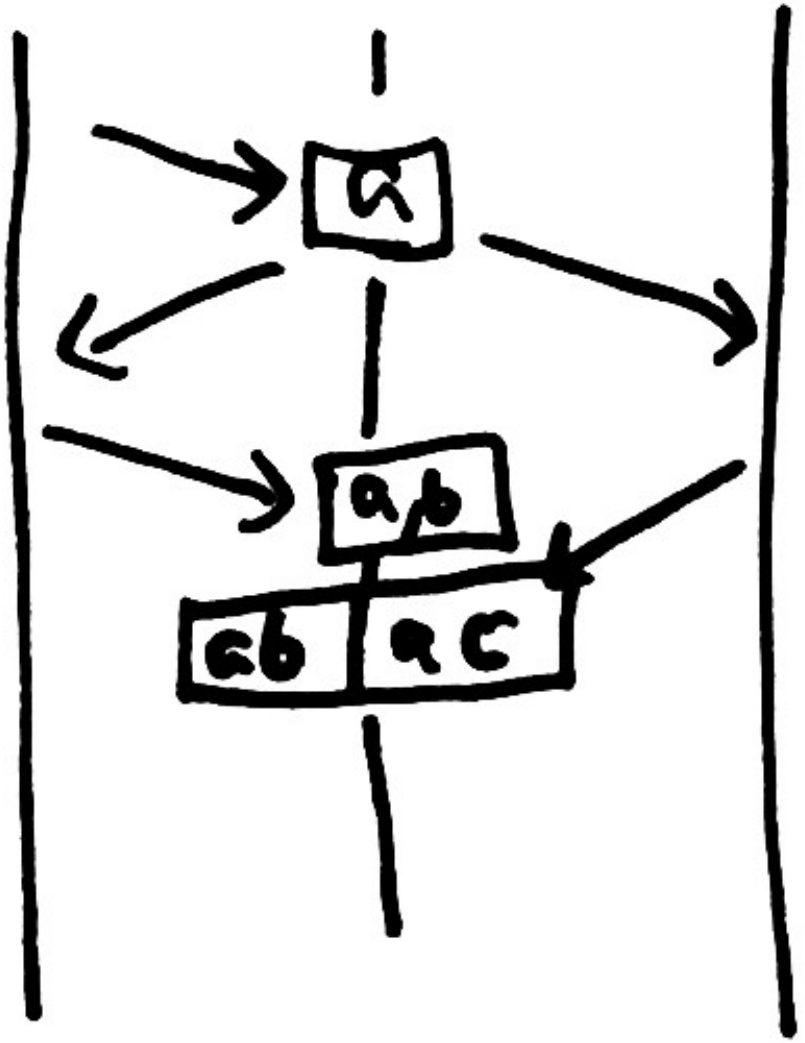
You can't safely change an LWW register.

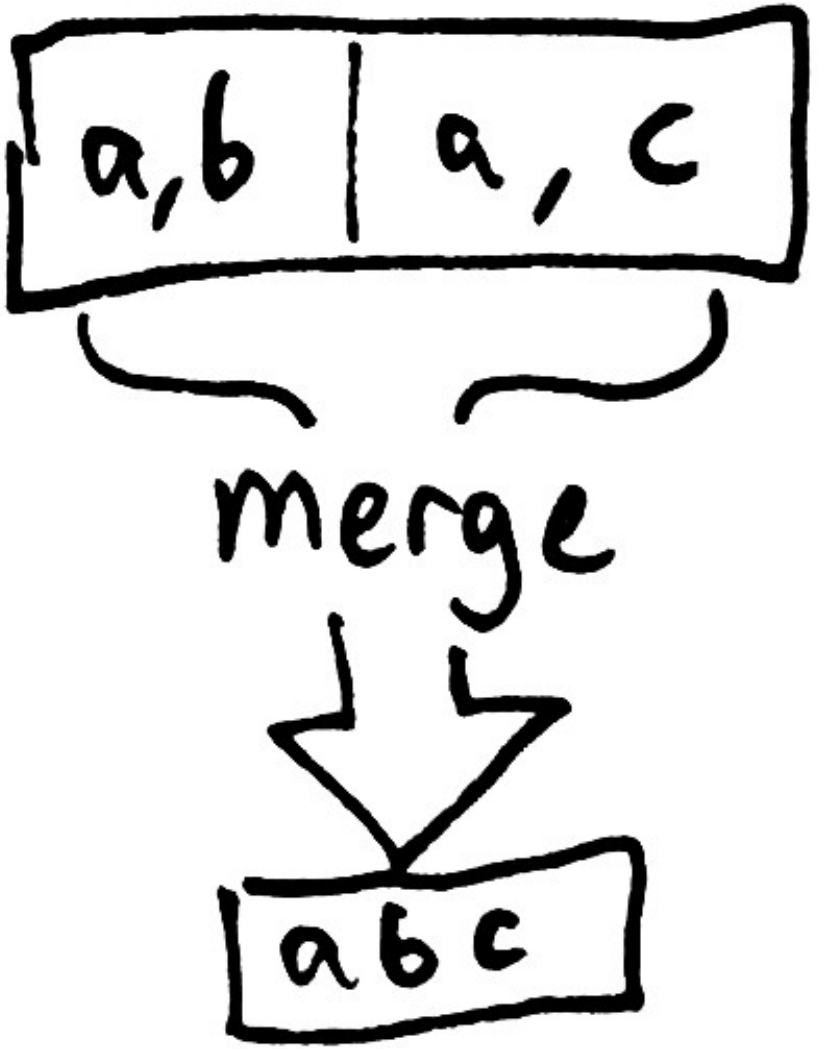
Safe?

- Guarantee that your write is causally connected to a future read.


register
a
a,b
a,c

Vector clocks identify these causally isolated writes, and give you both copies.


a
ab
ab
ac


a,b
a, c
merge
abc

Merge fn must be

- Associative
- Commutative
- Idempotent

CRDTs!

Early in Cassandra history, chose not to use vclocks for performance.

Commensurately, no safe way to modify a cell.

Cassandra Cell Change
(quorum r+w, locks)

44% acknowledged

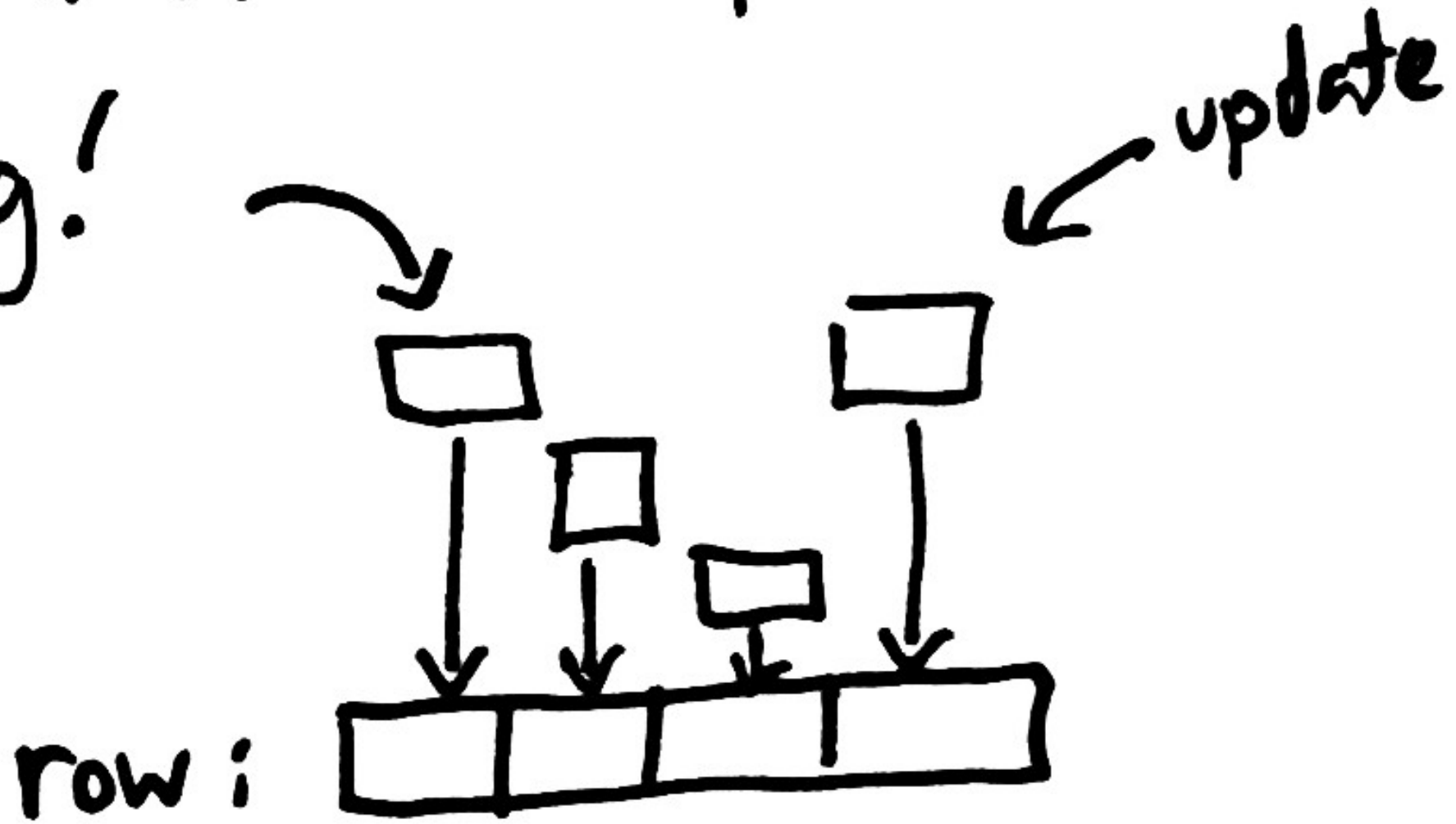
27% false positives

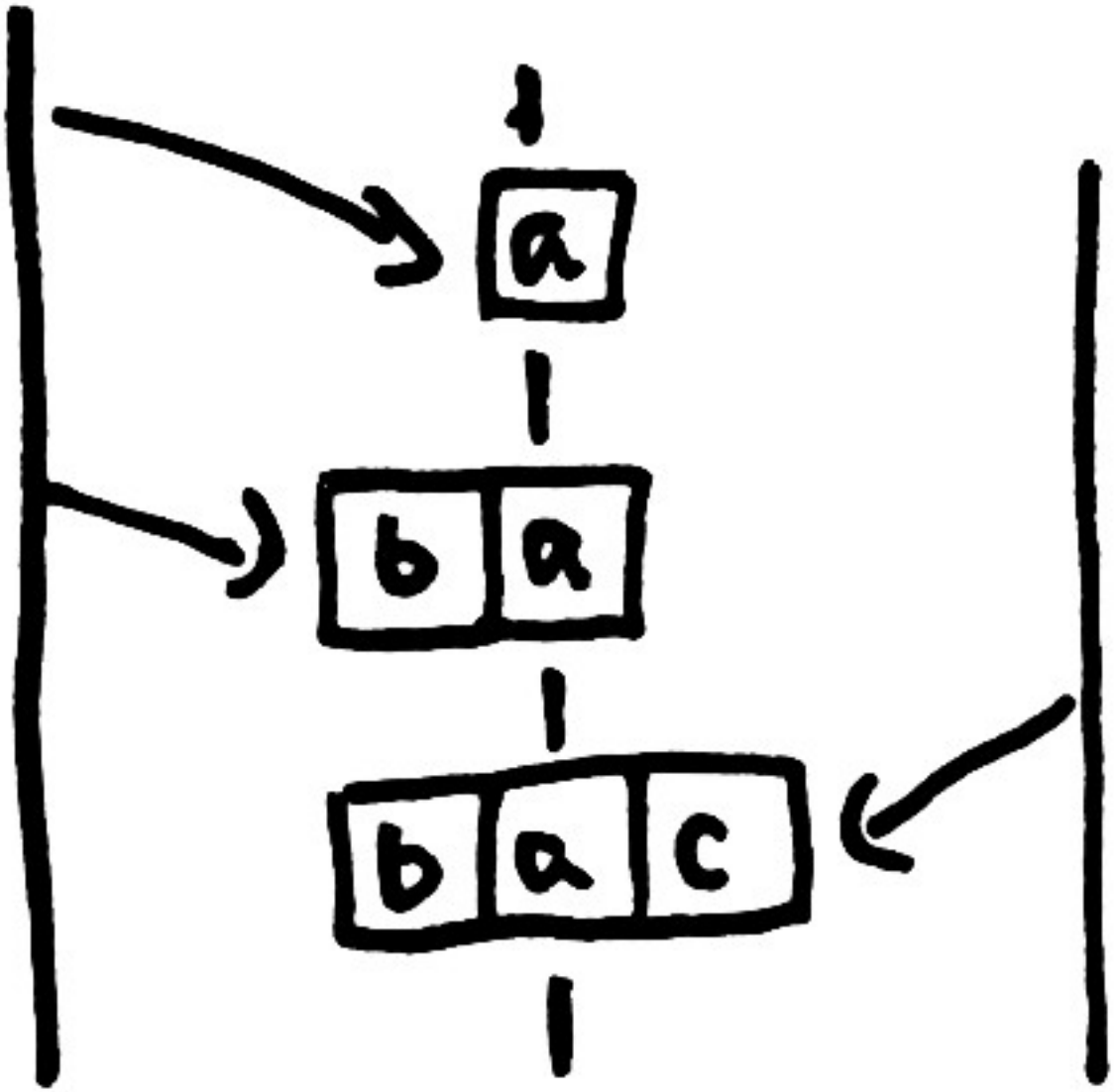
0% false negatives

or tables. Cassandra supports tuning availability and consistency, and always gives you partition tolerance. Cassandra can be tuned to give you strong consistency in the CAP sense where data is made consistent across all the nodes in a distributed database cluster. A user

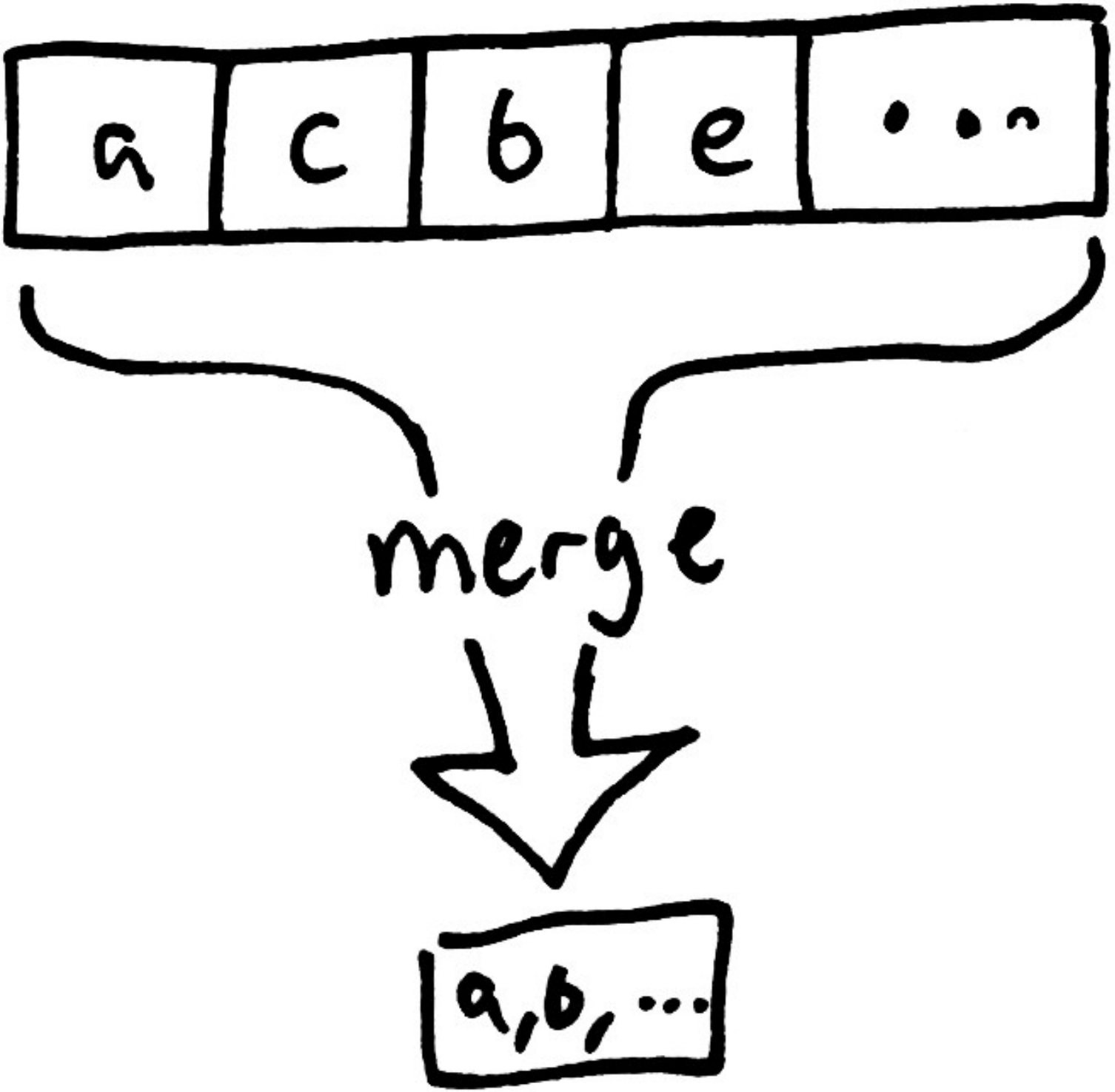
Instead, Cass. has evolved efficient ways to write every change to a distinct cell.

Immutable operation log!

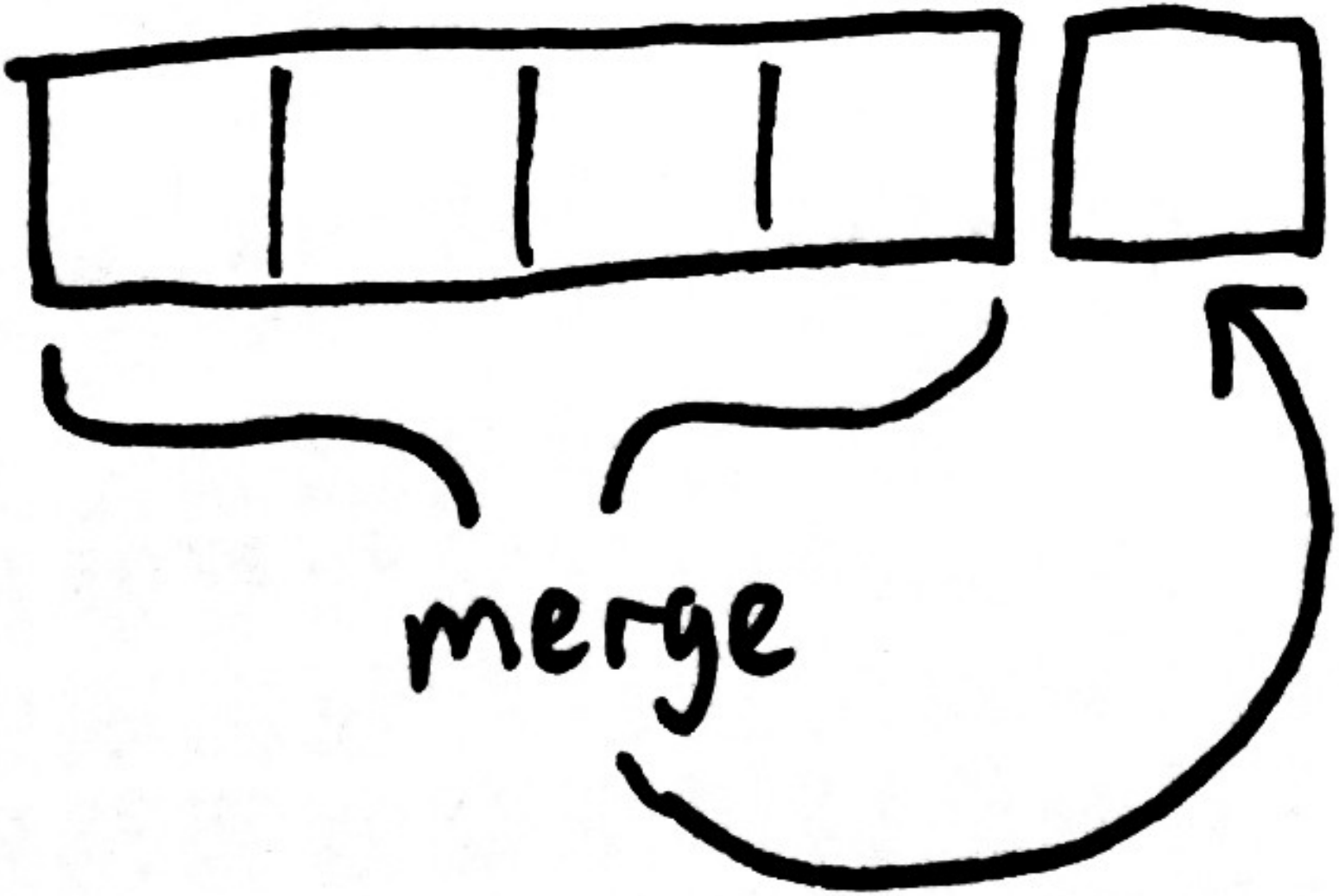



a
b
a
b
a
c

Reads require merge functions

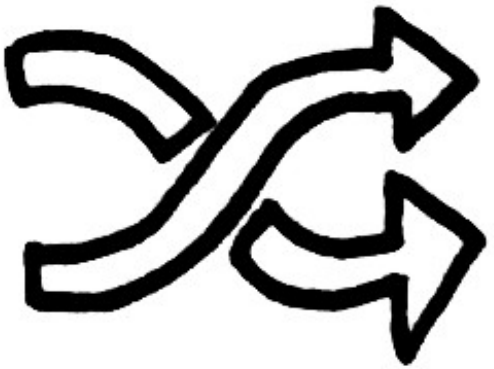

a
c
b
e
merge
a,b,...

Garbage Collection?

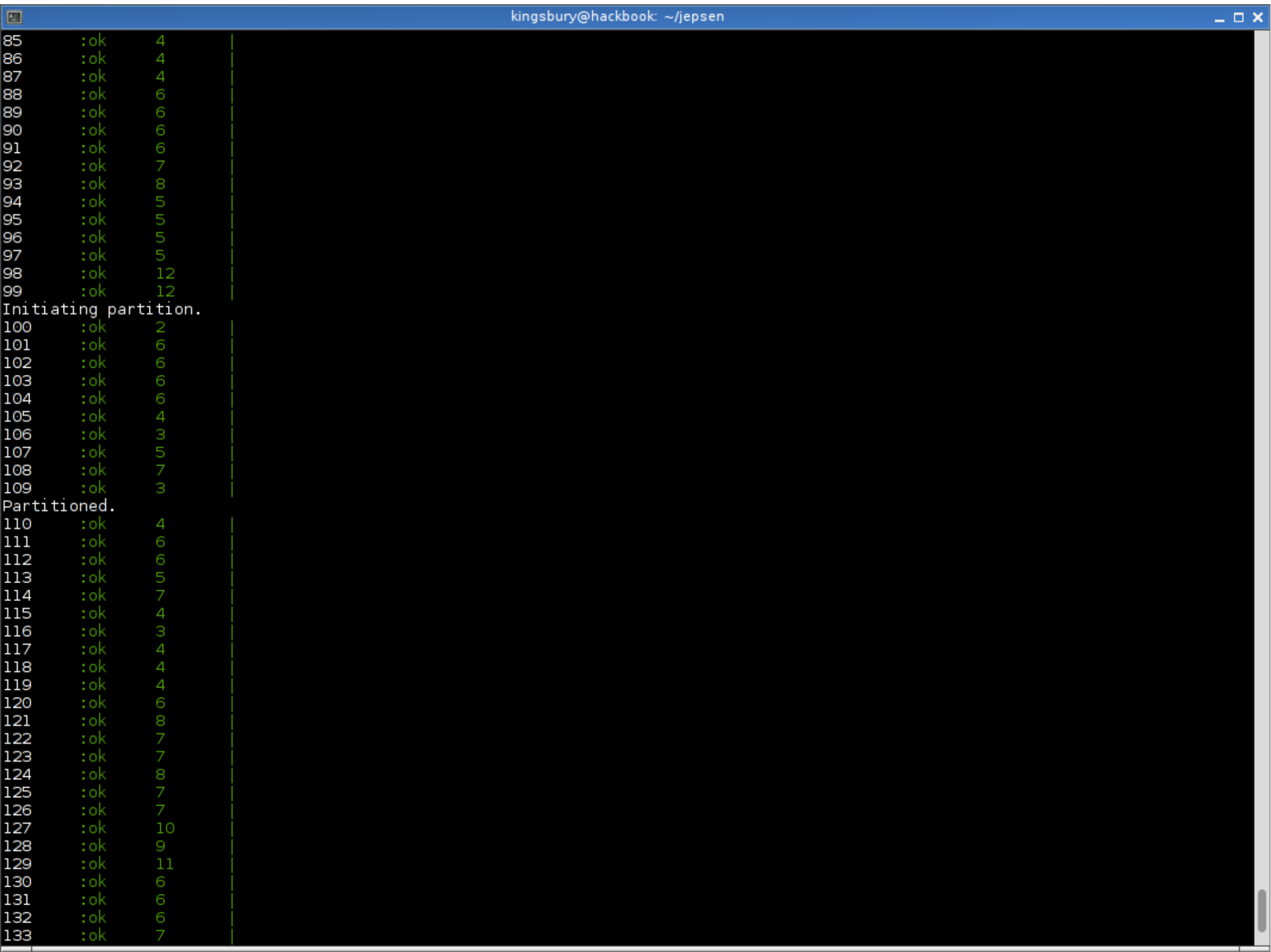


Vector clocks handle identifying old, unneeded data.

Since there is no global ordering for updates, still need commutative merges.



So! Cassandra and Riak have similar consistency semantics. CQL provides limited CRDTs.


kingsbury@hackbook: ~/jepsen
85 :ok 4
86 :ok 4
87 :ok 4
88 :ok 6
89 :ok 6
90 :ok 6
91 :ok 6
92 :ok 7
93 :ok 8
94 :ok 5
95 :ok 5
96 :ok 5
97 :ok 5
98 :ok 12
99 :ok 12
Initiating partition.
100 :ok 2
101 :ok 6
102 :ok 6
103 :ok 6
104 :ok 6
105 :ok 4
106 :ok 3
107 :ok 5
108 :ok 7
109 :ok 3
Partitioned.
110 :ok 4
111 :ok 6
112 :ok 6
113 :ok 5
114 :ok 7
115 :ok 4
116 :ok 3
117 :ok 4
118 :ok 4
119 :ok 4
120 :ok 6
121 :ok 8
122 :ok 7
123 :ok 7
124 :ok 8
125 :ok 7
126 :ok 7
127 :ok 10
128 :ok 9
129 :ok 11
130 :ok 6
131 :ok 6
132 :ok 6
133 :ok 7

Cassandra CQL sets

100% acknowledged

0% false positives

0% false negatives

CQL collections are partition tolerant!

- But semantics are subtle...

BEWARE DELETES

Cassandra counters

at consistency = quorum

+/- 50% actual value

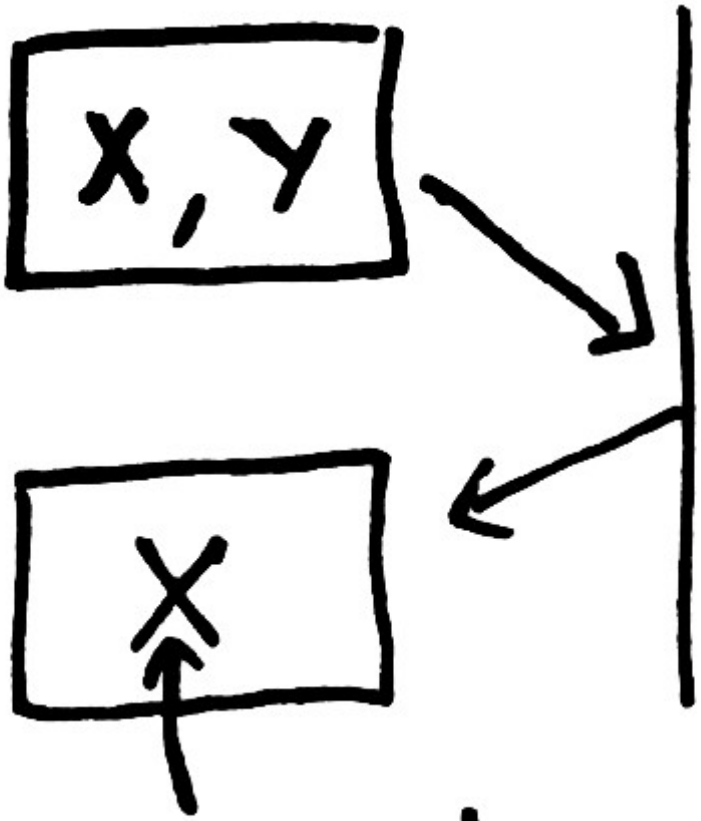
What about ISOLATION?

"From a transactional ACID... standpoint, [atomic row ops] give Cassandra transactional AID support"

—Datastax "About Writes"

"Atomic"

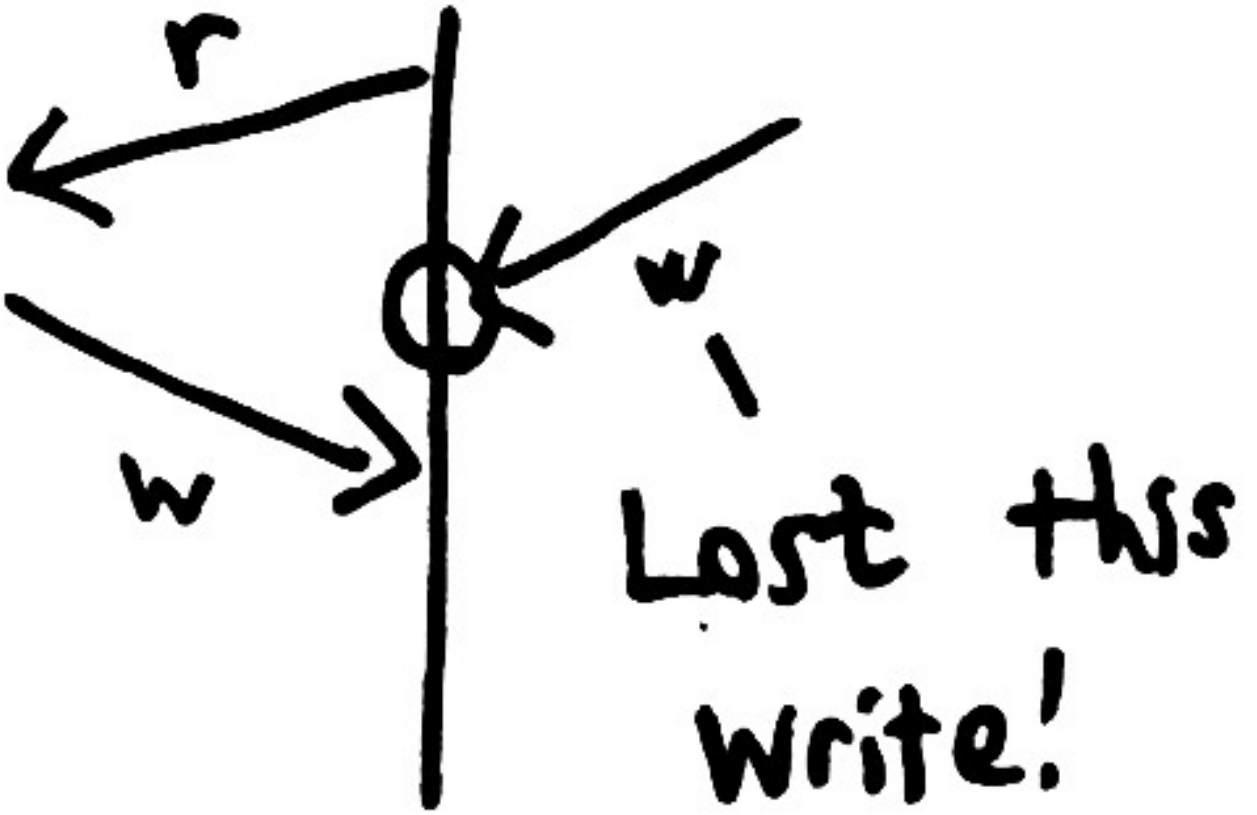
"Isolated"



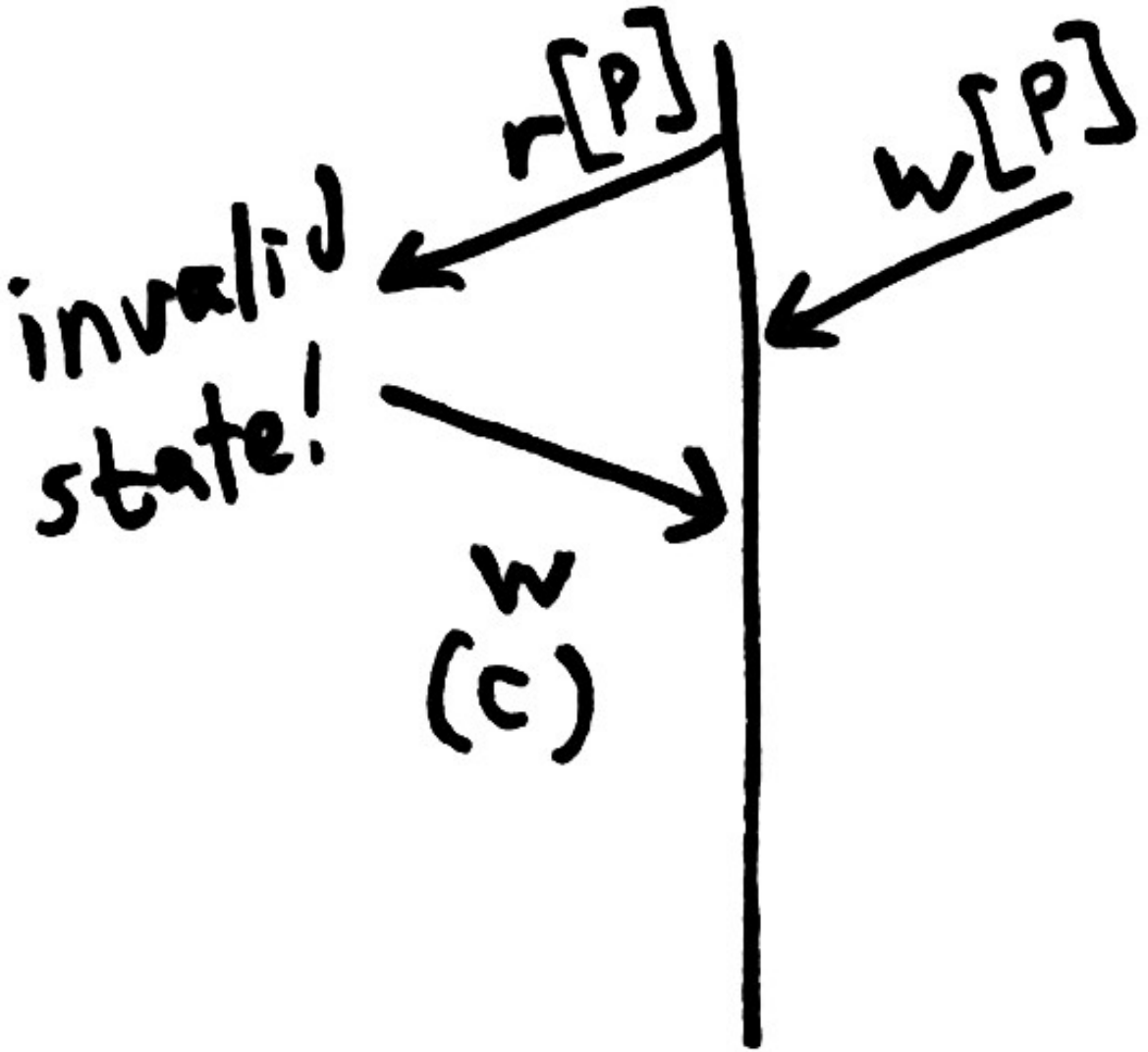
Wrong!

... um...

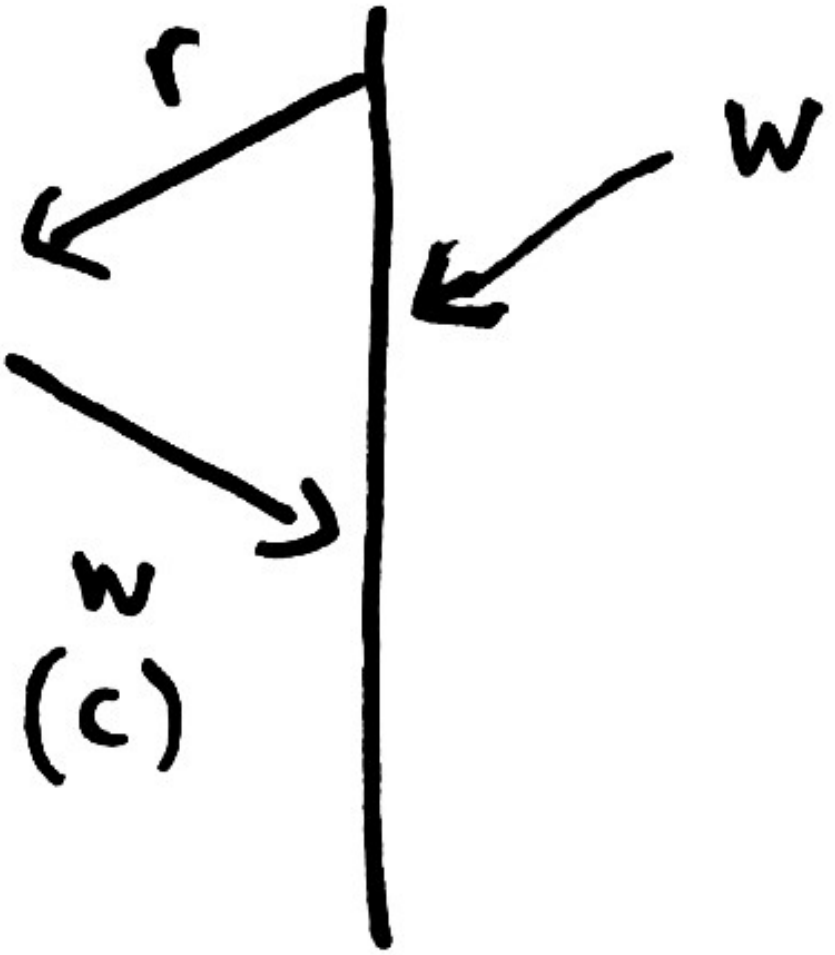
P4: Lost update (the LWW problem!)



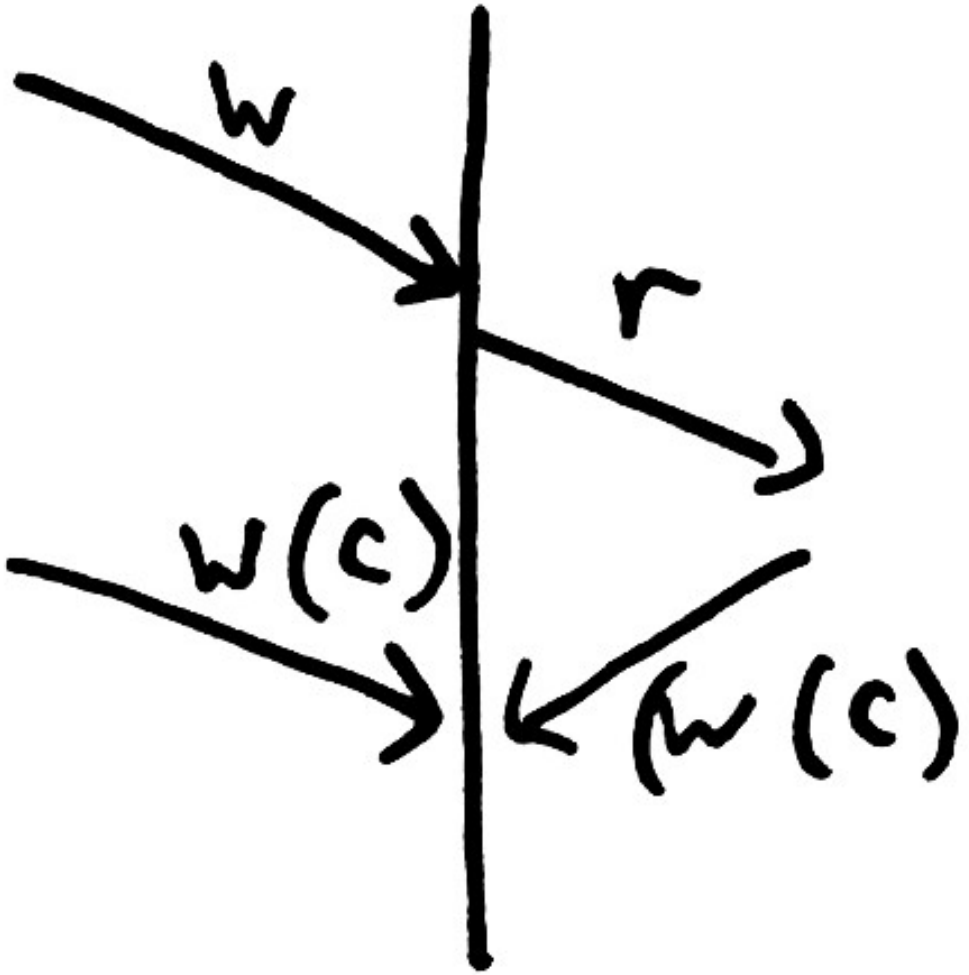
P3: Phantom



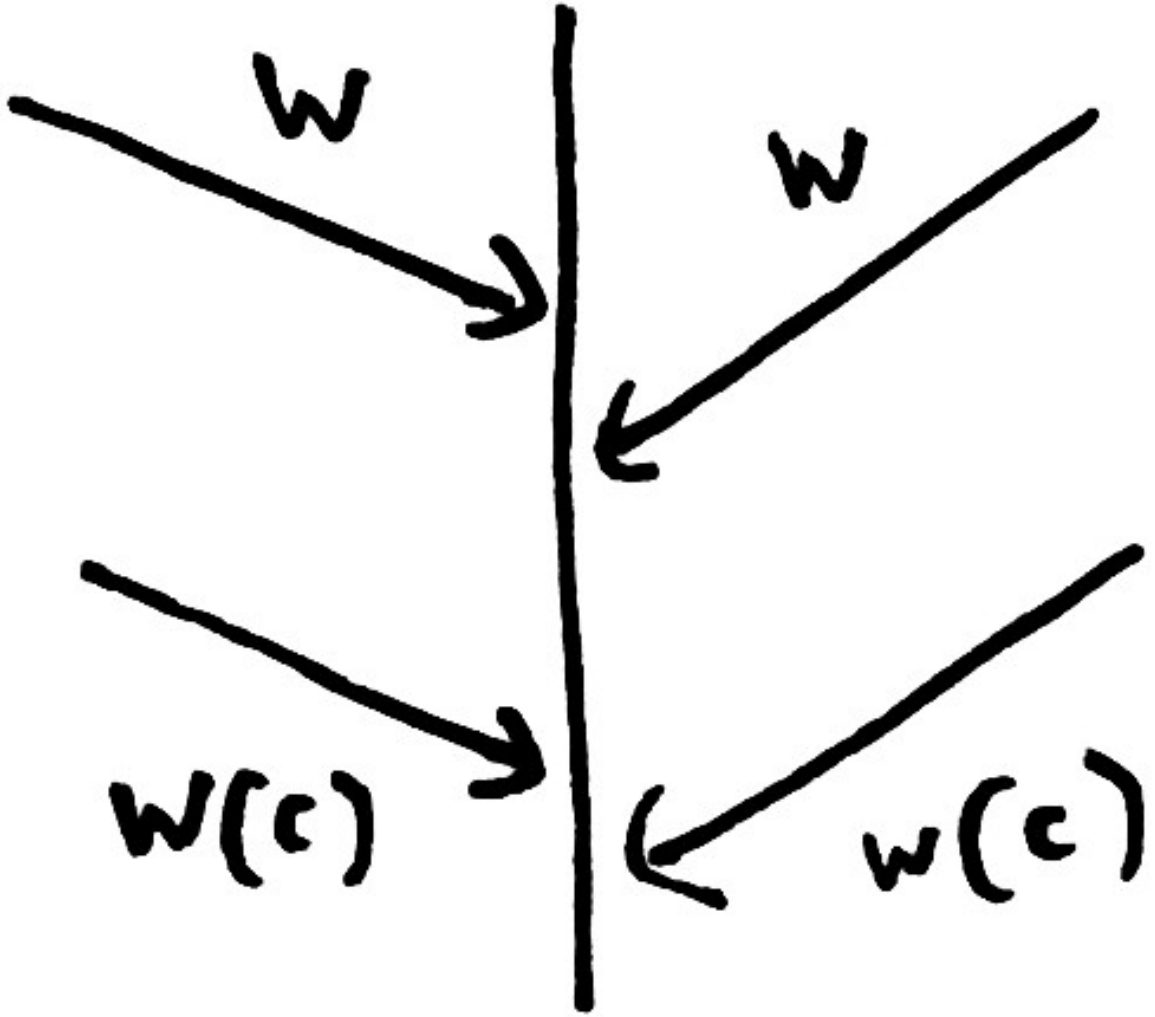
P2: Fuzzy Read

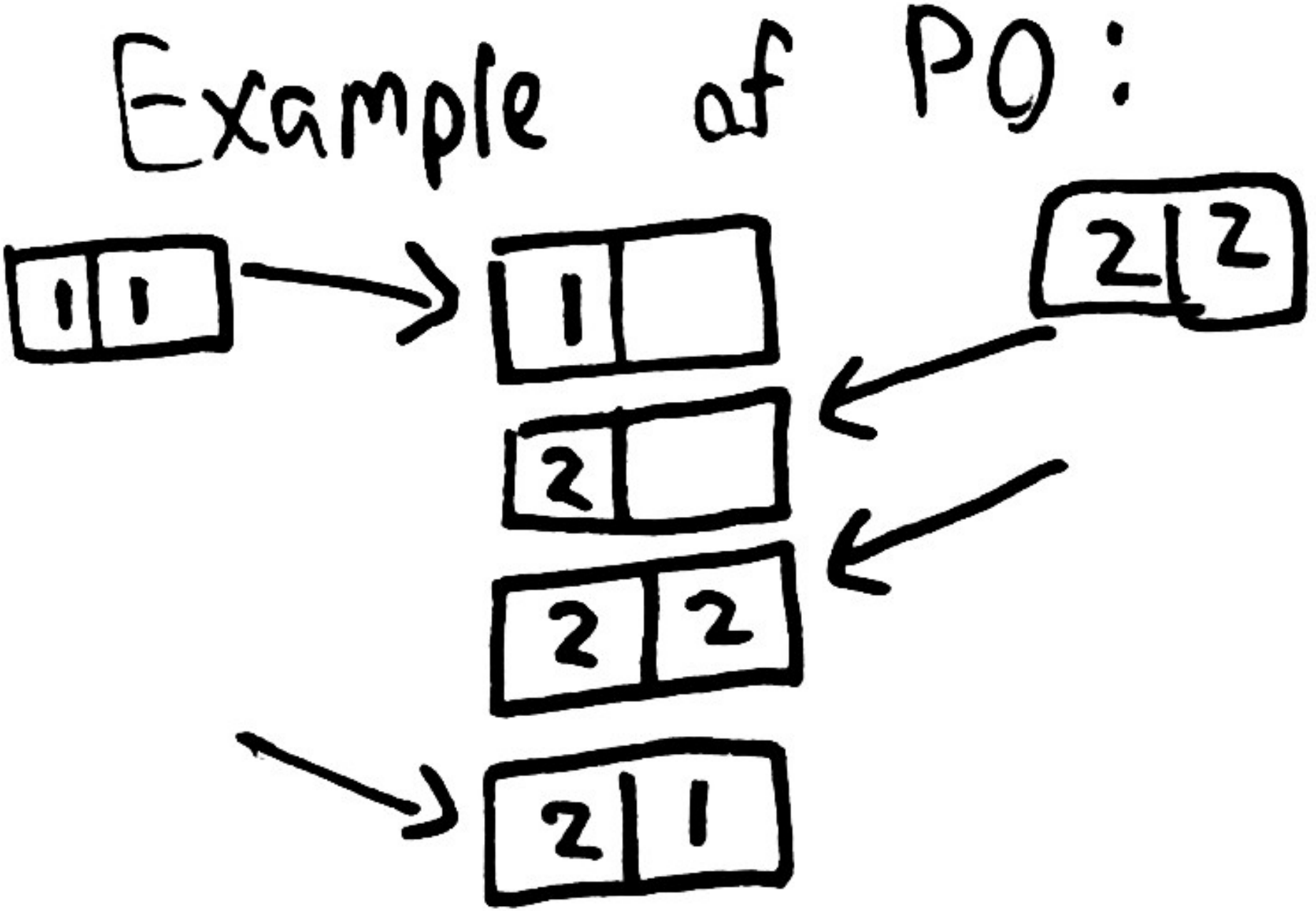


P2: Dirty Read



P0: Dirty write




Example of PO:
1 1
1
2 2
2
2 2
2 1

"ANSI SQL Isolation should ... require PO for all isolation levels"

-Berenson, et al

MSR-TR-95-51

Cassandra 1.1 as row-level updates are now made in isolation. Cassandra 1.1 guarantees that if you update both the login and password in the same update (for the same row key) then no concurrent read may see only a partial update.

From a transactional ACID (atomic, consistent, isolated, durable) standpoint, this enhancement now gives Cassandra transactional AID support. A write is isolated at the row-level in the storage engine.

Actually, they are isolated in the sense of ACID,

Certainly there is nothing like ACID, but a single multi-cell update to a given record is guaranteed to be _atomic_, and if you have two concurrent multi-cell updates to a single record, they are guaranteed to eventually resolve to a consistent ordering of those operations

The difference is that all the cells with the same `id` would belong to the same partition and stored together, so you'd be able to write them atomically and read them together cheaply in a single operation.

It wasn't really mentioned in the article but if you do both {X1,Y1} and {X2,Y2} as full row updates then no you can't end up with a mixed row.

http://www.datastax.com/dev/blog/row-level-isolation

In Cassandra the second update agreed upon by the cluster would "win", so either (X1,Y1) or (X2,Y2).

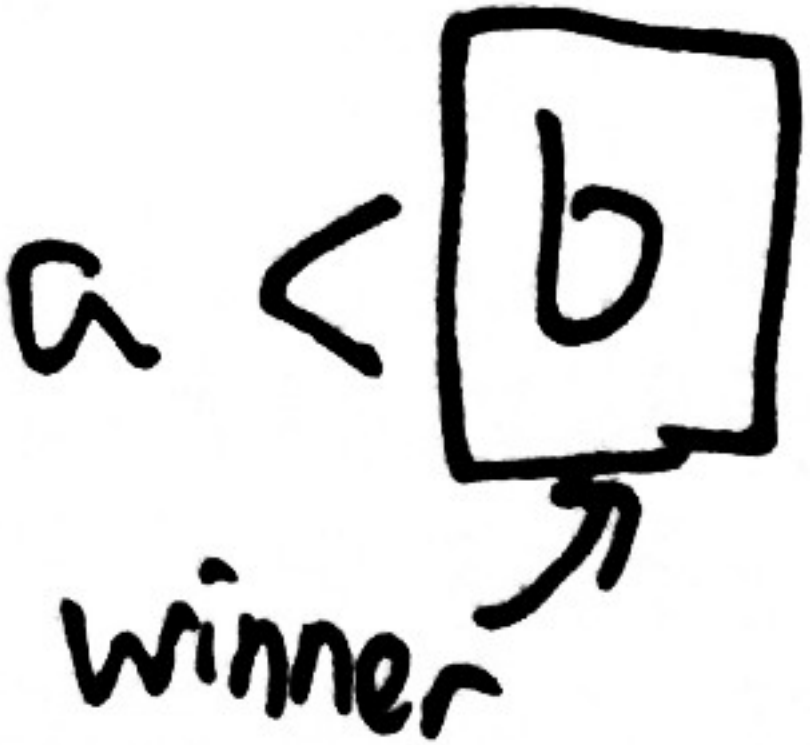
Cassandra allows PO (and more!)

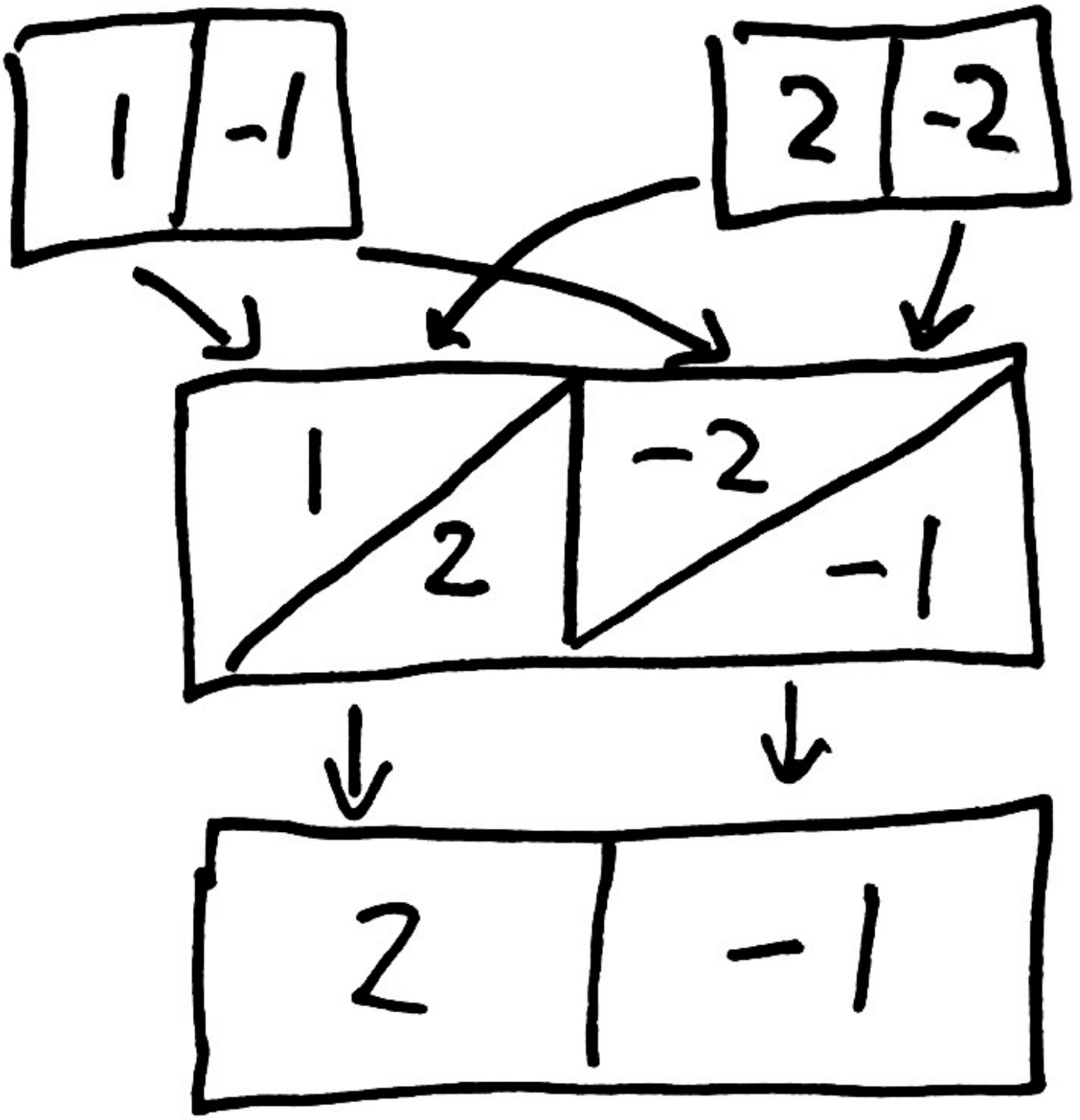
In Cassandra, write order doesn't matter. LWW always takes prio.

What if timestamps are equal?

$$t_a = t_b$$

Compare values



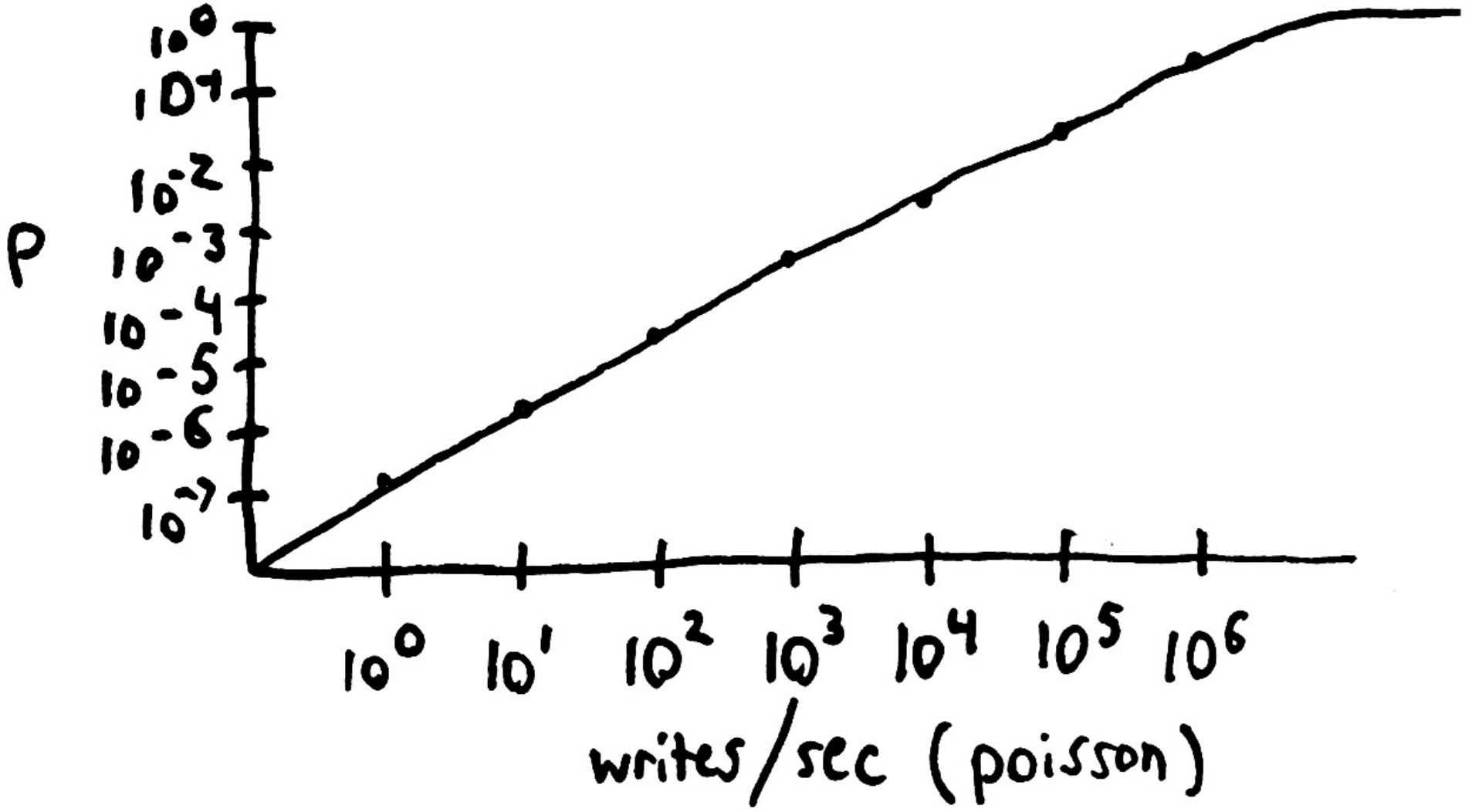

1
-1
2
-2
1
2
-2
-1
2
-1

"But timestamps never conflict!"

"But timestamps never conflict!"

It Depends!™

Probability of a microsecond conflict visible in a read:



10 writes/sec

1 read/sec

34% chance of collision in a day

2 writes per row,
separated by mean
latency 100 ms
(poisson processes)

$$p(\text{corrupt}) = 5 \times 10^{-6}$$

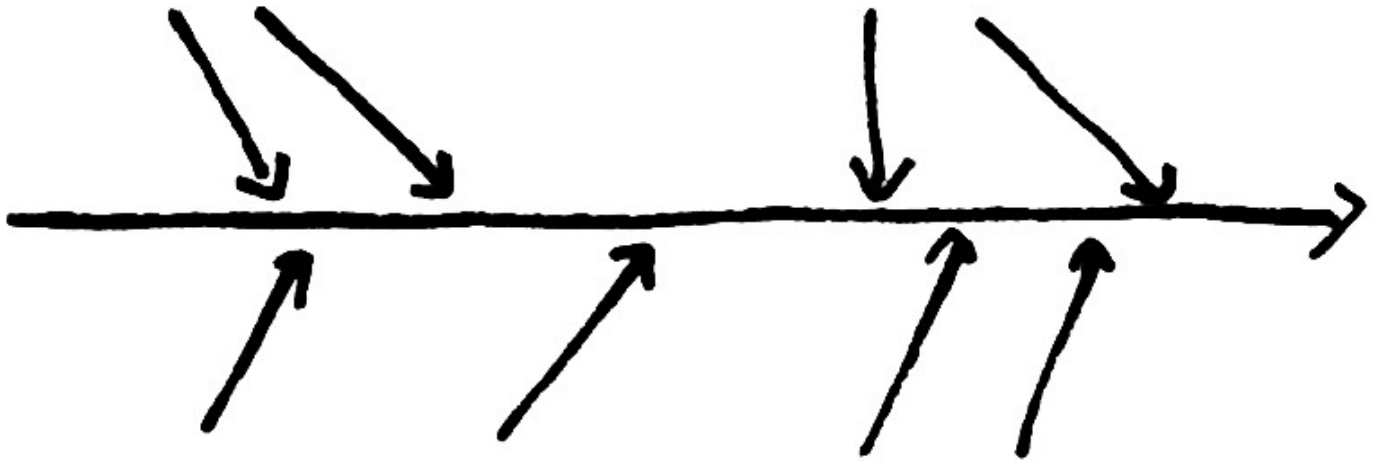
Theoretical Limit!

. With Cassandra
1.2.8 & 2.0.0,
Java driver 1.0.3,
2 writes / row
mean latency 100ms

1 in 250
rows found
corrupt!

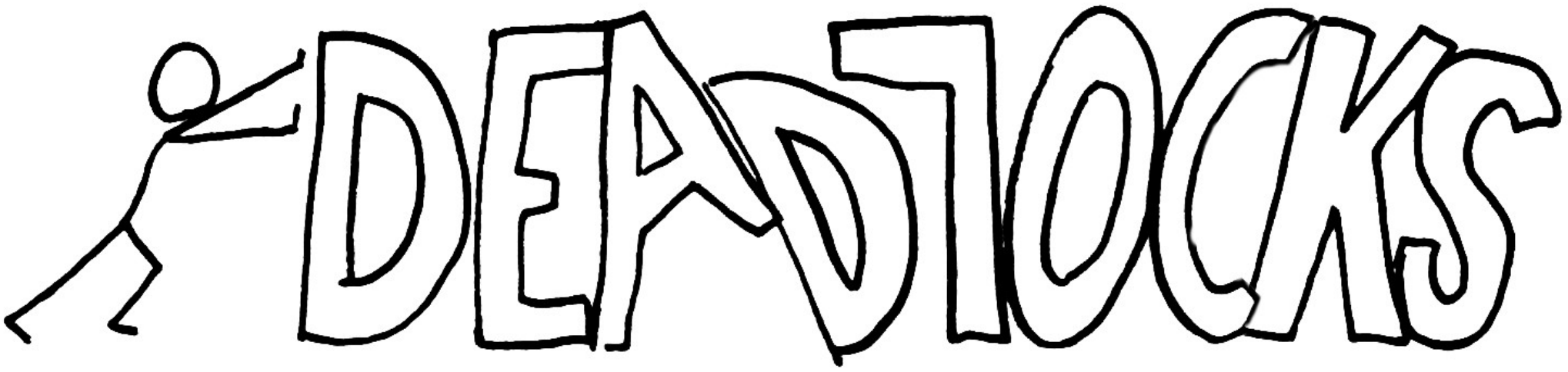

USE
lightweight
TRANSACTIONS

"Linearizable Consistency"



- DataStax, "Lightweight transactions in Cassandra 2.0"

Java driver does not support Paxos transactions.


DEADLOCKS

Paxos txns lock up the cluster permanently.

-clear system.paxos

```
$ git checkout paxos-fixed-hopefully
```

Paxos transactions are not linearizable.

#6012 #6013

Can accept two values for same paxos round.

Failed proposals can still succeed in later rounds.

Cassandra Transactions

5% acknowledged

4% false positives

9% false negatives

"Paxos made without stress tests."

Open source has had the reputation of producing good imitations, but not innovation. Perhaps Cassandra's origins as a hybrid of Dynamo and Bigtable did not disprove this, but Apache Cassandra's development of lightweight transactions and CQL are true industry firsts.

Cassandra strategies

- Use as a strict AP store
- No isolation/atomicity guarantees for writes
- Avoid transactions

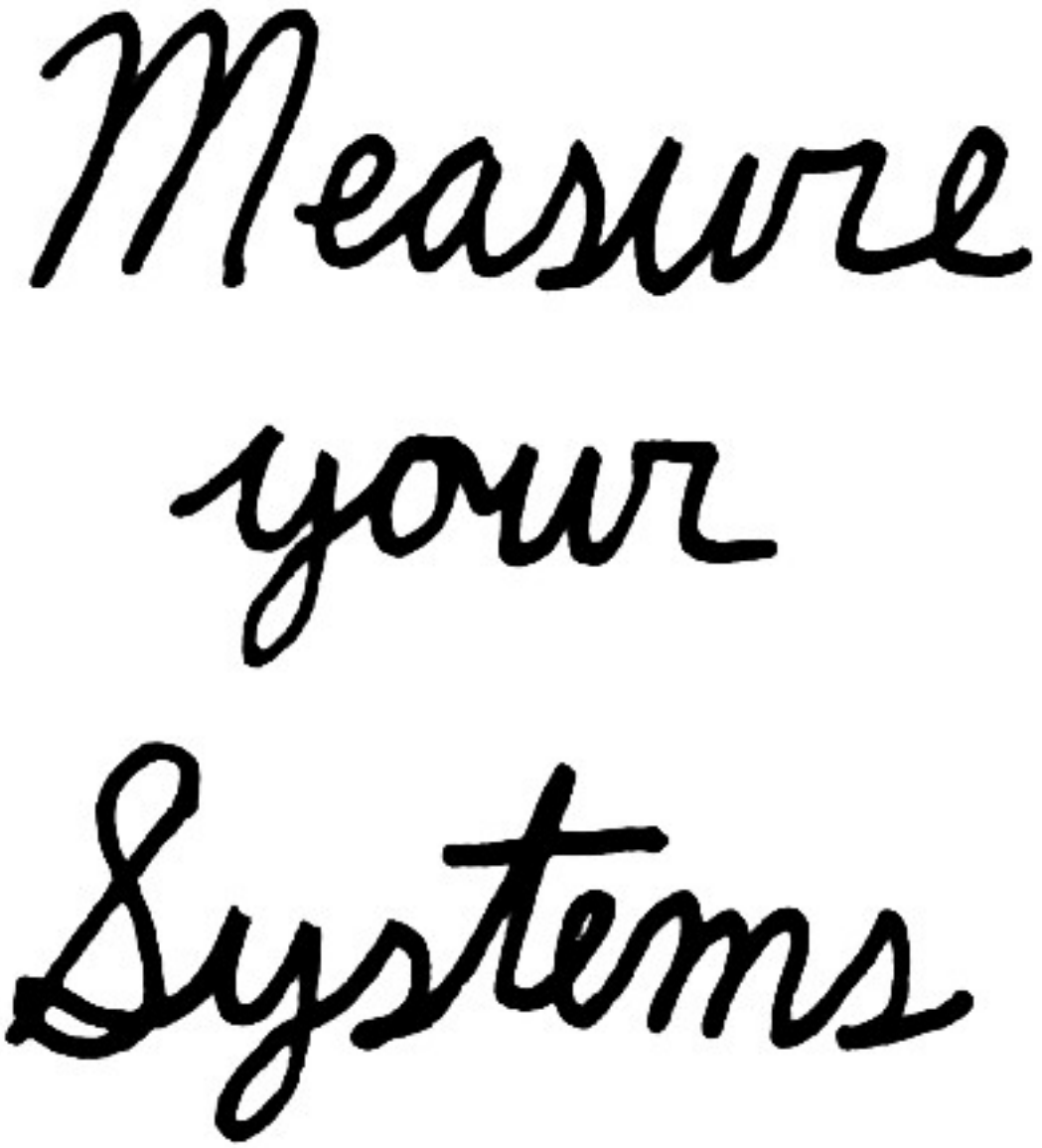
to

ReCAP

Jepsen only tells you about one system

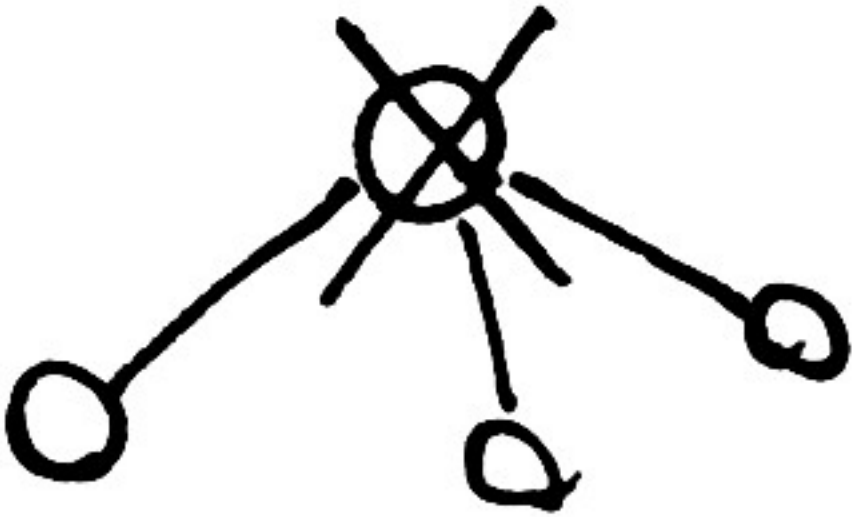
Clients are
a part of the
distributed system!

- Clients
- Servers
- Network
- Semantics


Measure
your
Systems

ASK:

"What if we lose this node?"

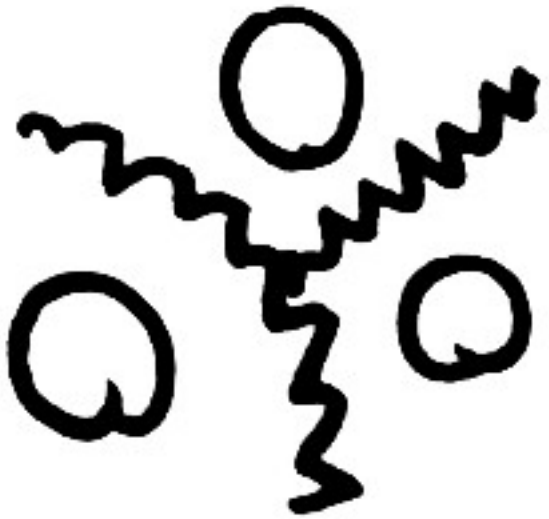


"What if these nodes paure for 3 minutes?"

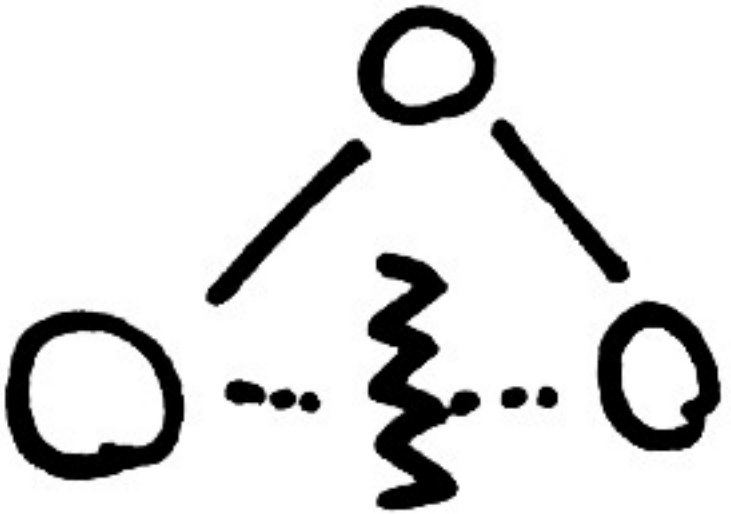
"How do we rely on clocks?"

"What if the network delivers this message 5 minutes late?"

"What if there is no majority?"



"What about partial visibility?"



"What safety and liveness guarantees does this system really need?"

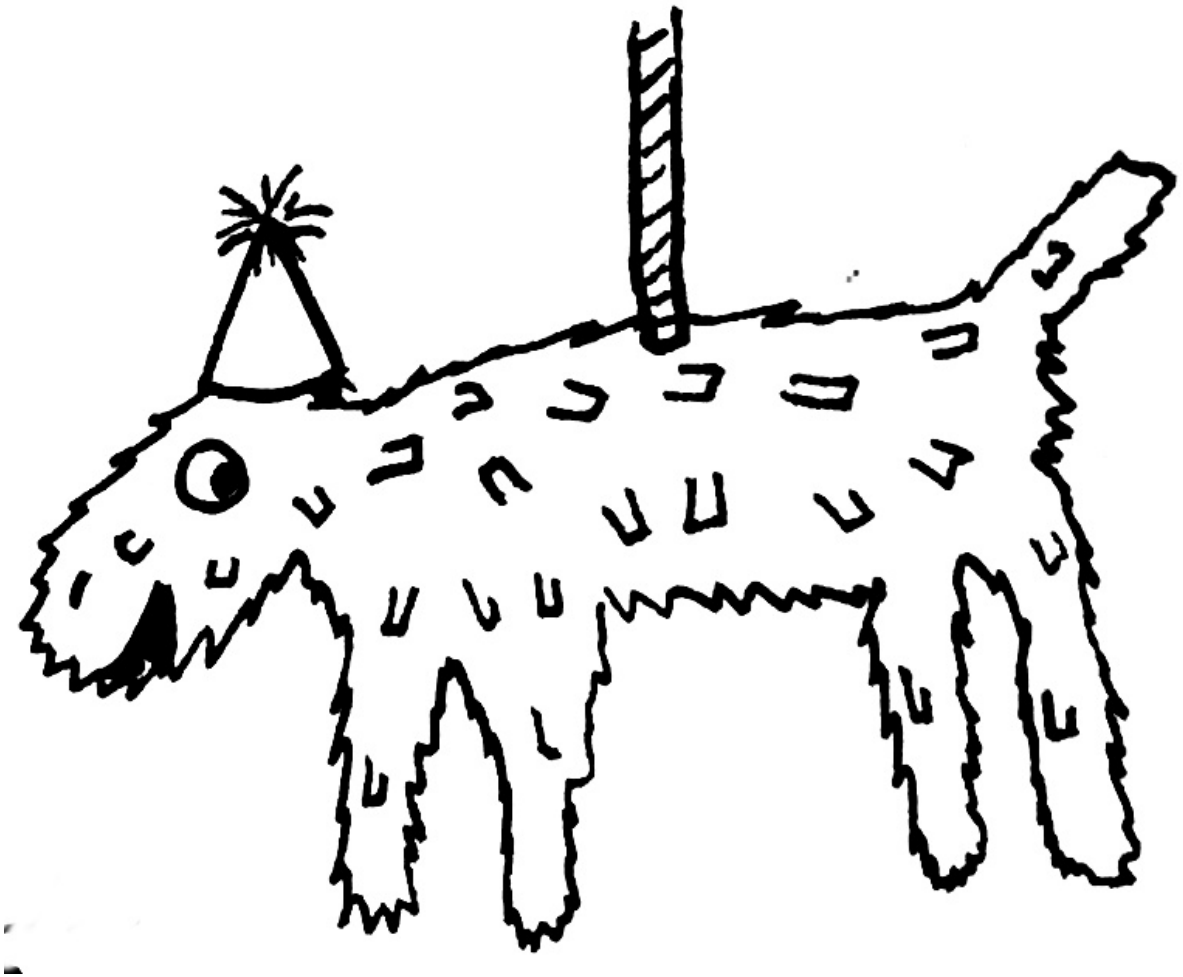
Data loss or inconsistency can be OK!

Balance

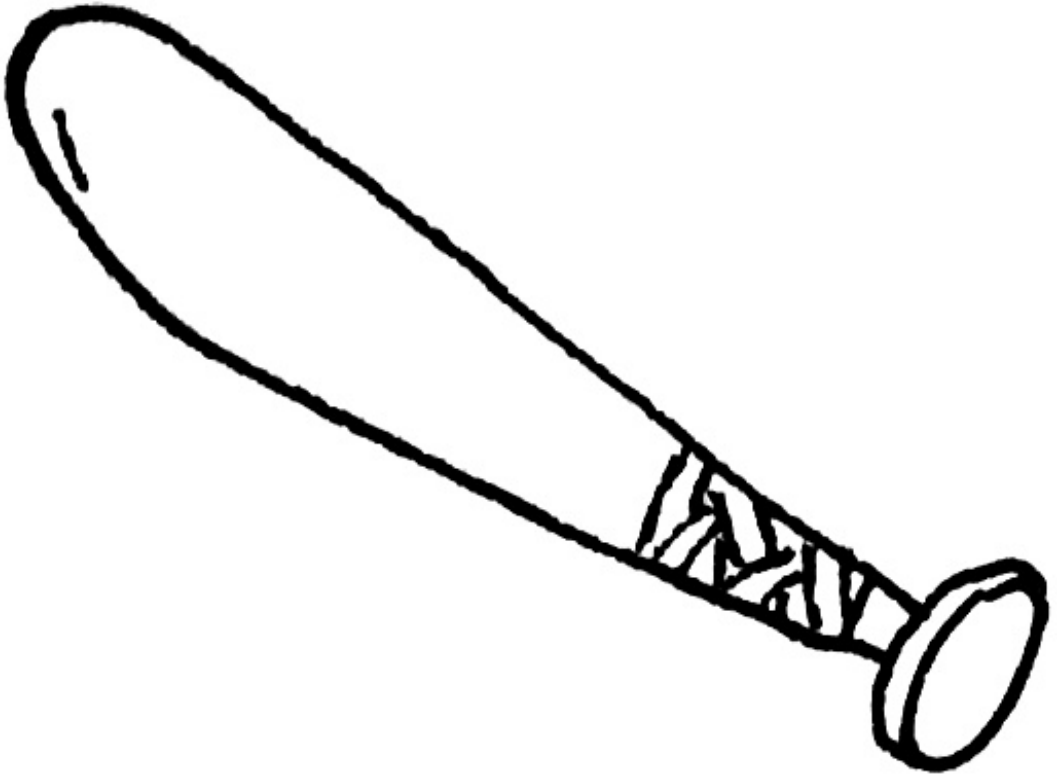
Complexity

— with —

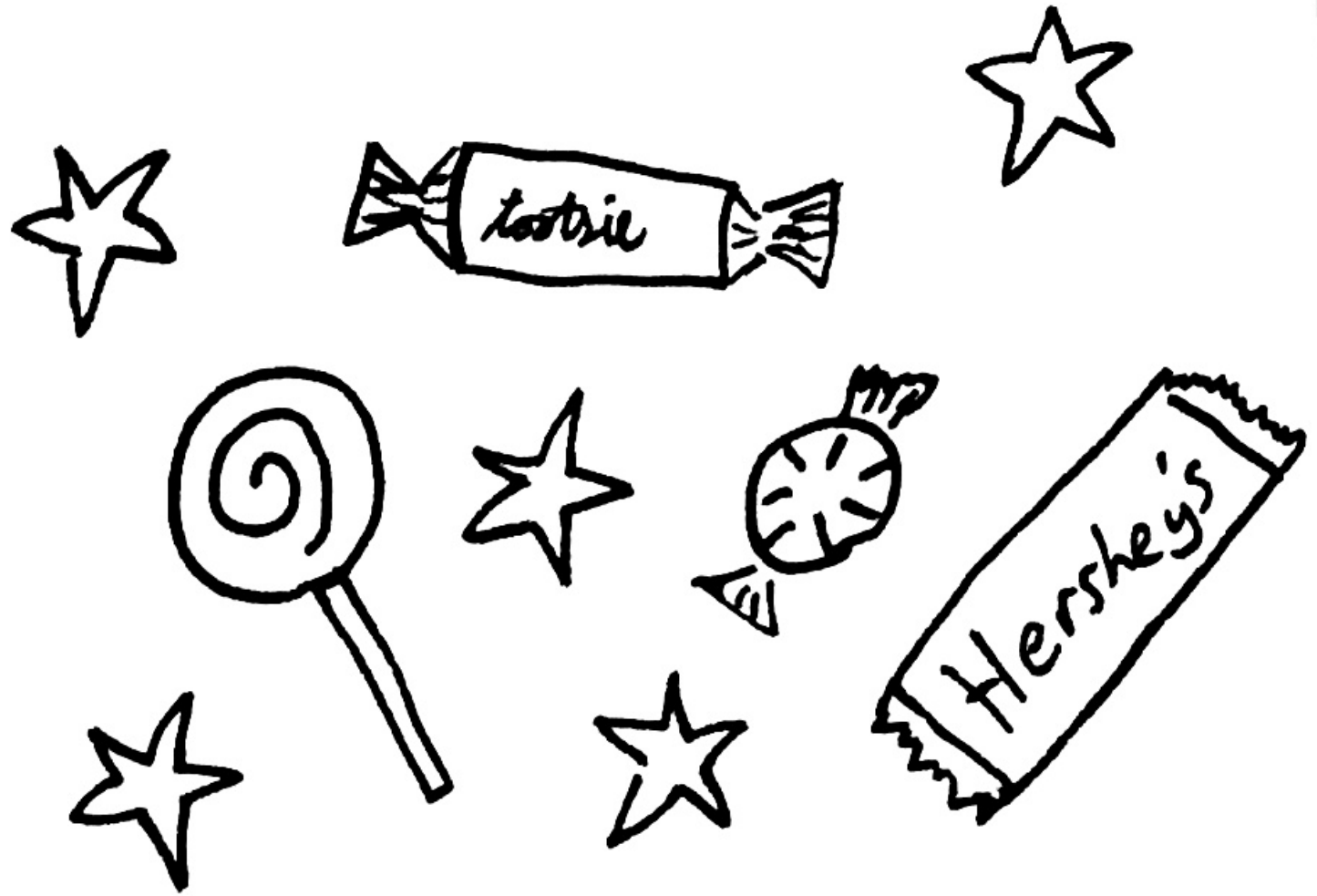
Correctness



Write a
test program!



Cause
Failures



SEE WHAT HAPPENS!

Thank You!

Jonathan Ellis

Rick Branson

Eric Lindvall

leftside

Patrick McFadin

Aleksey Yeschenko

Seth Proctor

Duke Adamonis

Neha Narkhede

Jun Rao

Kelly Sommers

Drew Blas

Michael Klishin

Camille Fournier

Jay Kreps

Joeseph Blomstedt

Peter Bailis

Thanks!

github.com/aphyr/jepsen