High Performance Transaction Systems

2022

JEPSEN 15

UNSAFE AT ANY SPEED

Kyle Kingsbury

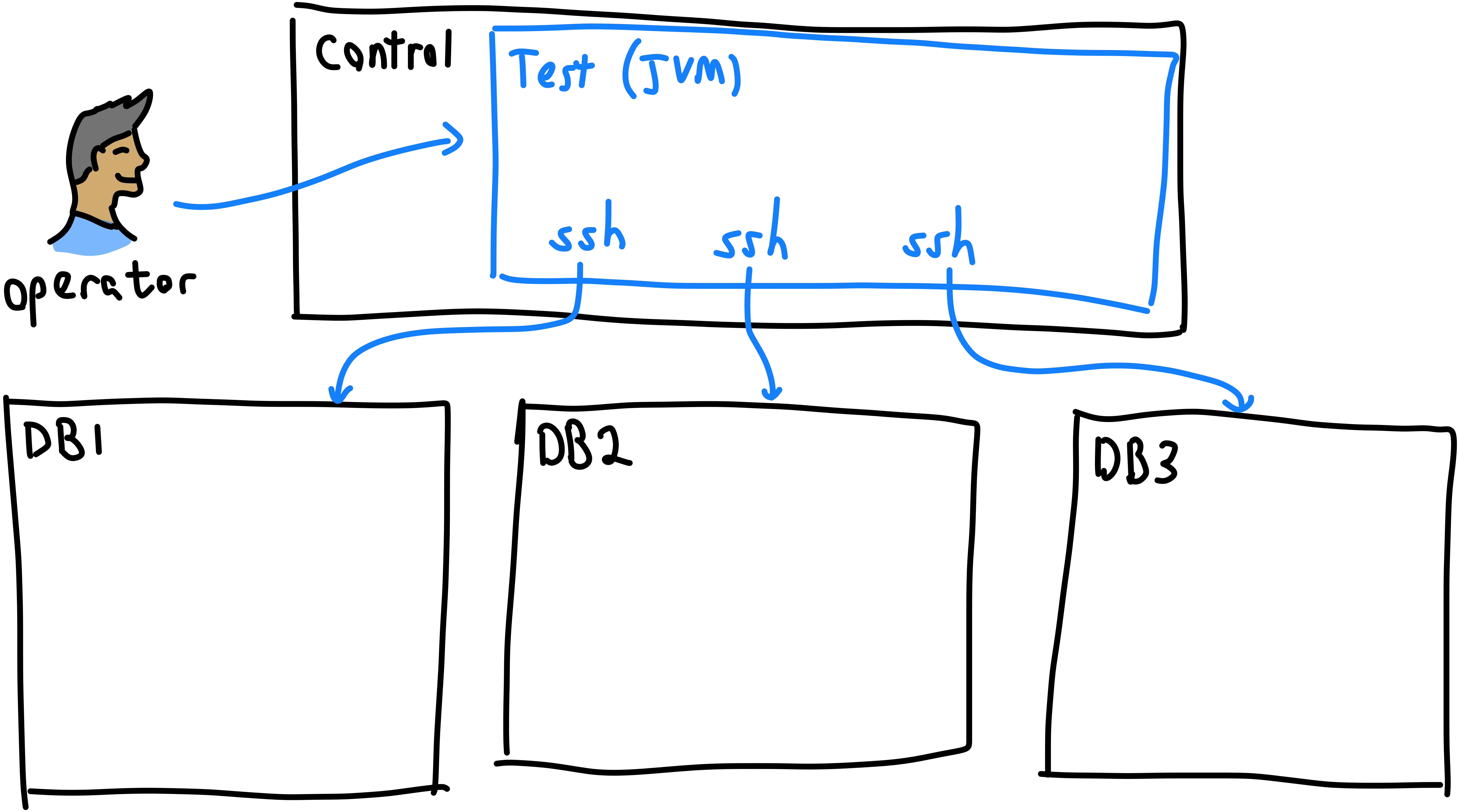
aphyr@jepsen.io

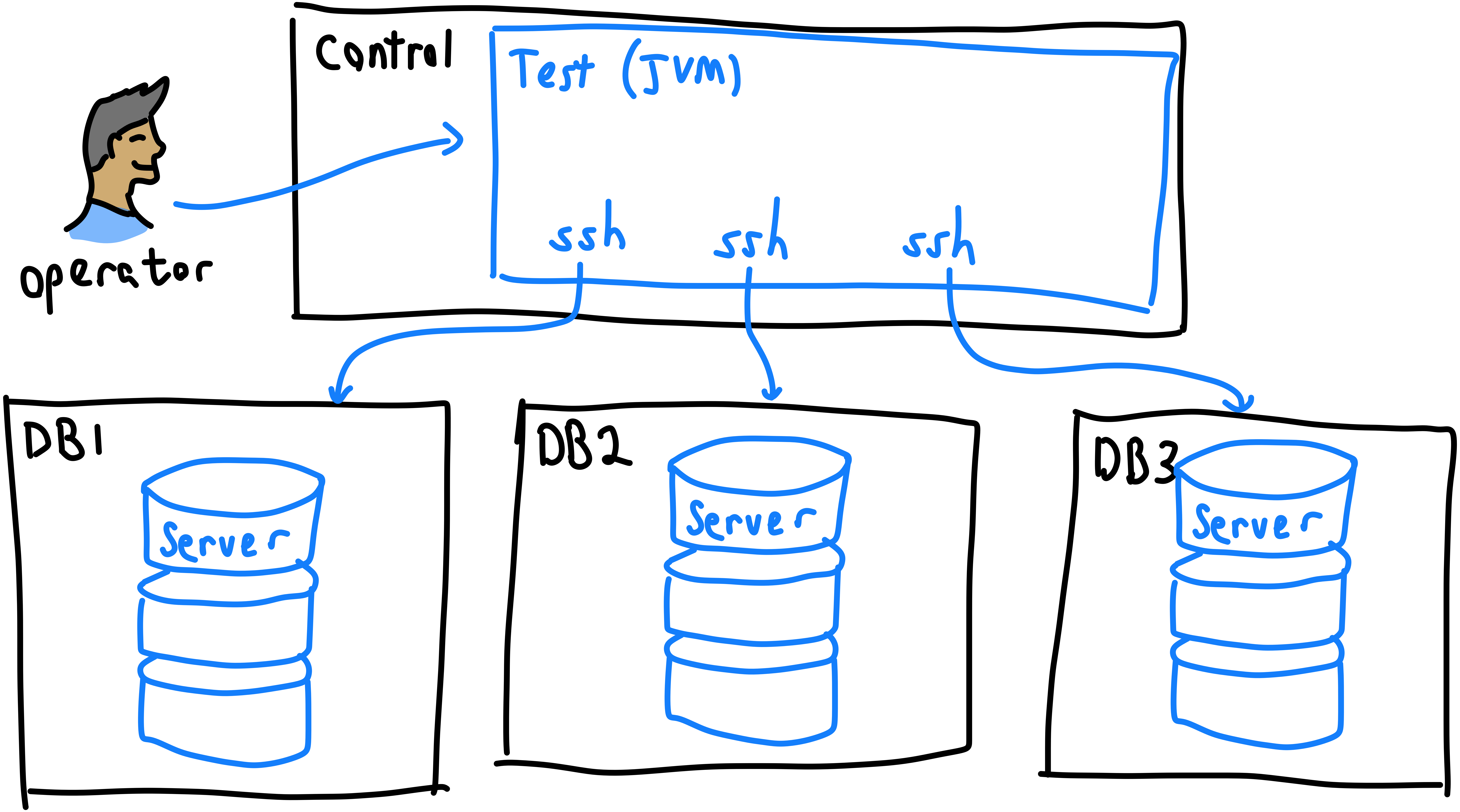
He/him

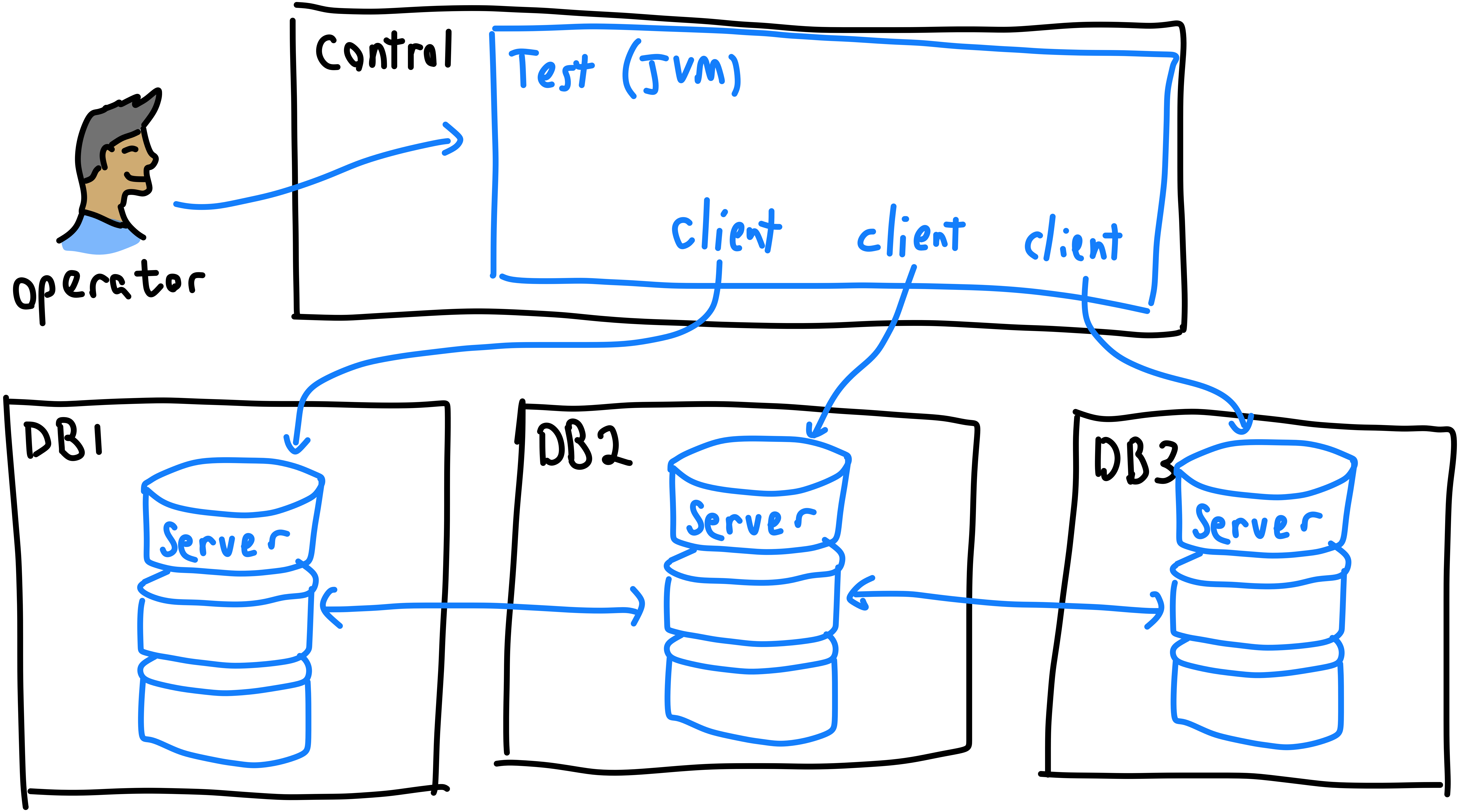
What Do Systems Actually Do?

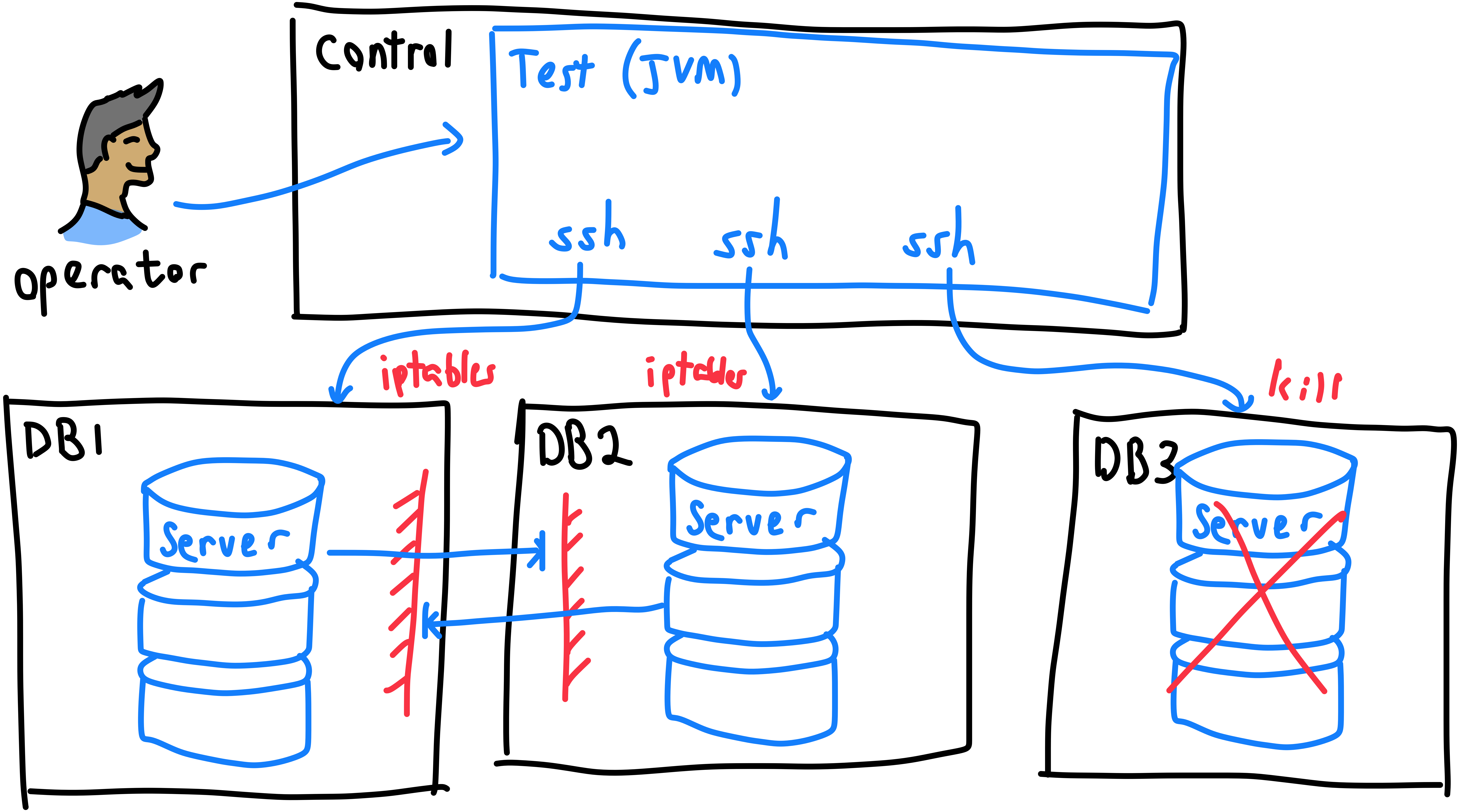
Jepsen

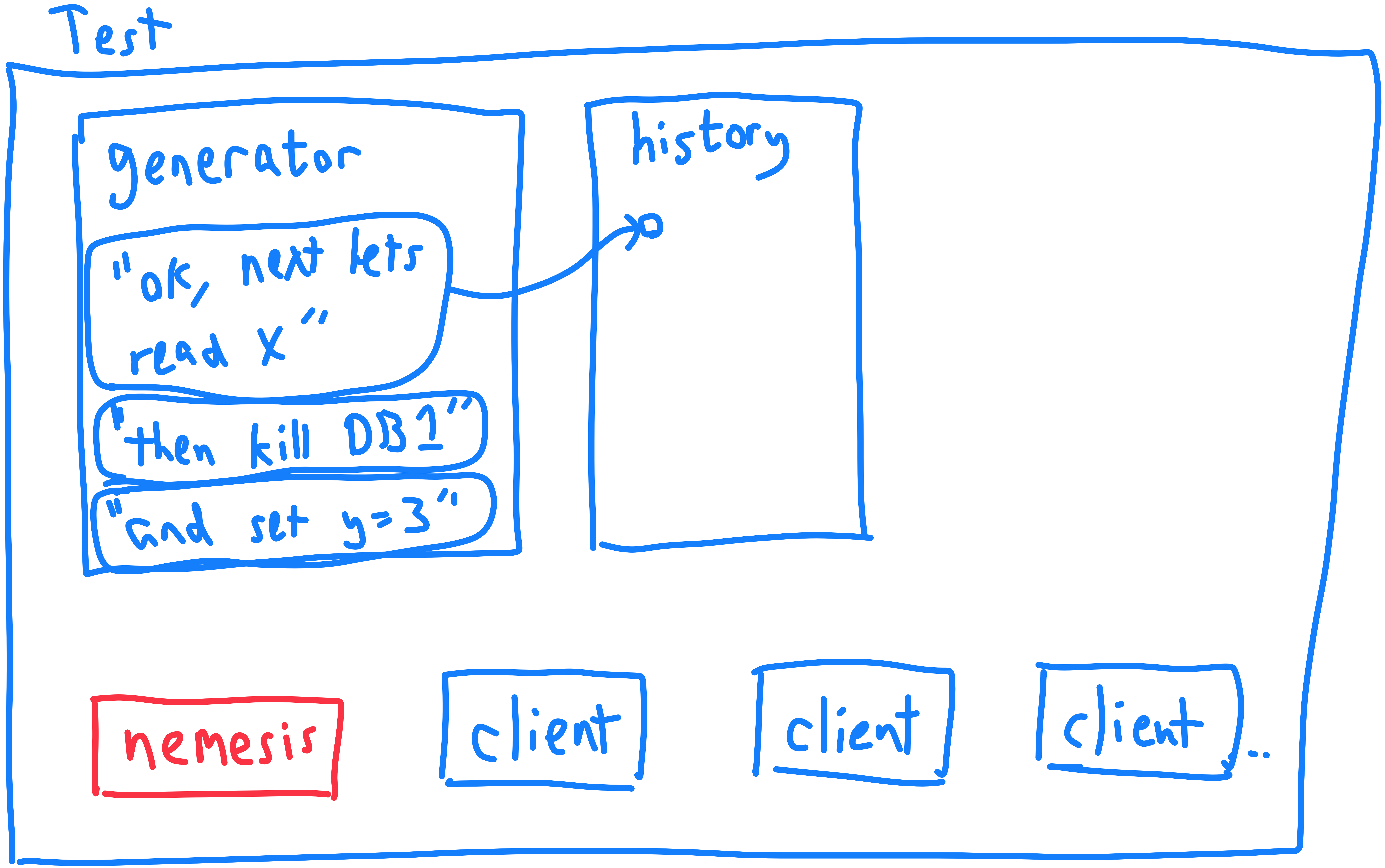
https://github.com/jepsen-io/jepsen

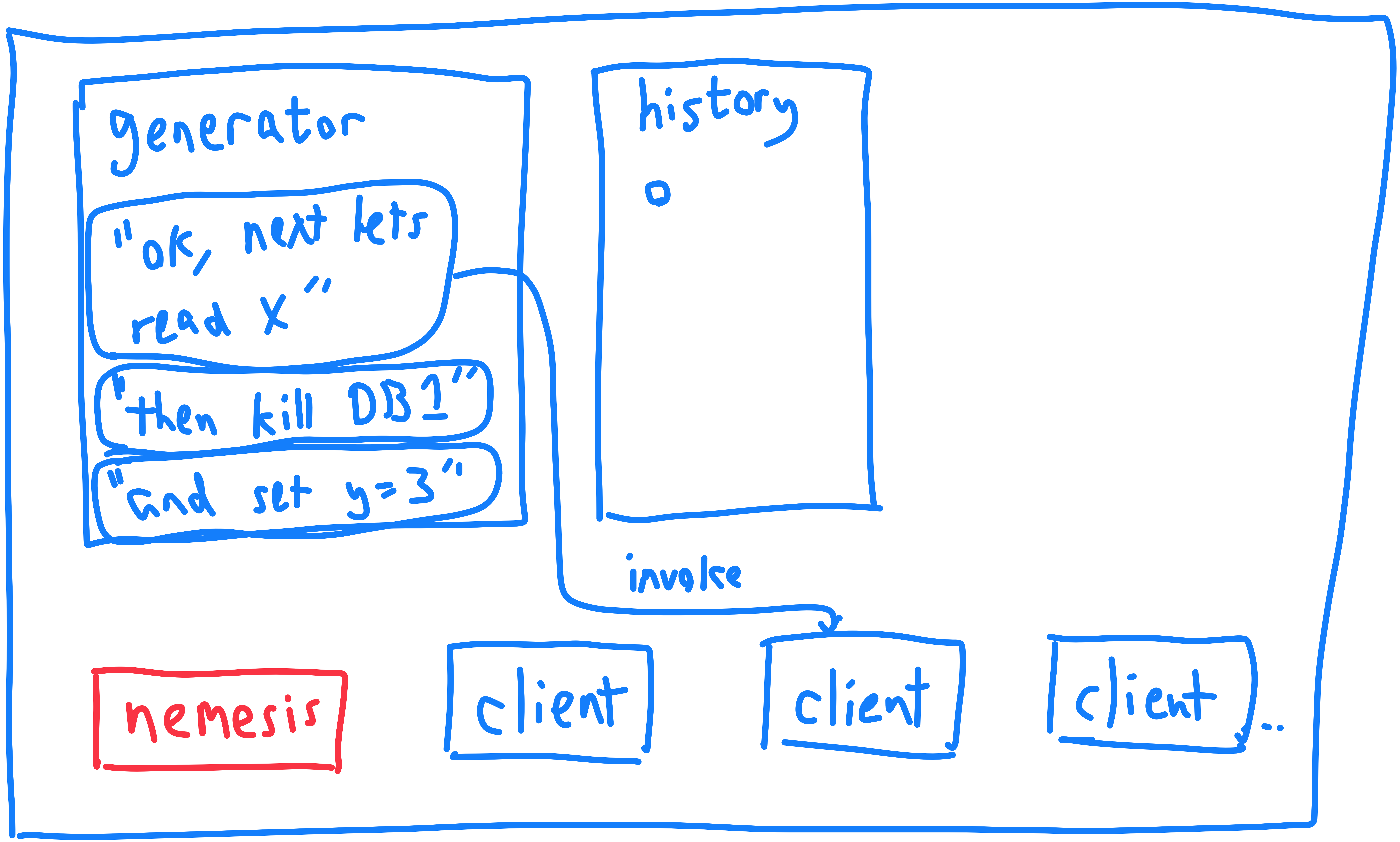

operator
Control
Test (JVM)
ssh
ssh
ssh
DB1
DB2
DB3

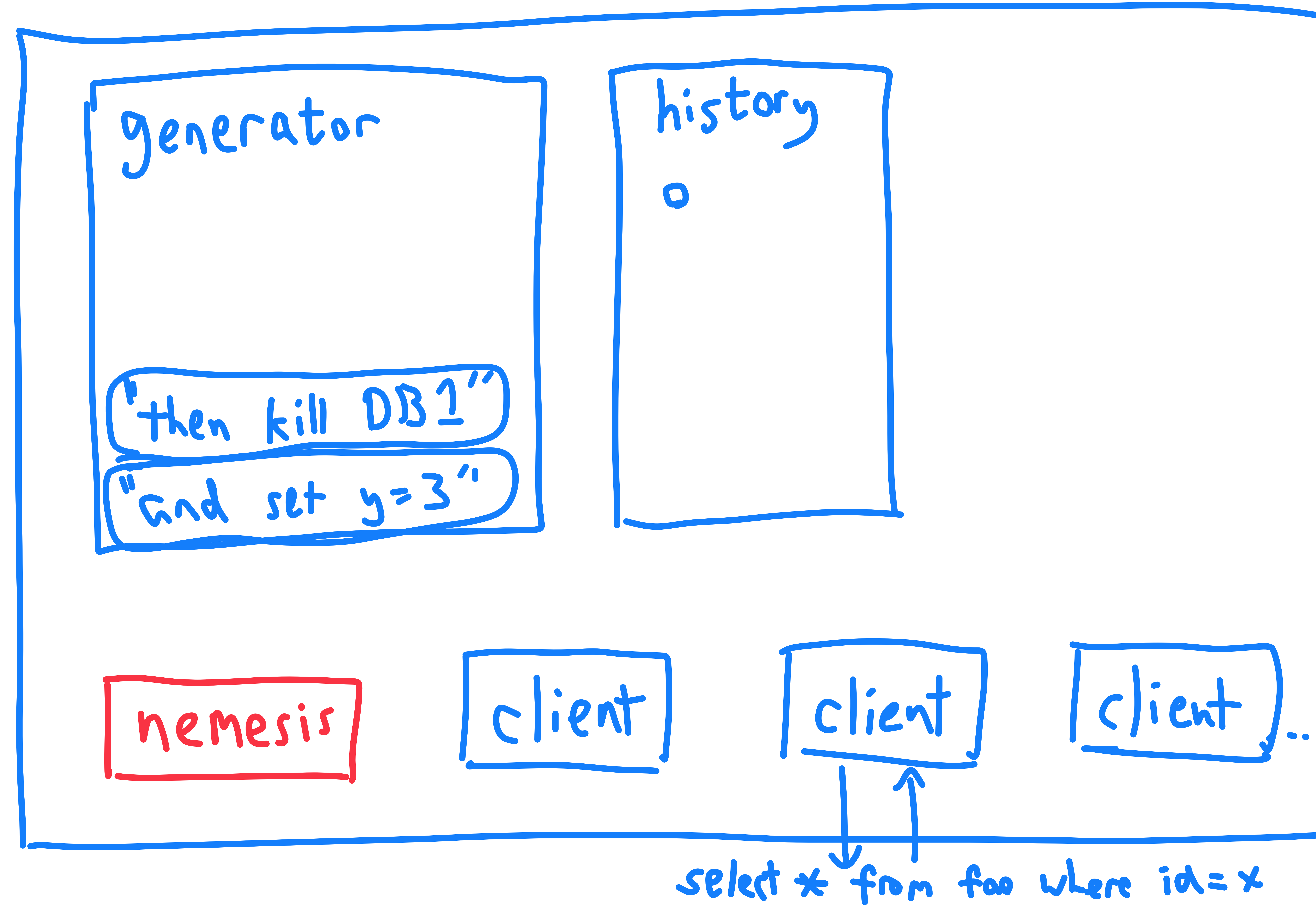

operator
Control
Test (JVM)
ssh
ssh
ssh
DB1
Server
DB2
Server
DB3
Server

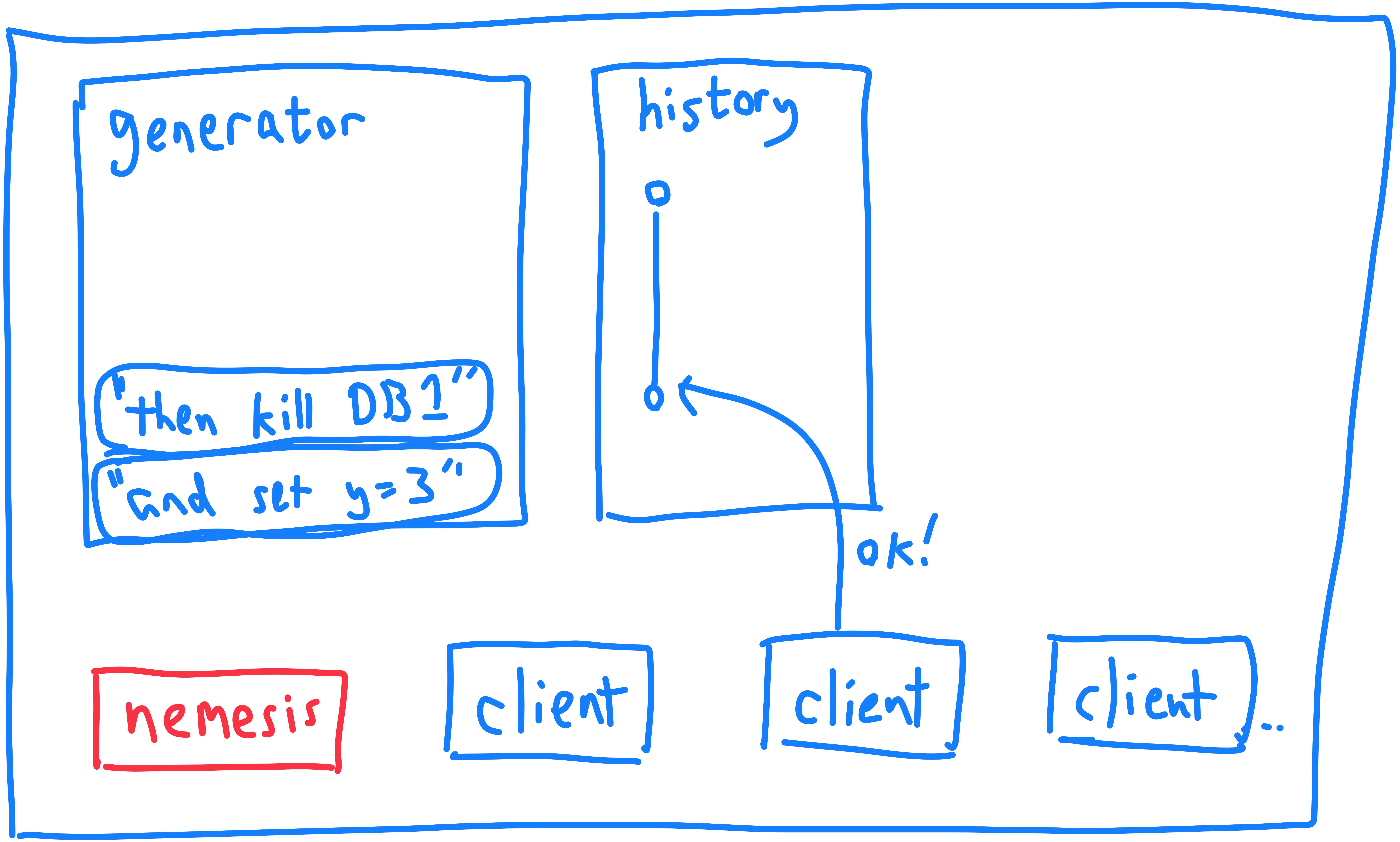

operator
Control
Test (JVM)
client
client
client
DB1
Server
DB2
Server
DB3
Server

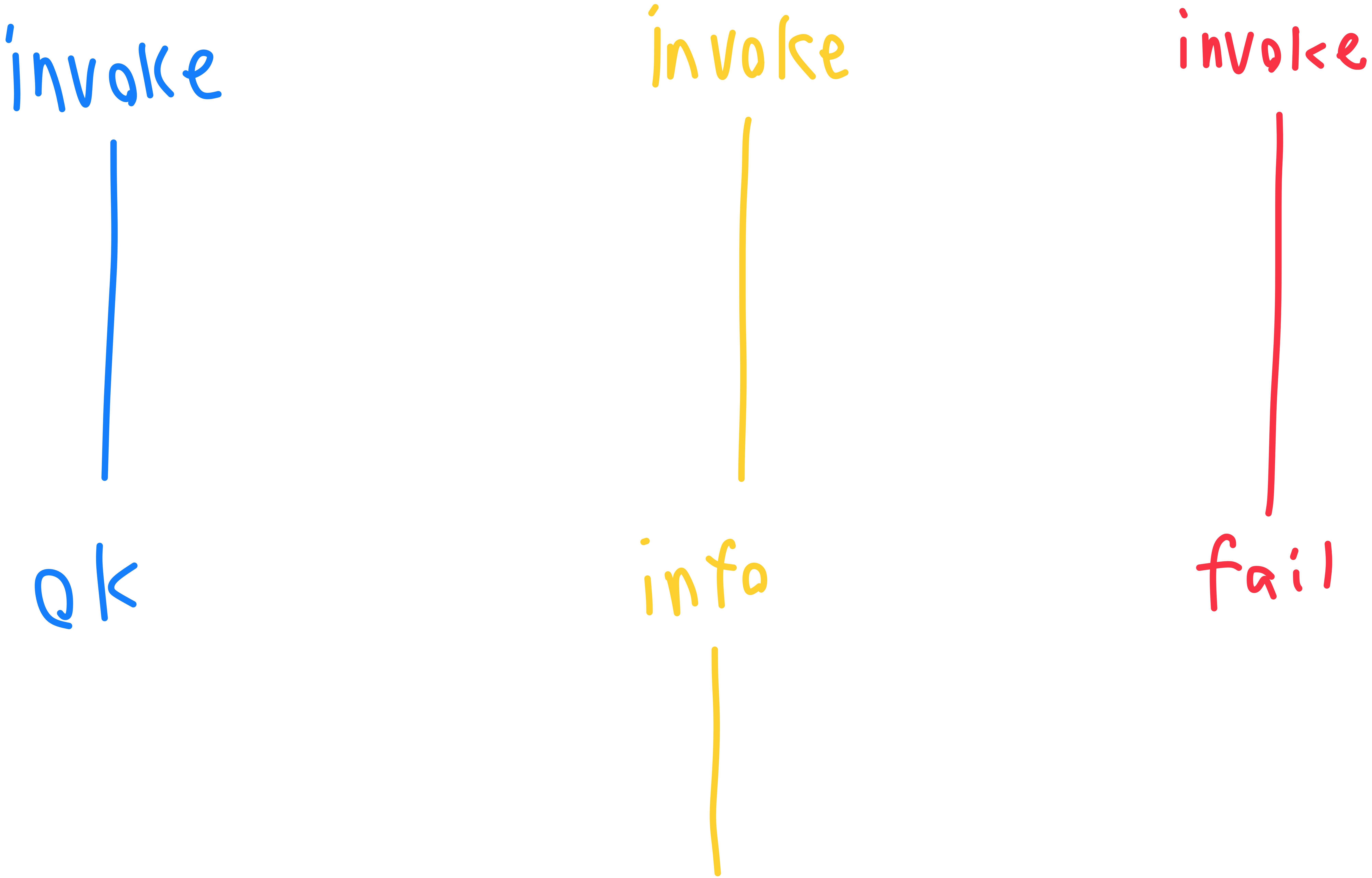

operator
Control
Test (JVM)
ssh
ssh
ssh
iptables
iptables
kill
DB1
Server
DB2
Server
DB3
Server

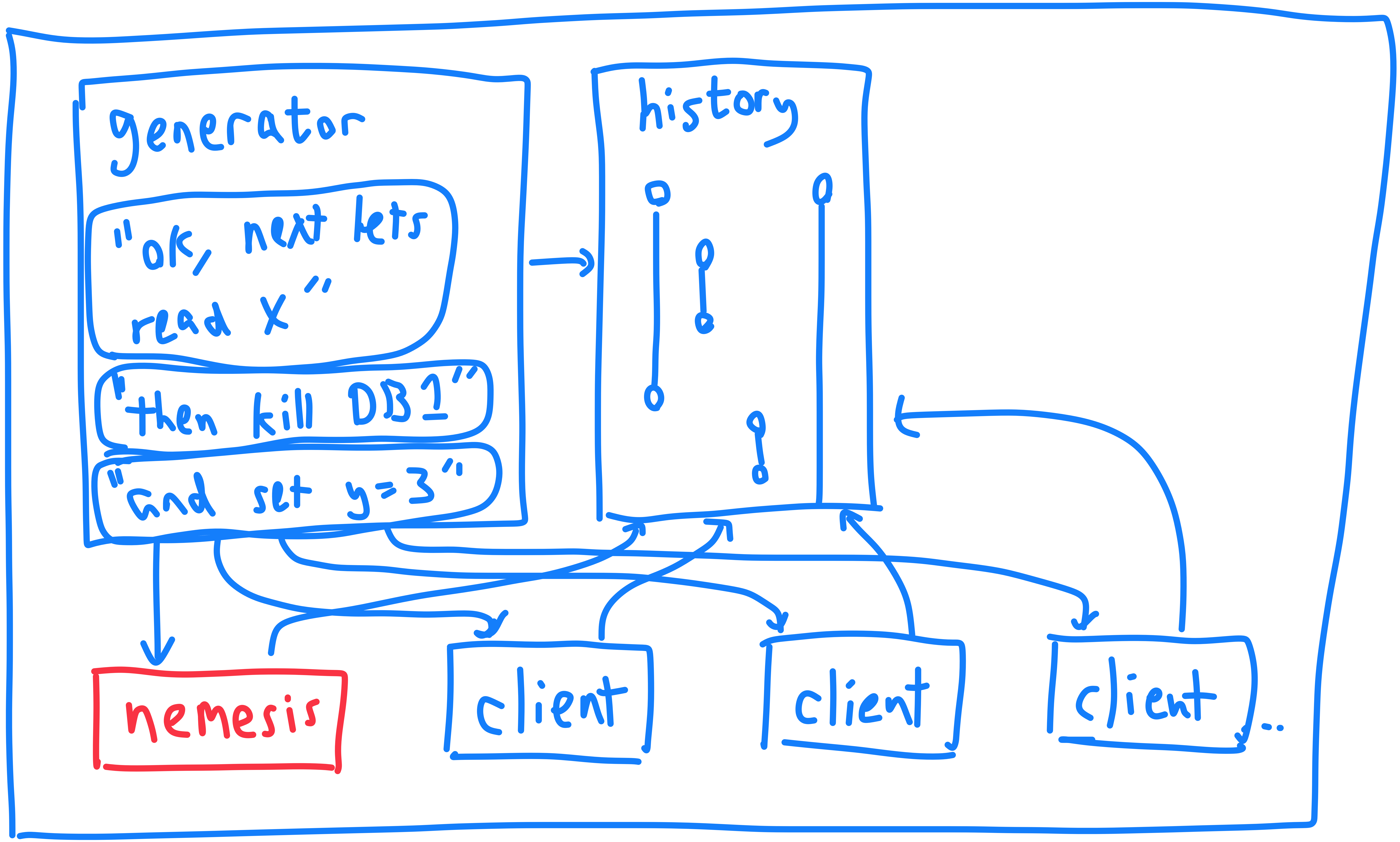

Test
generator
"ok, next lets read x"
"then kill DB 1"
"and set y=3"
history
nemesis
client
client
client
...

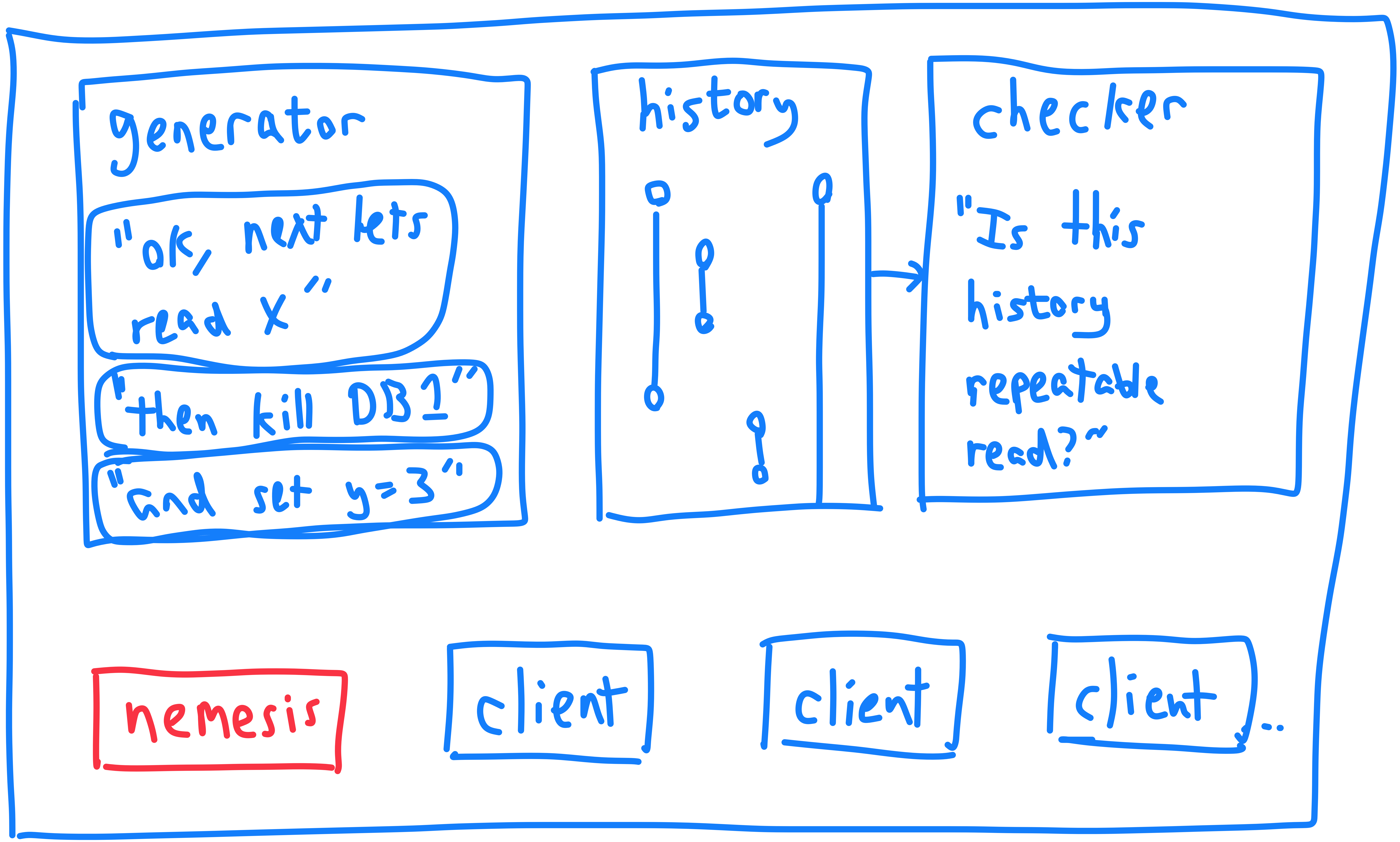

generator
"ok, next lets read x"
"then kill DB 1"
"and set y=3"
history
o
invoke
nemesis
client
client
client...

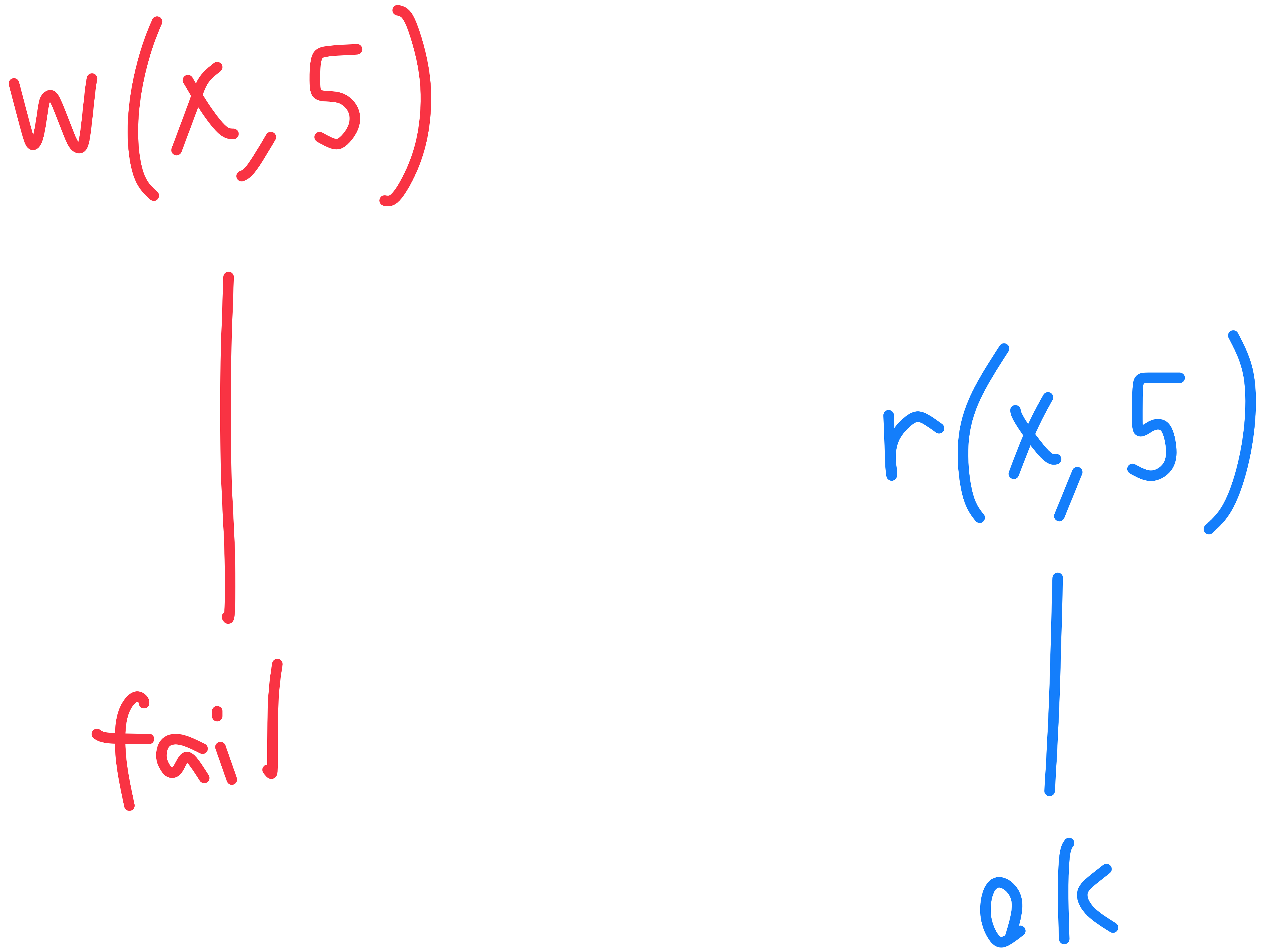

generator
"then kill DB 1"
"and set y=3"
history
o
nemesis
client
client
client
...
select * from foo where id=x


generator
"then kill DB 1"
"and set y=3"
history
ok!
nemesis
client
client
client
...

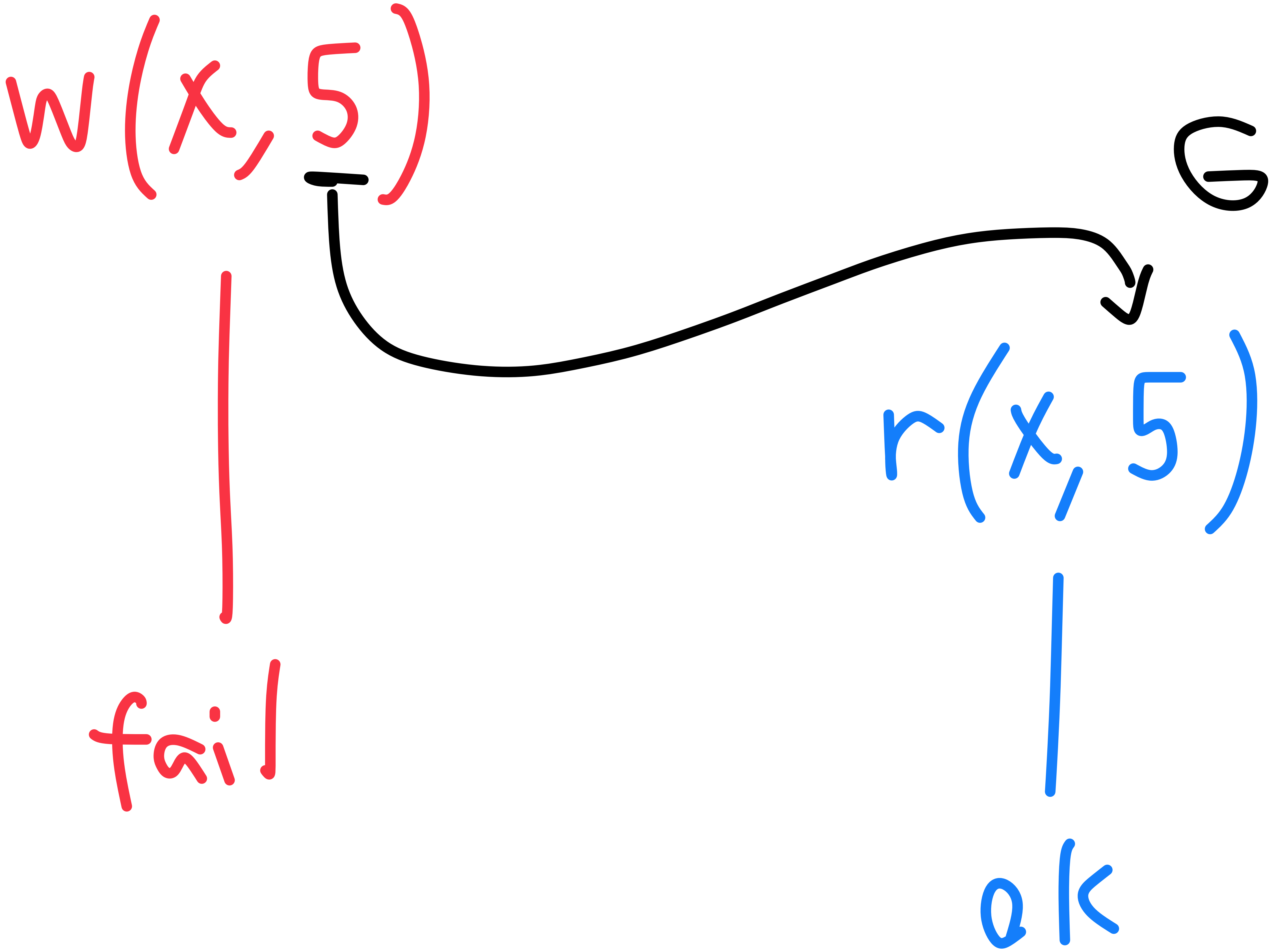

invoke
ok
invoke
info
invoke
fail


generator
"ok, next lets read x"
"then kill DB 1"
"and set y=3"
history
nemesis
client
client
client
...


generator
"ok, next lets read X"
"then kill DB 1"
"and set y=3"
history
checker
"Is this history repeatable read?"
nemesis
client
client
client...


w(x, 5)
fail
r(x, 5)
ok

G1a: aborted read!



√RADIX

1.0-beta35.1 . . . 1.0.2

- Distributed ledger for decentralized finance
- Permissionless (Proof of Stake)
- Byzantine fault tolerant

Think "Ethereum"

- Accounts w/ tokens
  - Transfer
  - Read balance
  - Read history of txns on account
- Planned: smart contracts in homegrown language, "atomic composeability"

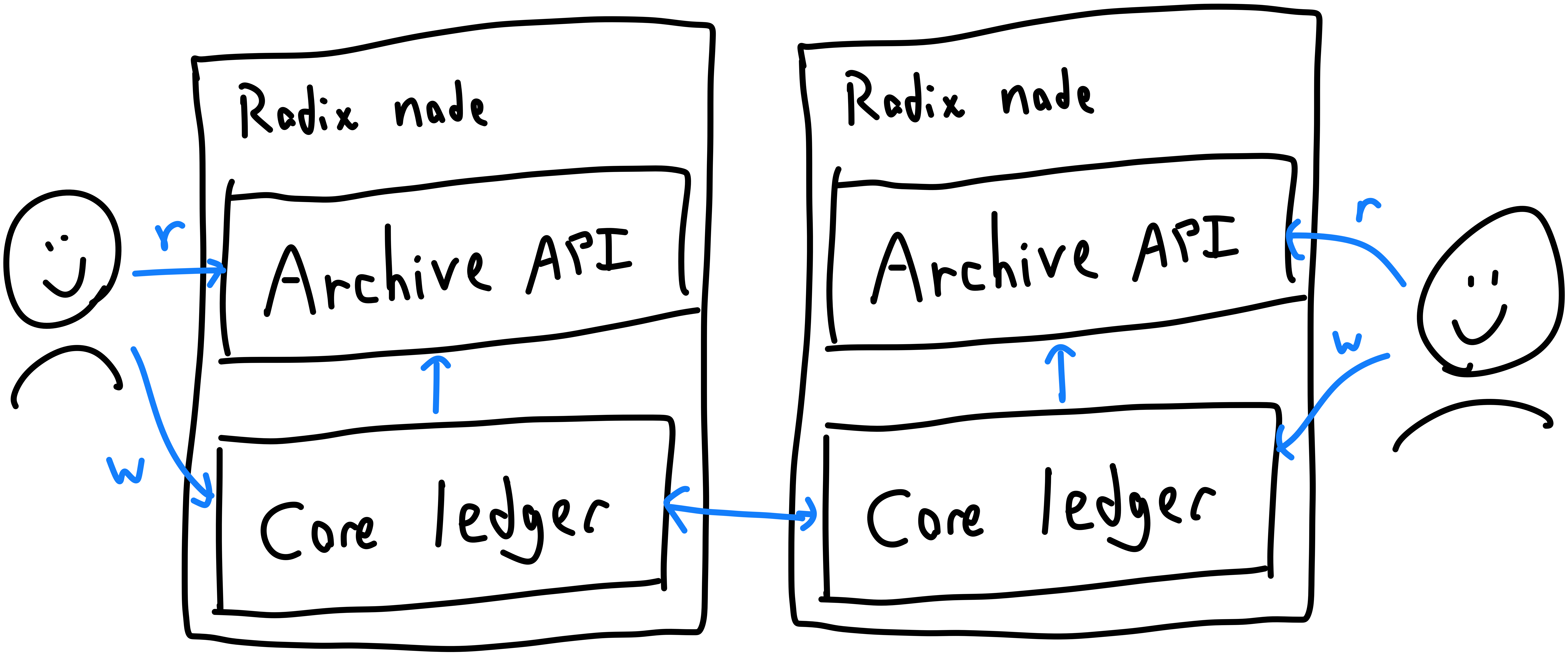
"Blockchain"/"DLT" → "Database"
(replicated log → state machine)

"Taken"/"Coin" → "Entry in DB"

"Transaction" → "Transaction"

"Smart contract" → "Stored procedure"

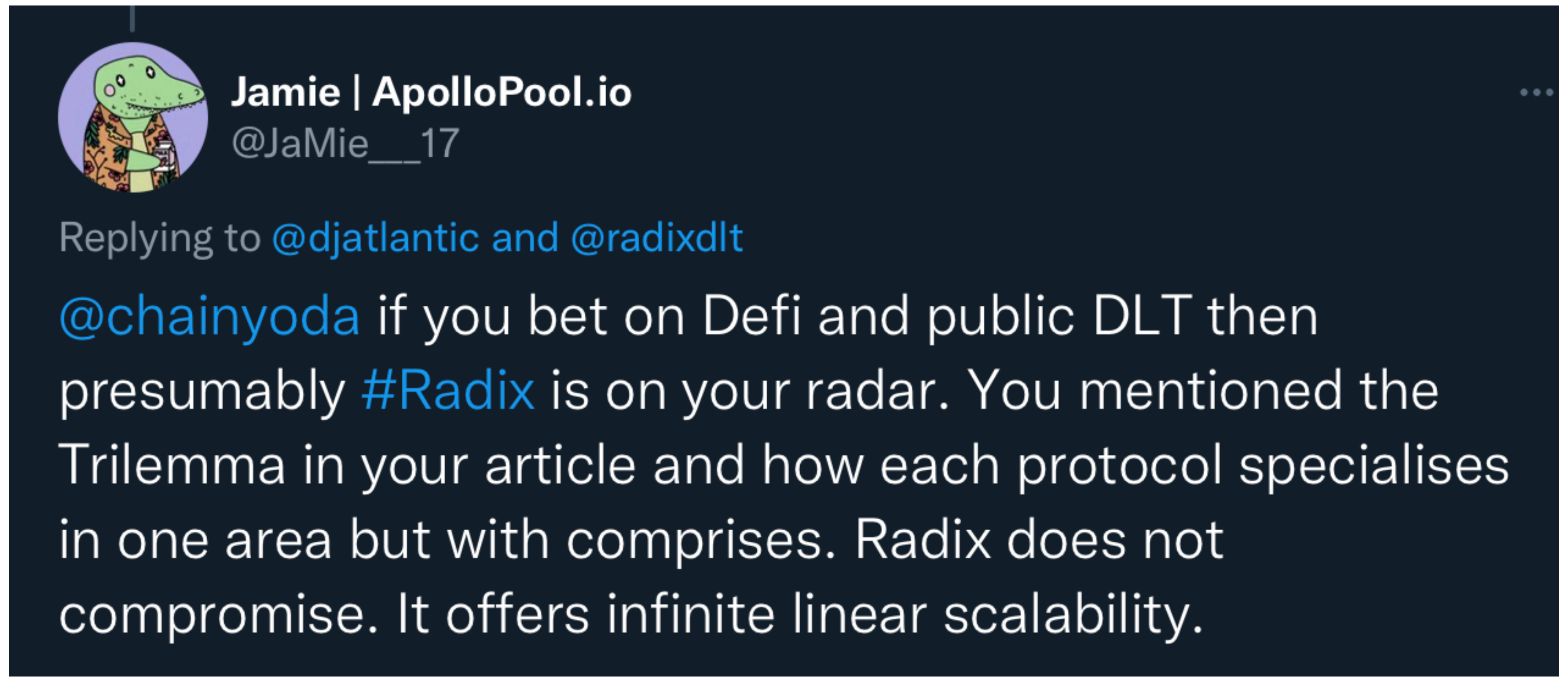
"Atomic comparscability" → "Serializability"


Radix node
Archive API
Core ledger
Radix node
Archive API
Core ledger
r
w
r
w

High
Performance
Transaction
Systems

"1000x more scalability than Ethereum"

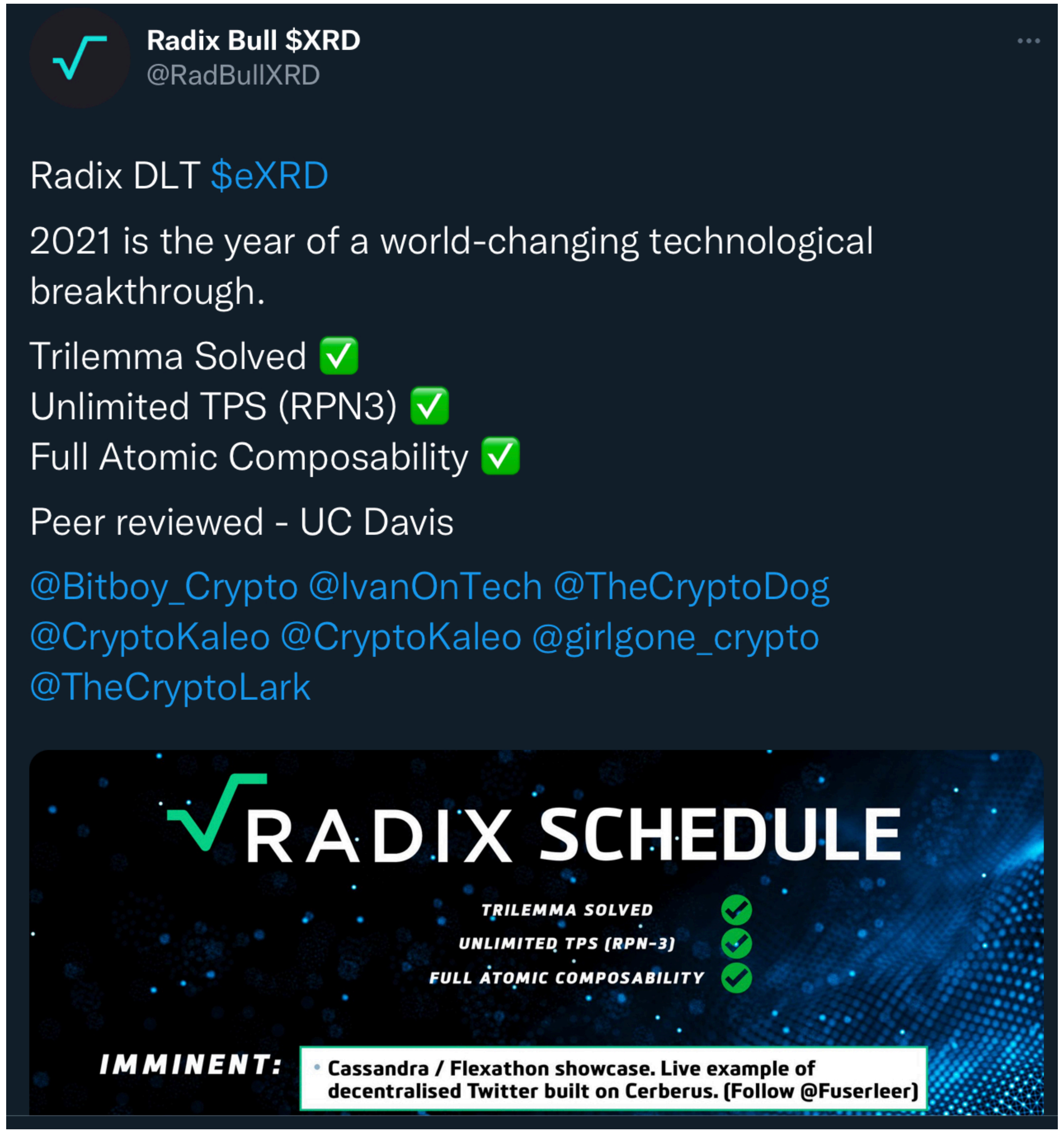
"Radix's last consensus algorithm 'Tempo' publicly achieved 1.4m TPS in 2018, the current world record... the new algorithm 'Cerberus' is theoretically infinitely scalable..."



Jamie | ApolloPool.io
@JaMie__17

Replying to @djatlantic and @radixdlt

@chainyoda if you bet on Defi and public DLT then presumably #Radix is on your radar. You mentioned the Trilemma in your article and how each protocol specialises in one area but with comprises. Radix does not compromise. It offers infinite linear scalability.



Radix Bull $XRD
@RadBullXRD

Radix DLT $eXRD

2021 is the year of a world-changing technological breakthrough.

Trilemma Solved ✅
Unlimited TPS (RPN3) ✅
Full Atomic Composability ✅

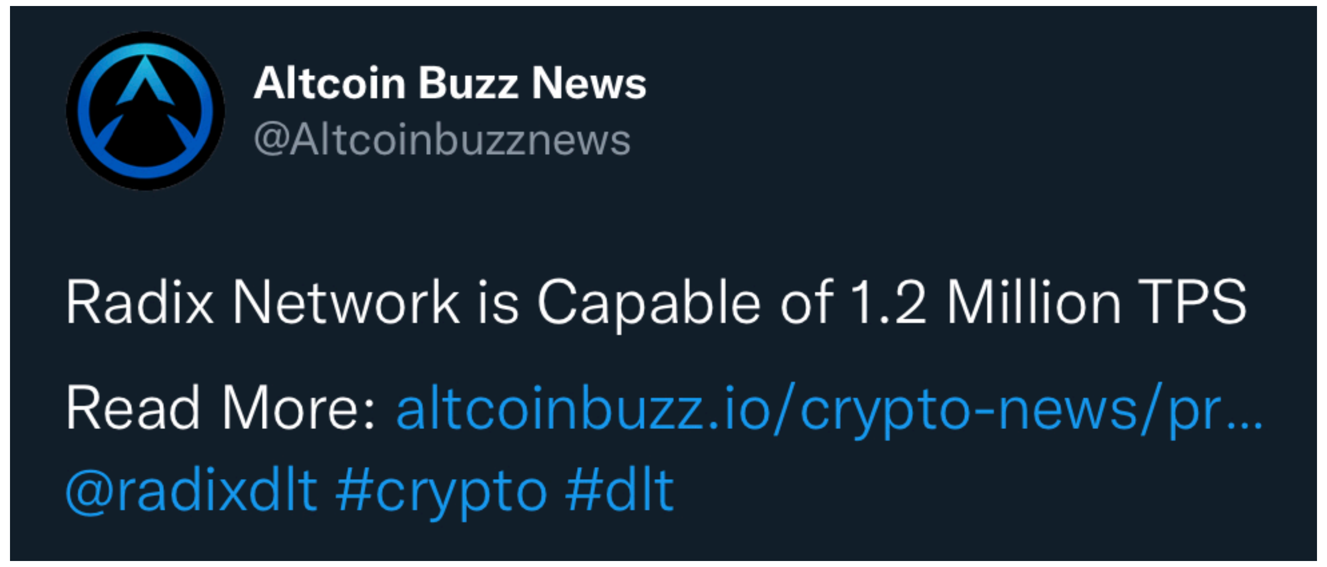
Peer reviewed - UC Davis

@Bitboy_Crypto @IvanOnTech @TheCryptoDog @CryptoKaleo @CryptoKaleo @girlgone_crypto @TheCryptoLark

RADIX SCHEDULE

TRILEMMA SOLVED ✓
UNLIMITED TPS (RPN-3) ✓
FULL ATOMIC COMPOSABILITY ✓

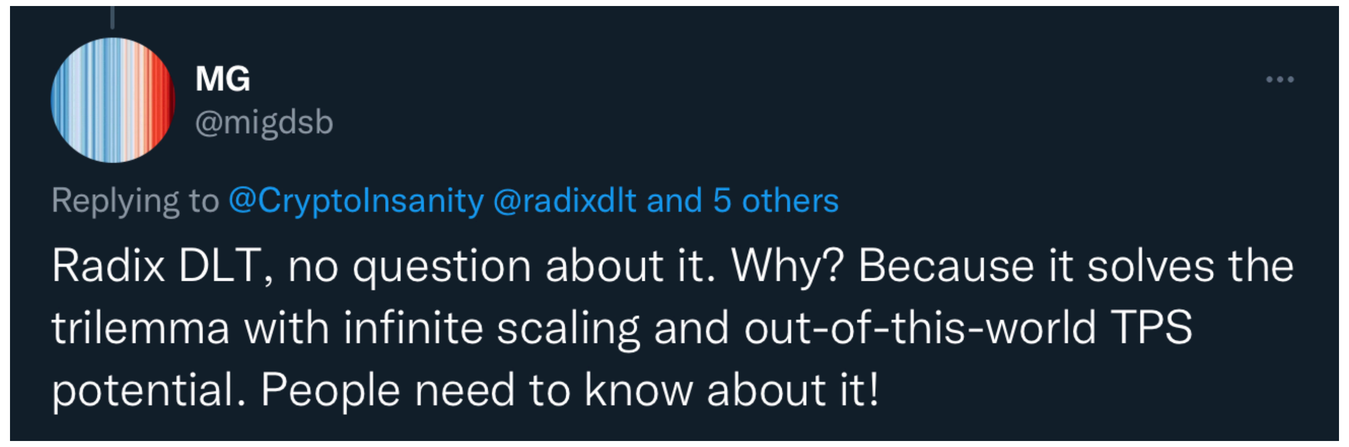
IMMINENT: • Cassandra / Flexathon showcase. Live example of decentralised Twitter built on Cerberus. (Follow @Fuserleer)



Altcoin Buzz News
@Altcoinbuzznews

Radix Network is Capable of 1.2 Million TPS

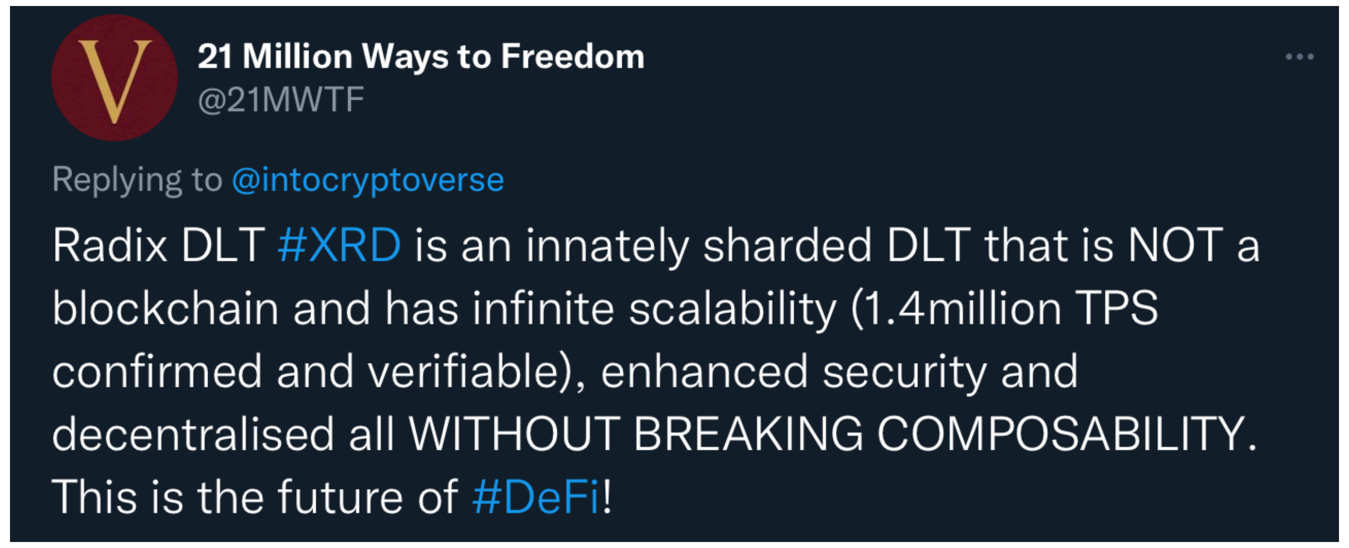
Read More: altcoinbuzz.io/crypto-news/pr...
@radixdlt #crypto #dlt



MG
@migdsb

Replying to @CryptoInsanity @radixdlt and 5 others

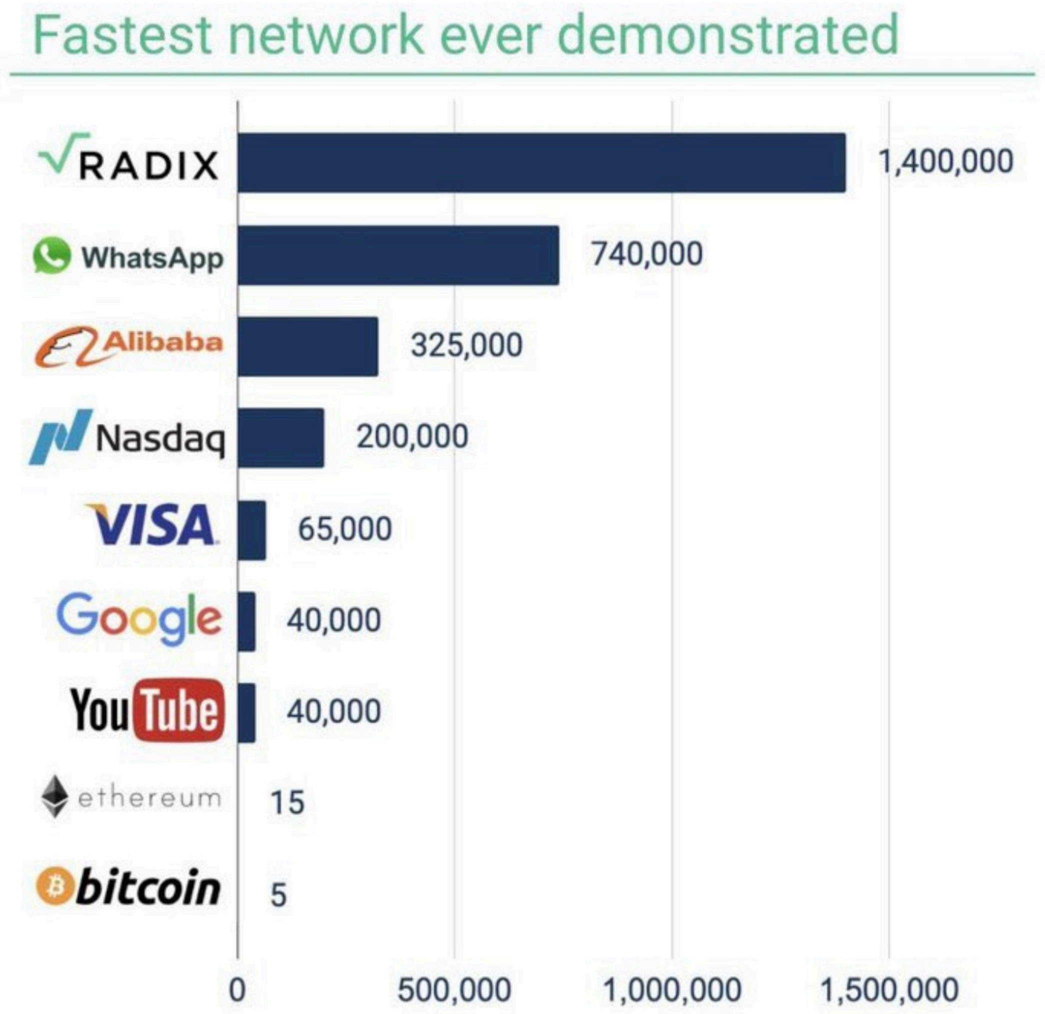
Radix DLT, no question about it. Why? Because it solves the trilemma with infinite scaling and out-of-this-world TPS potential. People need to know about it!



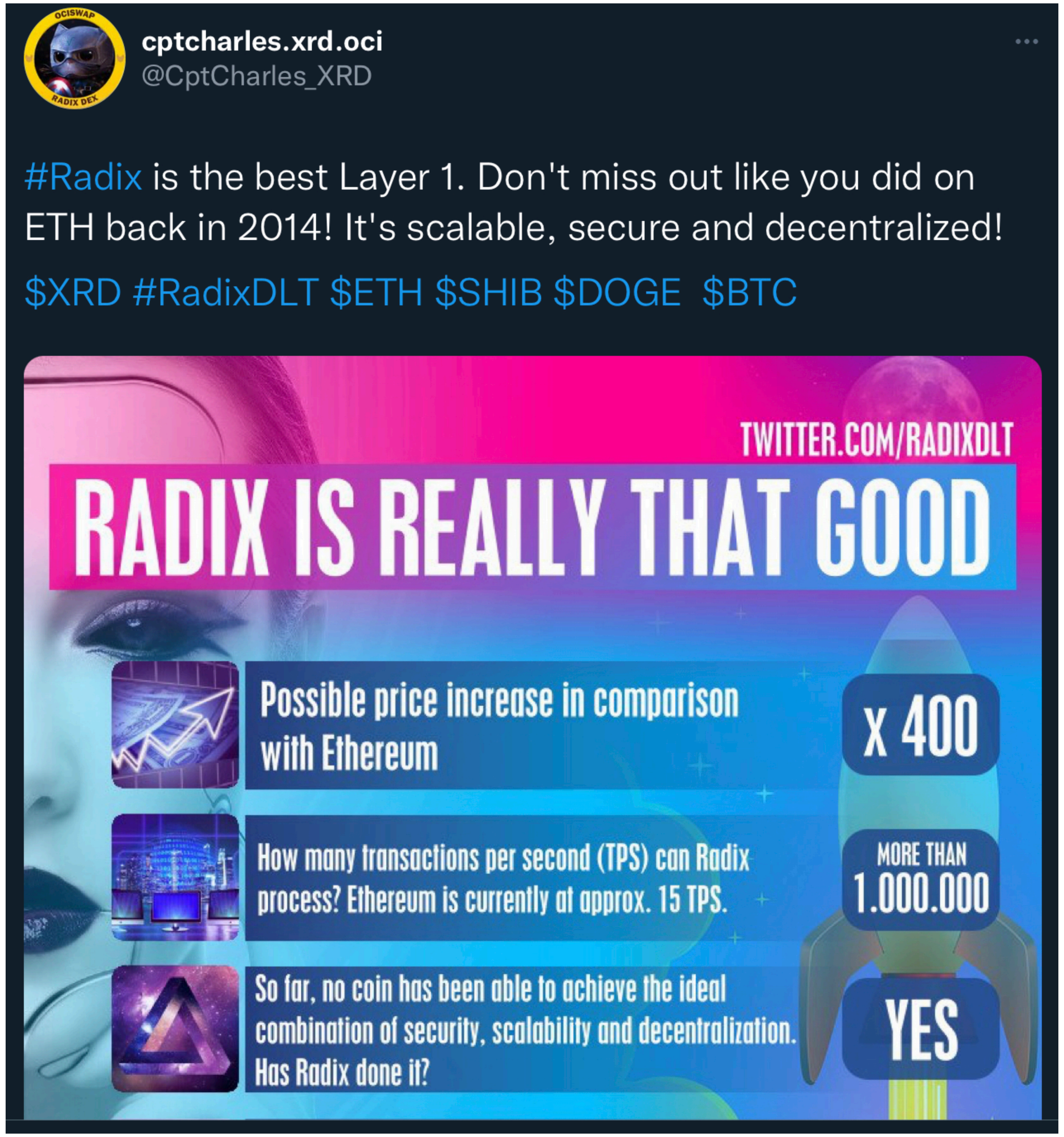
21 Million Ways to Freedom
@21MWTF

Replying to @intocryptoverse

Radix DLT #XRD is an innately sharded DLT that is NOT a blockchain and has infinite scalability (1.4million TPS confirmed and verifiable), enhanced security and decentralised all WITHOUT BREAKING COMPOSABILITY. This is the future of #DeFi!



Fastest network ever demonstrated



cptcharles.xrd.oci
@CptCharles_XRD

#Radix is the best Layer 1. Don't miss out like you did on ETH back in 2014! It's scalable, secure and decentralized!
$XRD #RadixDLT $ETH $SHIB $DOGE $BTC

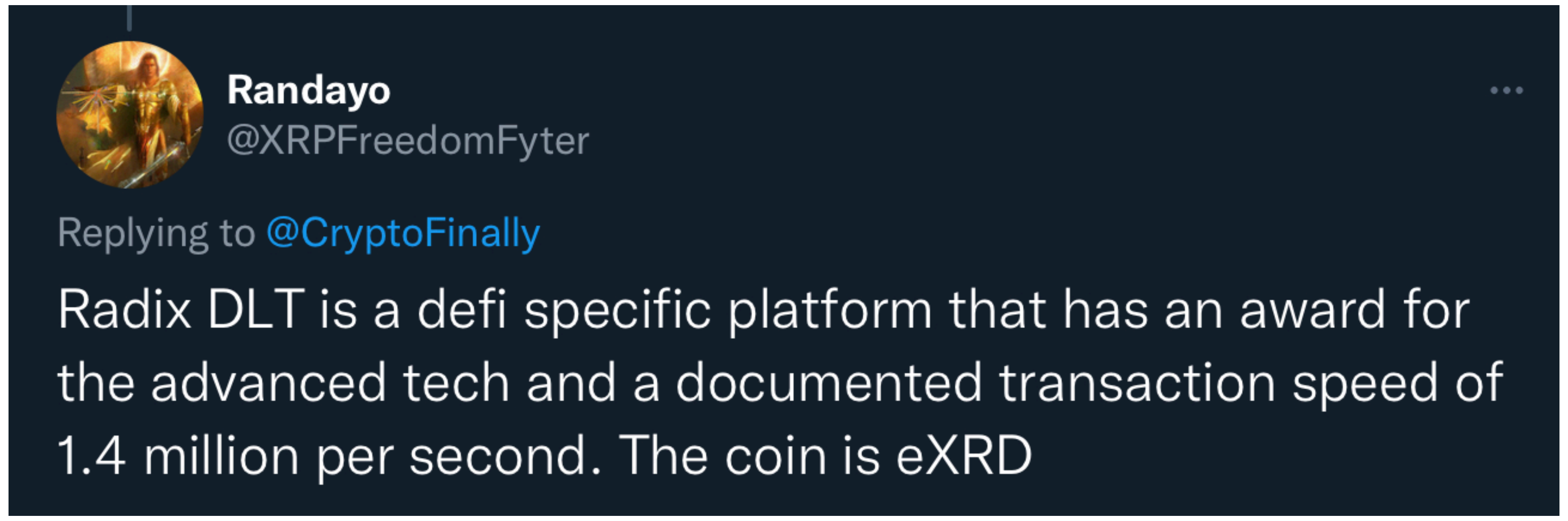
TWITTER.COM/RADIXDLT

RADIX IS REALLY THAT GOOD

Possible price increase in comparison with Ethereum — X 400

How many transactions per second (TPS) can Radix process? Ethereum is currently at approx. 15 TPS. — MORE THAN 1.000.000

So far, no coin has been able to achieve the ideal combination of security, scalability and decentralization. Has Radix done it? — YES

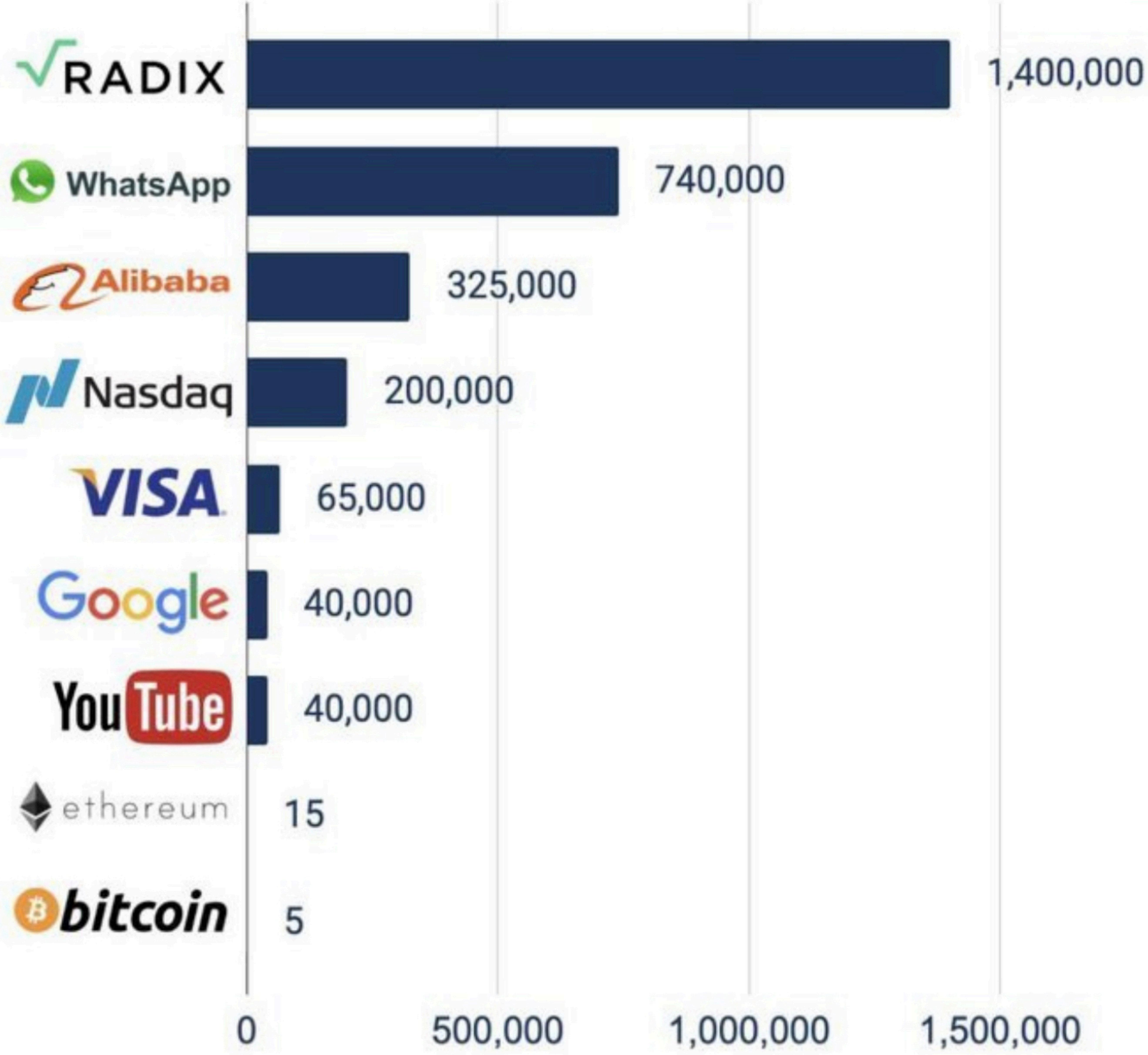


Randayo
@XRPFreedomFyter

Replying to @CryptoFinally

Radix DLT is a defi specific platform that has an award for the advanced tech and a documented transaction speed of 1.4 million per second. The coin is eXRD

Fastest network ever demonstrated



Radix execs: Crypto people understand these statements to be about the future

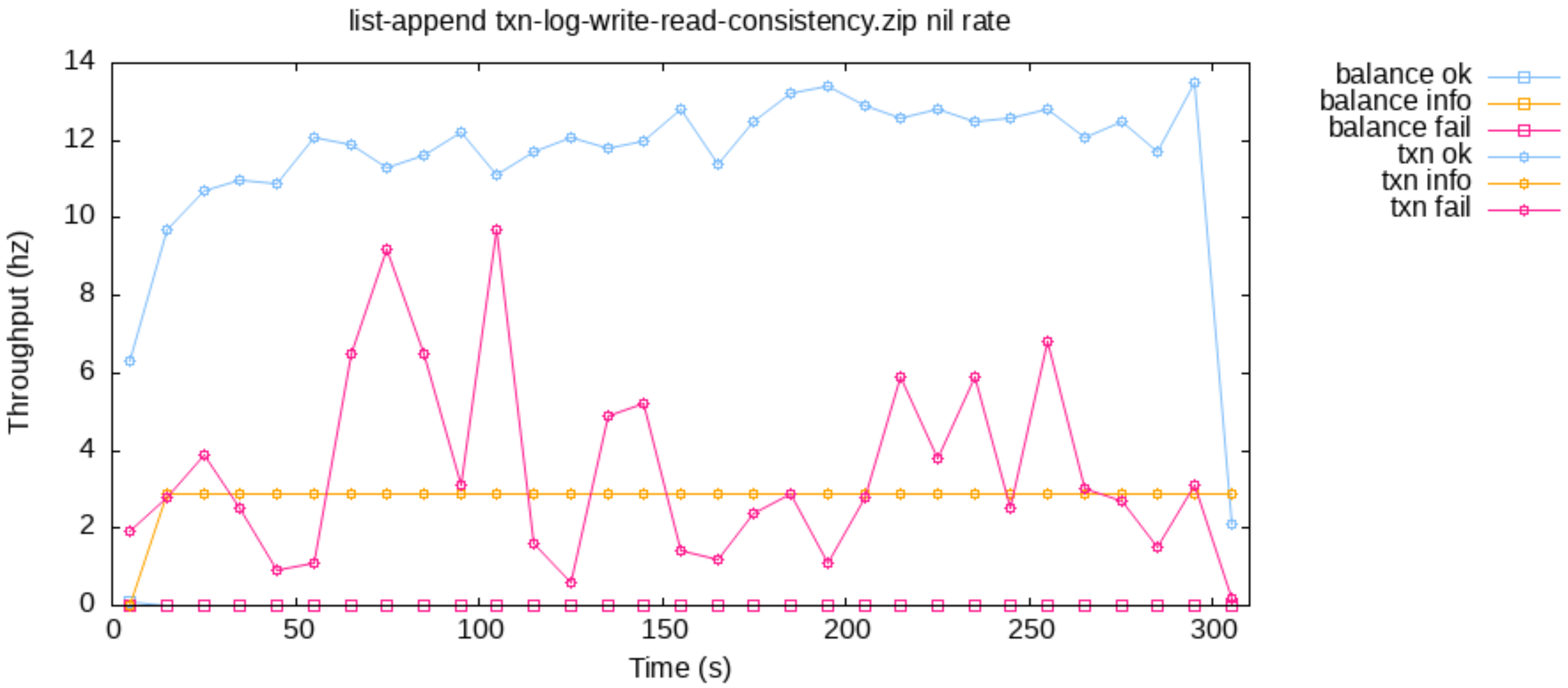
Radix 1.0.beta35.1 – 1.0.2

- All ops through single consensus instance
- No sharding
- Constant scalability

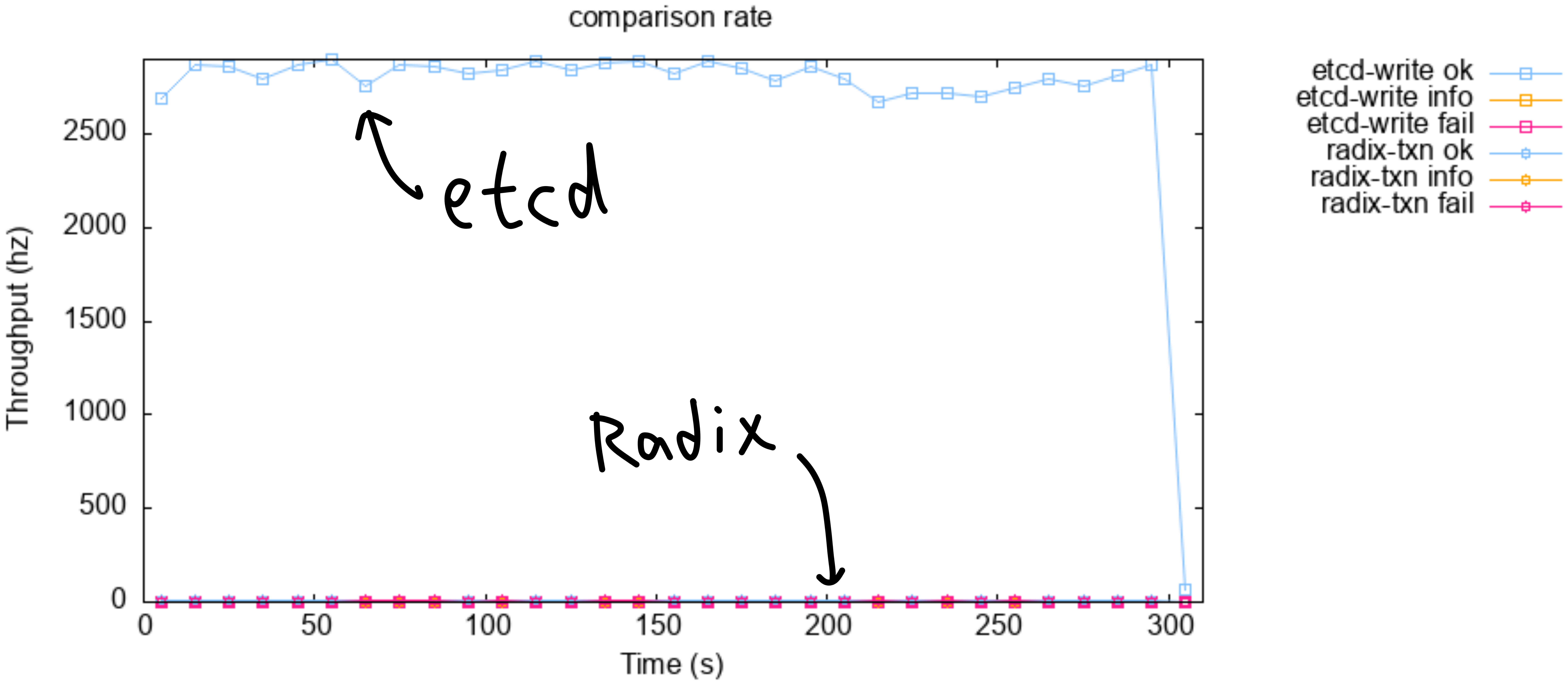
Target: ~50 txns/sec,

~5s max latency

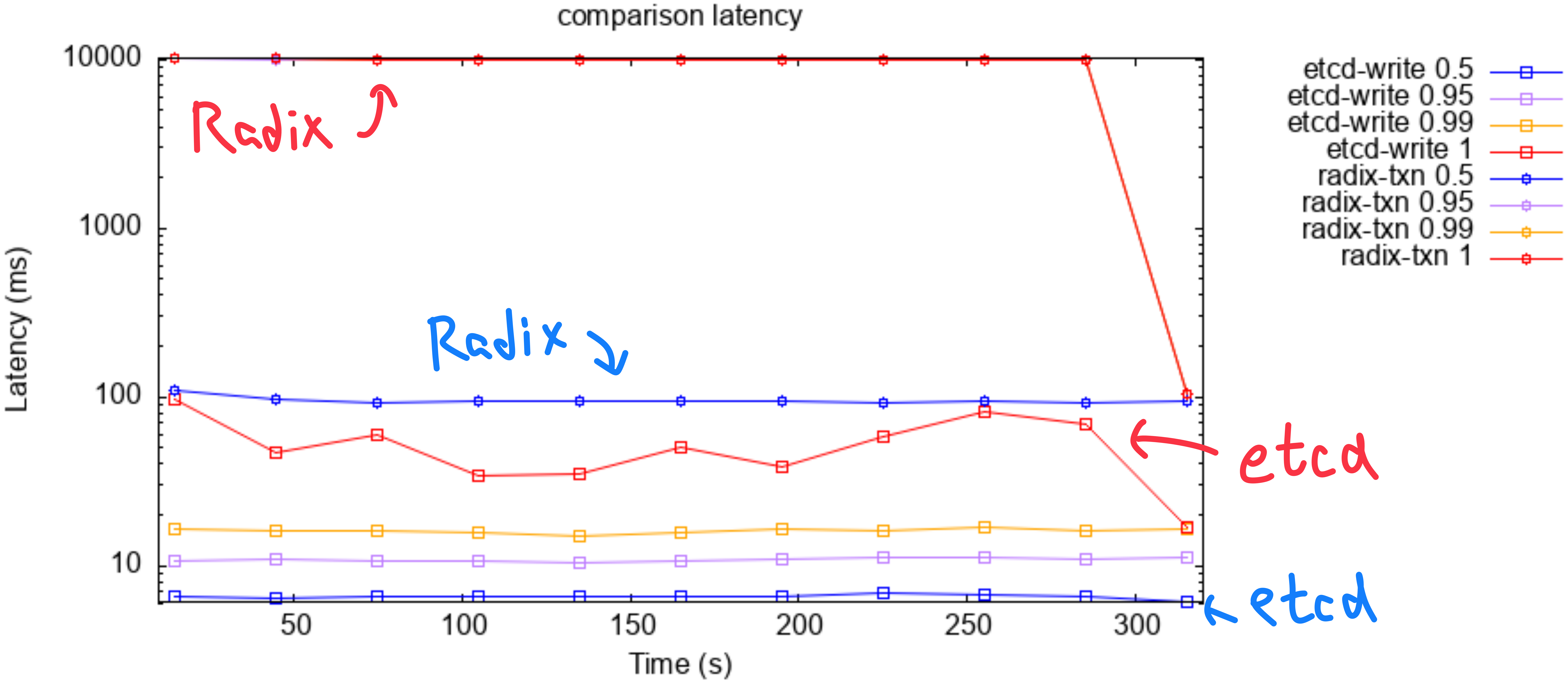
5x m5.xlarge nodes, EBS, writes only



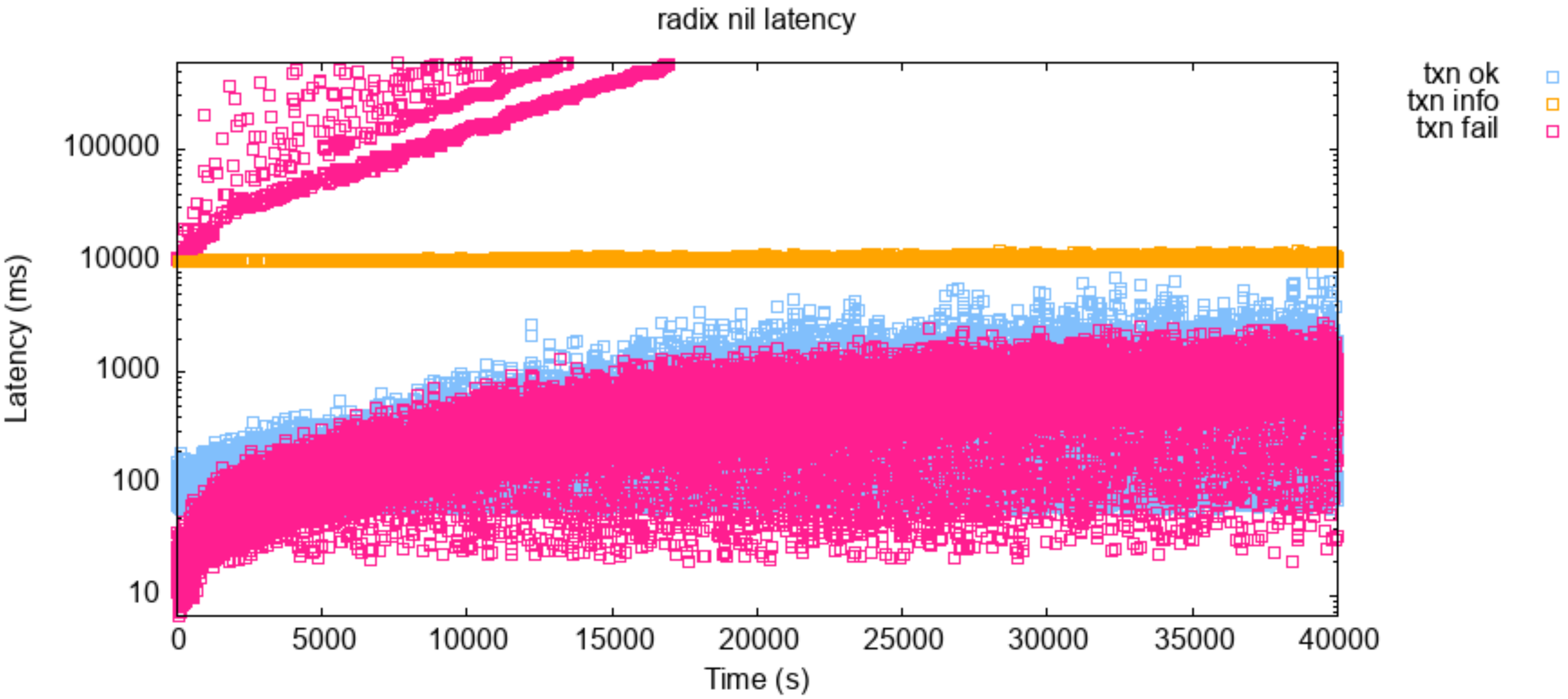
Same nodes, etcd, ~250B writes



2 orders of magnitude longer commit times

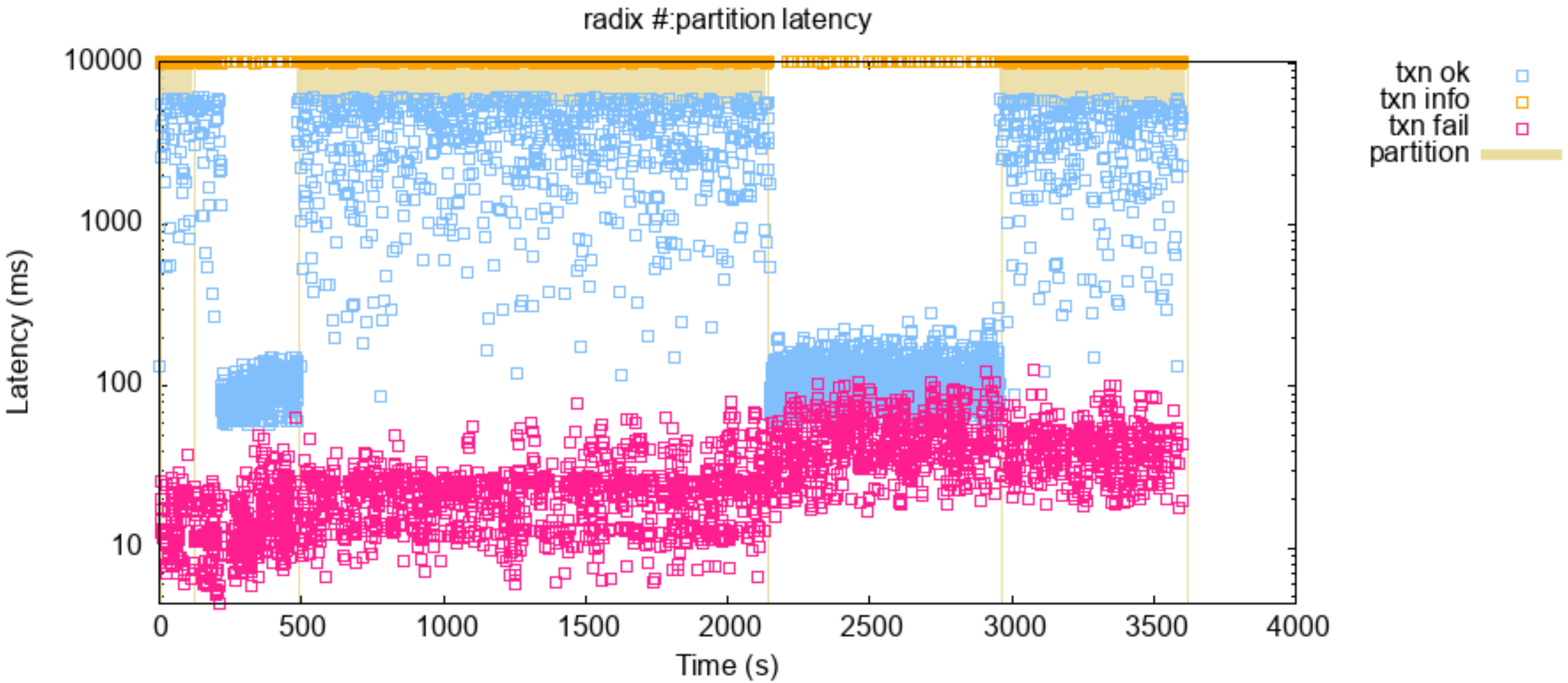


5x m5.large instances, EBS, 3 txns/sec r+w

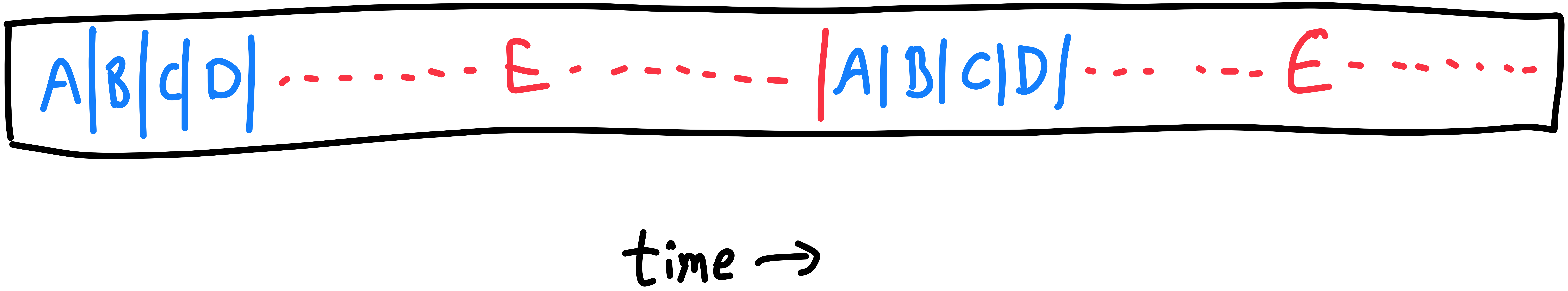


- Goodput peak: ~16 txns/sec
- At 5 txns/sec, ~5-10% of txns never resolved to confirmed/failed!

Single-node faults caused 50x higher latency



- Nodes take turns being leader
- If a node fails, remaining nodes must wait for it to time out each round

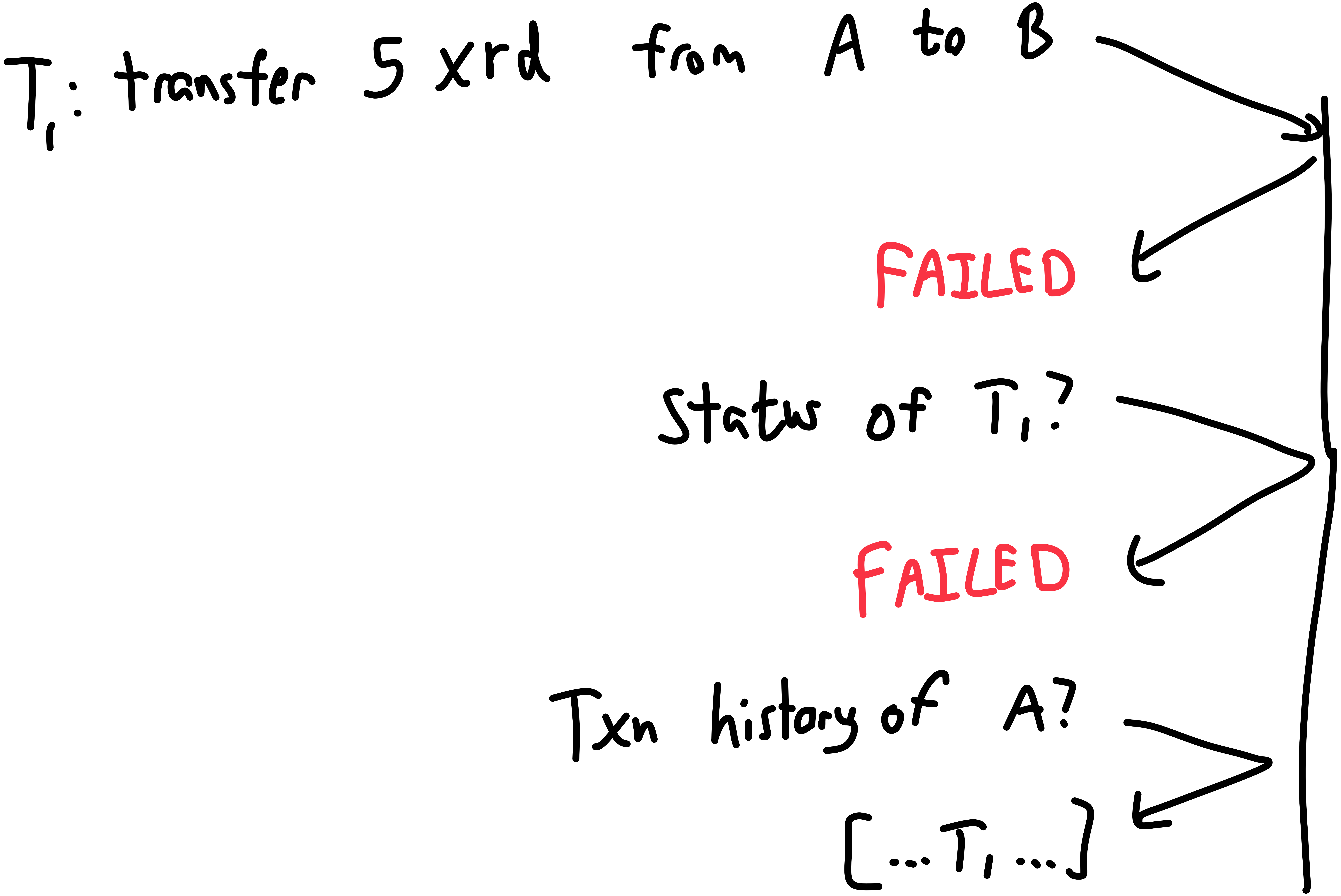


- RDX Works speculated this might not affect production: 100 nodes, higher inter-node latency

- Feb 2022: single validator ("Viskosity") went down for days, degrading network performance

Safety

G1a



T_1: transfer 5 xrd from A to B

FAILED

Status of T_1?

FAILED

Txn history of A?

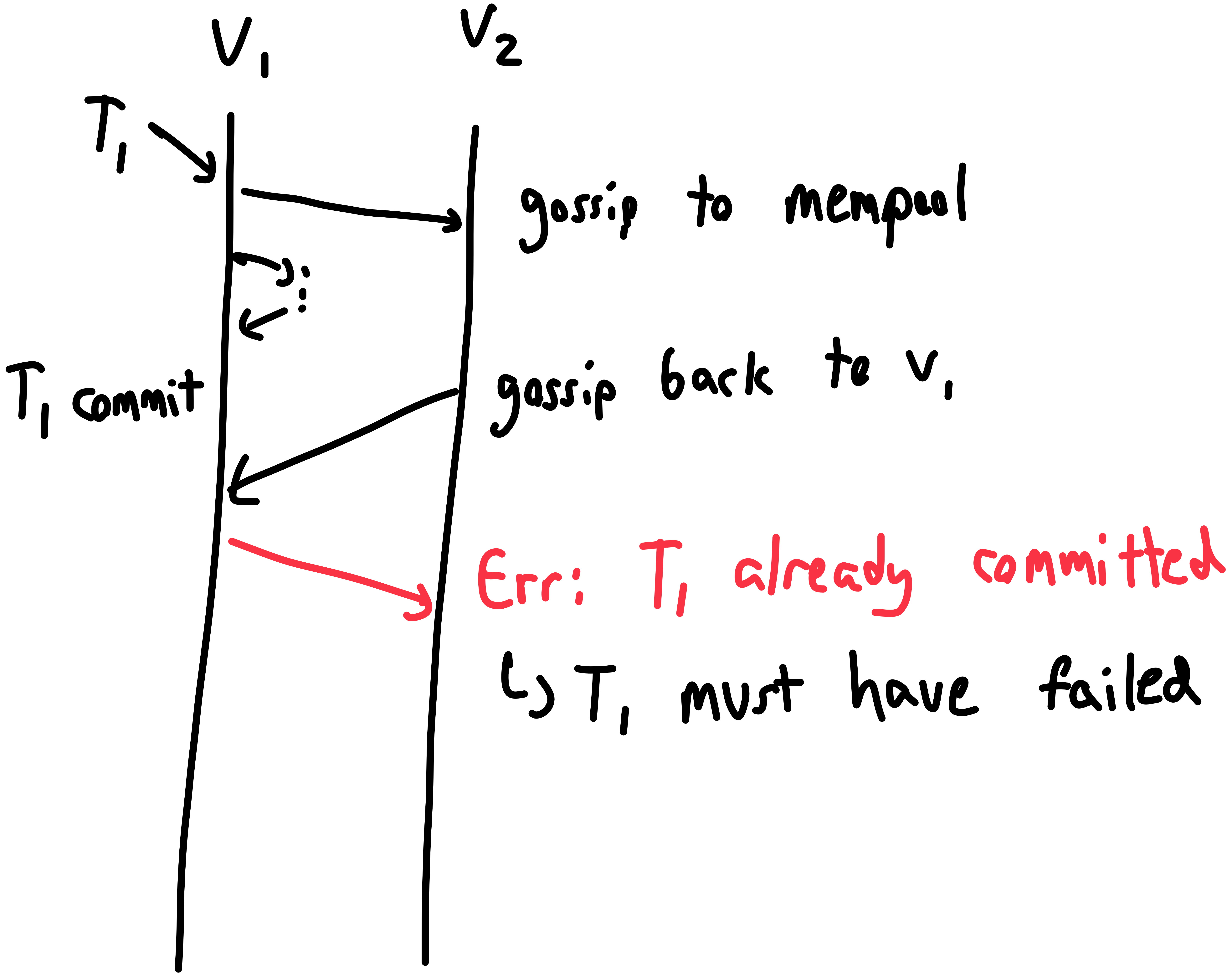
[...T_1...]

G1a in healthy clusters
w/just 1.5 txns/sec

"Surely that's not going to affect real users..."

- October 2021: wrote crawler for Radix public mainnet
- Caught 5 failed-but-actually-committee transactions!
- Flipped from failed -> confirmed in hours/days

(Guess)



gossip to mempool

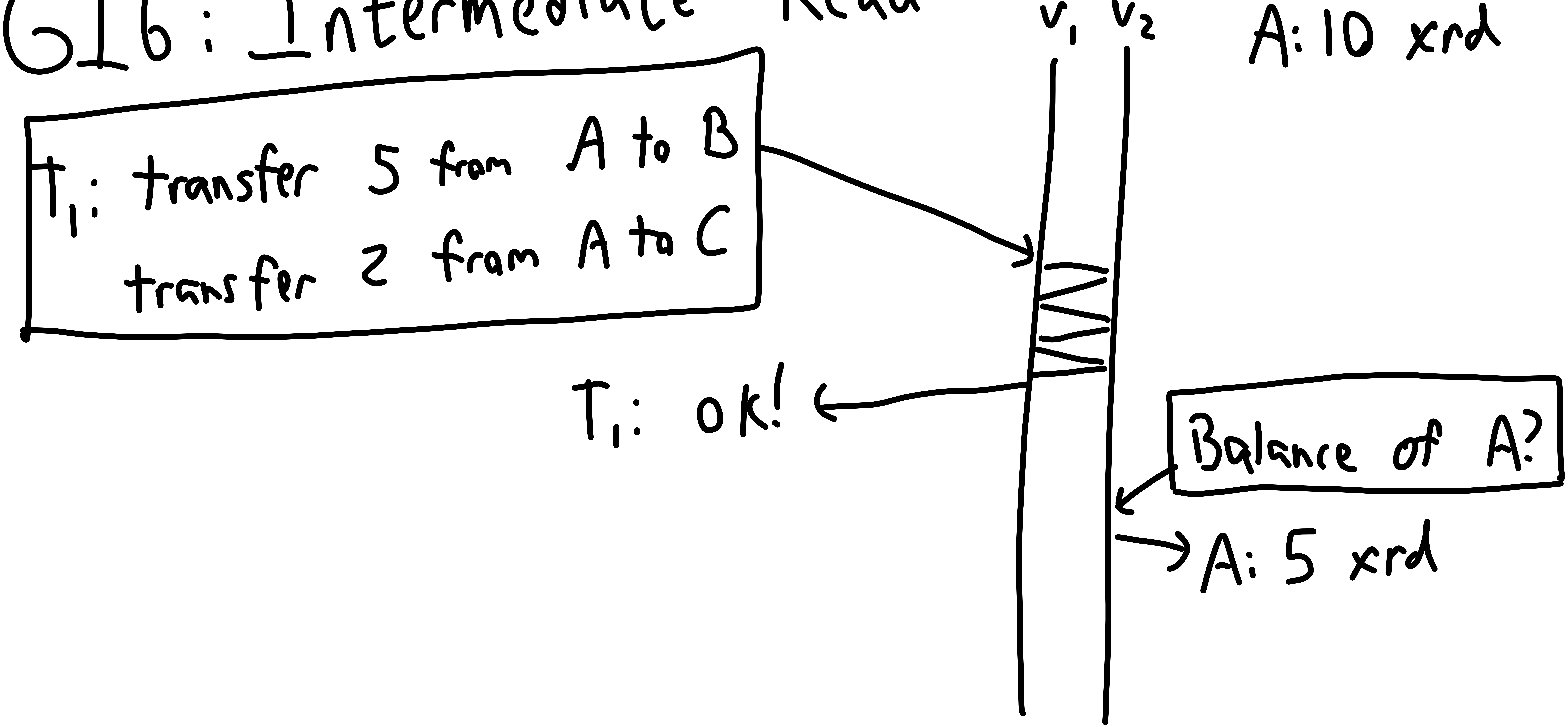
gassip back to v_1

Err: T_1 already committed

↳ T_1 must have failed

Fixed in 48461c4

G1b: Intermediate Read

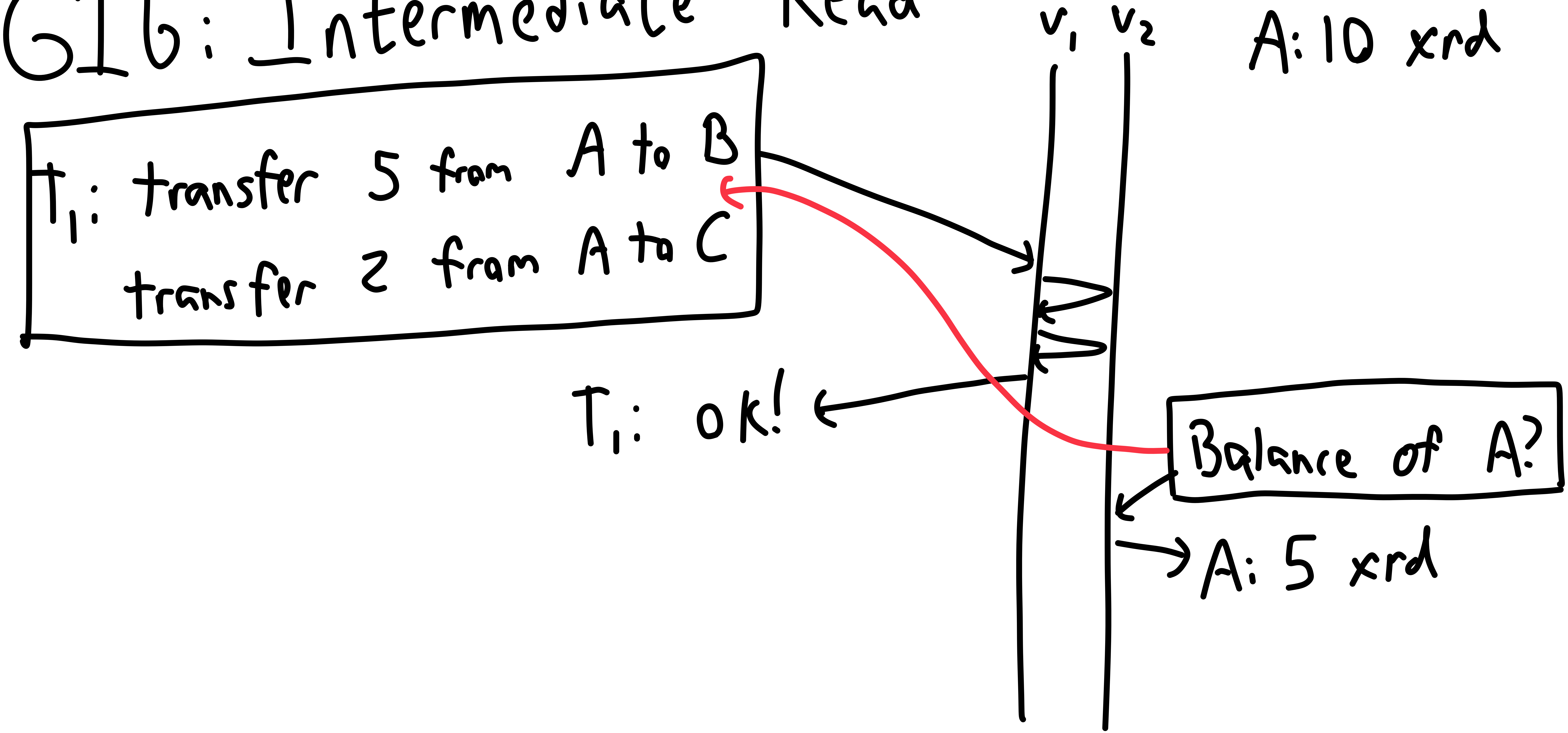


T1: transfer 5 from A to B
transfer 2 from A to C

T1: OK!

Balance of A?

G1b: Intermediate Read



T1: transfer 5 from A to B
transfer 2 from A to C

T1: ok!

Balance of A?

A: 5 xrd

- In healthy clusters, 10 txns/sec...
- ~1:300 reads obsrerved intermediate state

→ Fixed in f616c43 by rewriting account info storage service

G0: Dirty write

T_1: Transfer A->B

T_2: Transfer A->B

T_3: Transfer A->B

Log A: $[T_1, T_3]$

Log B: $[T_1, T_2, T_3]$

G0: Dirty write

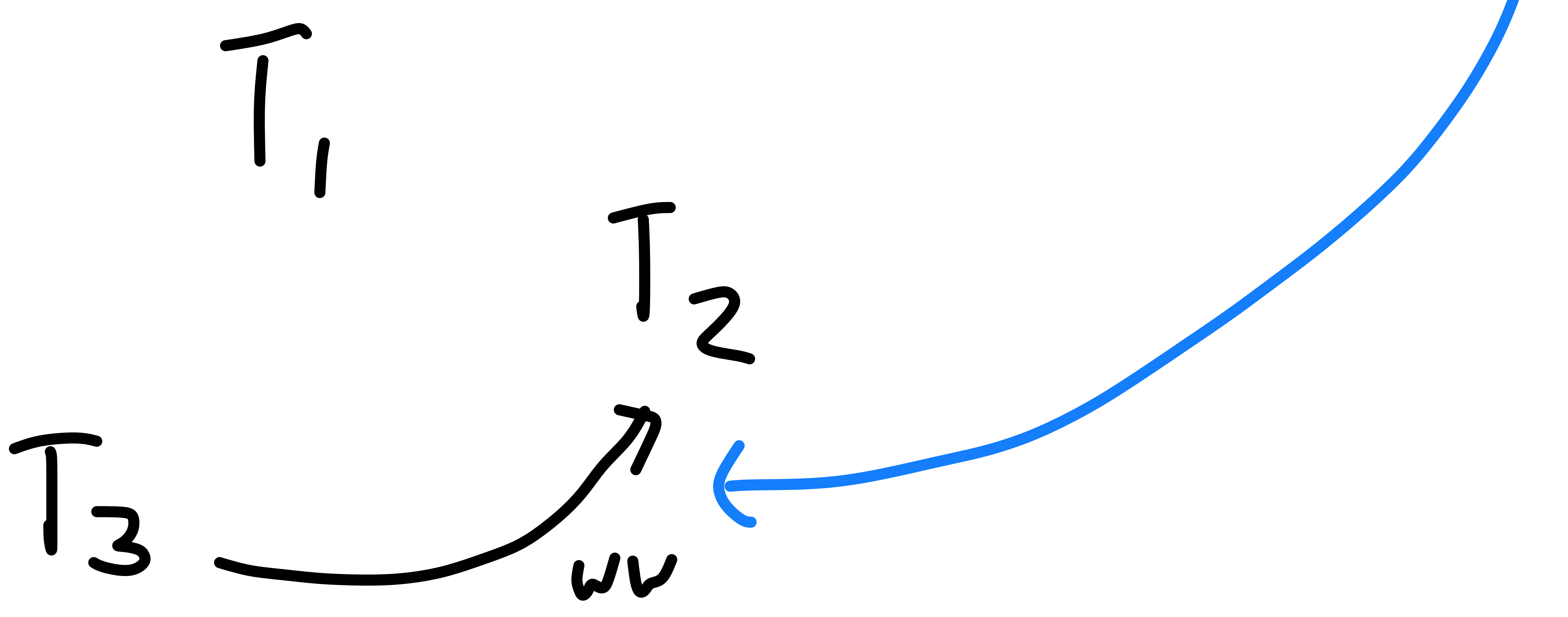
T_1: Transfer A→B

T_2: Transfer A→B

T_3: Transfer A→B

Log A: $[T_1, T_3]$ ... T_2?

Log B: $[T_1, T_2, T_3]$



G0: Dirty write

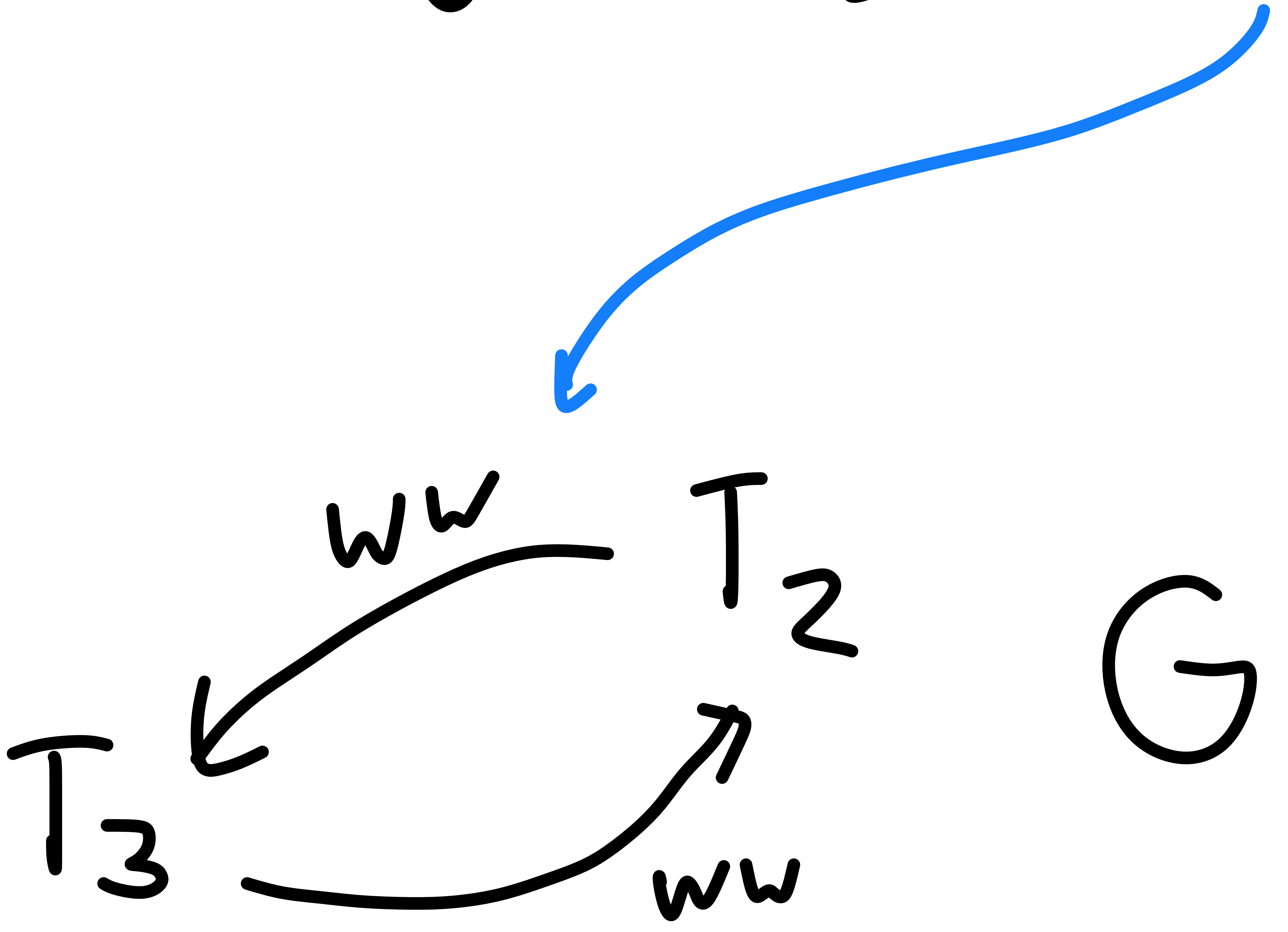
T_1: Transfer A->B

T_2: Transfer A->B

T_3: Transfer A->B

Log A: $[T_1, T_3]$

Log B: $[T_1, T_2, T_3]$



G0!

Log of A: [+10, -2]

Balance of A: 15

Log of A: $[+10, \overbrace{+7}^{\text{invisible}}, -2]$

Balance of A: 15

- Happened rautinely in healthy clusters
- In 1.0.0, 1.0.1, 1.0.2
- Across all nodes
- At 1/8 txn/sec

Reproduced in Stakenet

1 txn/sec

5-10% of txns vanished!

"But surely not in normal usage?"

Mainnet Crawler

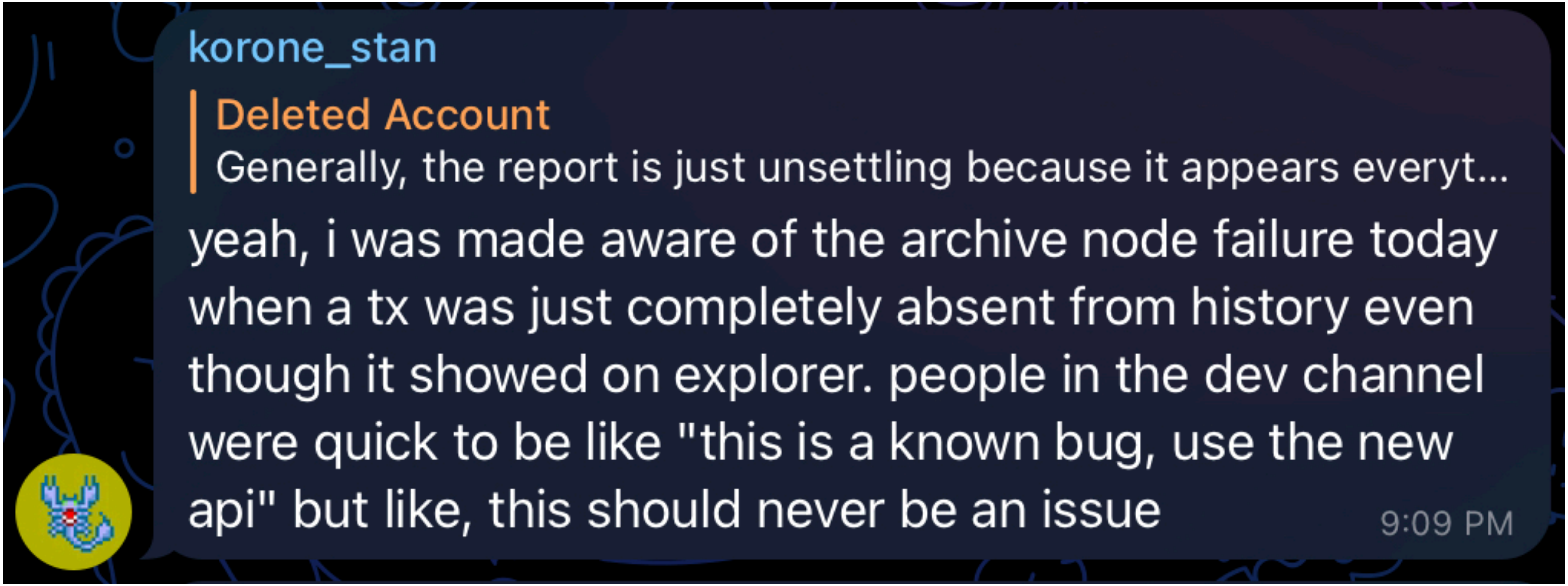
txn 6368485:

transfer 0.8 xrd from ...ge2 -> ...xfx

- present in xfx's log
- missing from ge2's log!

Cause never identified

Fixed by rewrite of txn log archive subsystem in 655dad3



korone_stan

Deleted Account
Generally, the report is just unsettling because it appears everyt...

yeah, i was made aware of the archive node failure today when a tx was just completely absent from history even though it showed on explorer. people in the dev channel were quick to be like "this is a known bug, use the new api" but like, this should never be an issue

9:09 PM

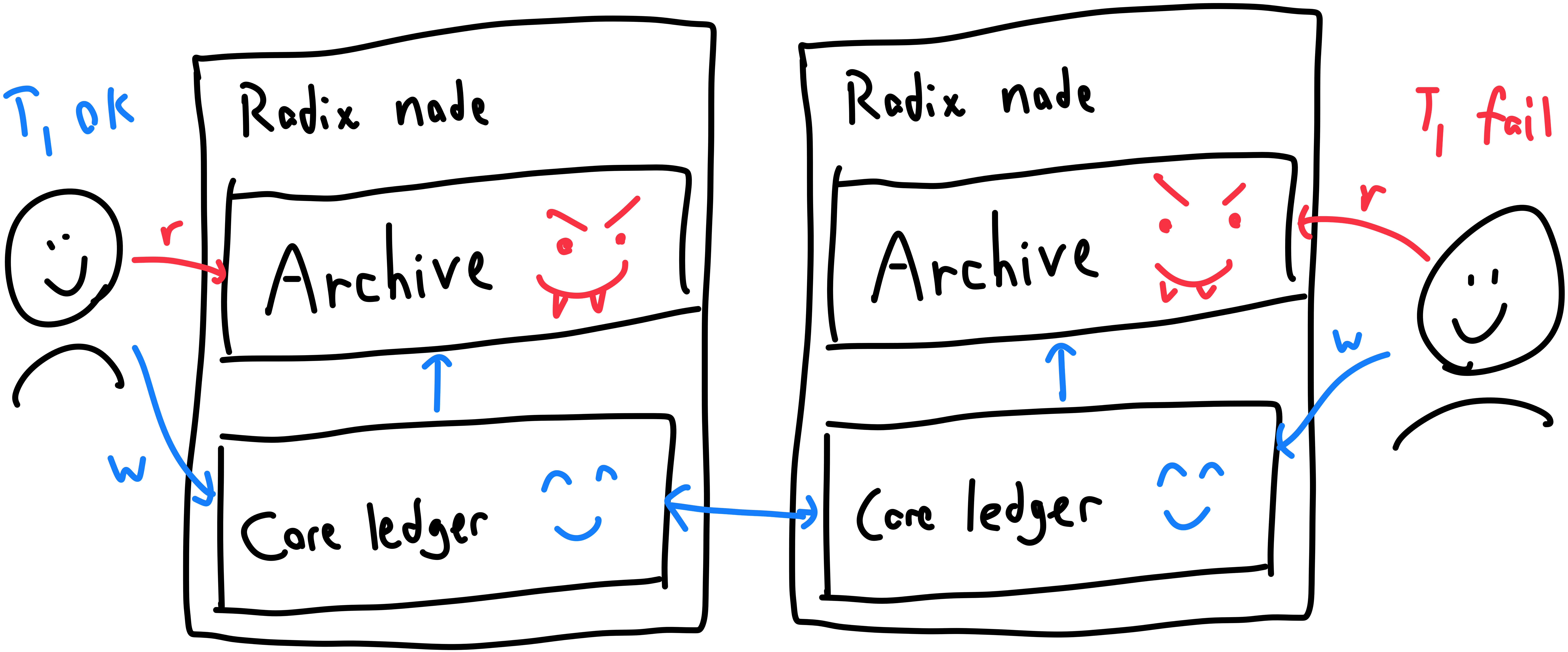
Feb, 2022

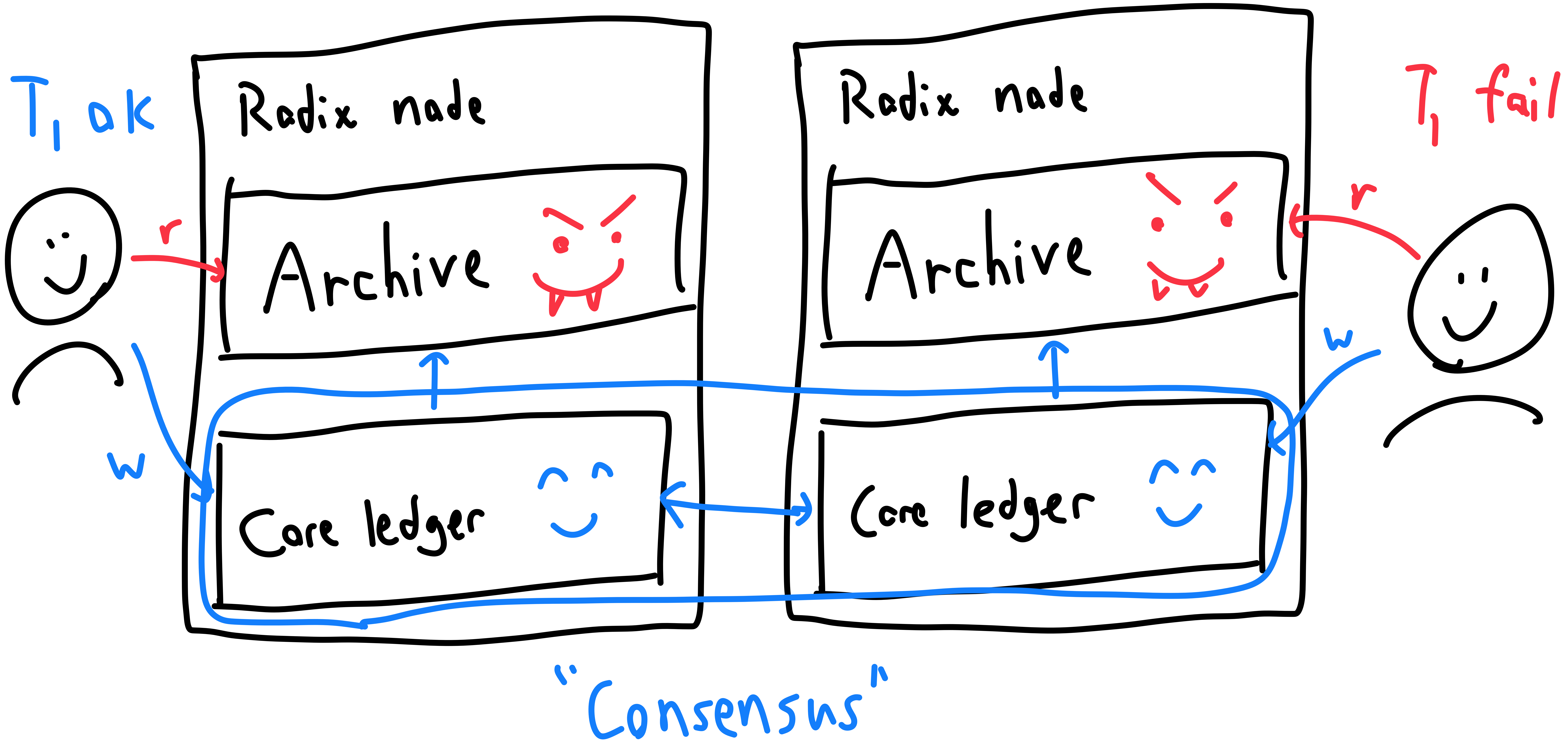
Split-Brain on process crash

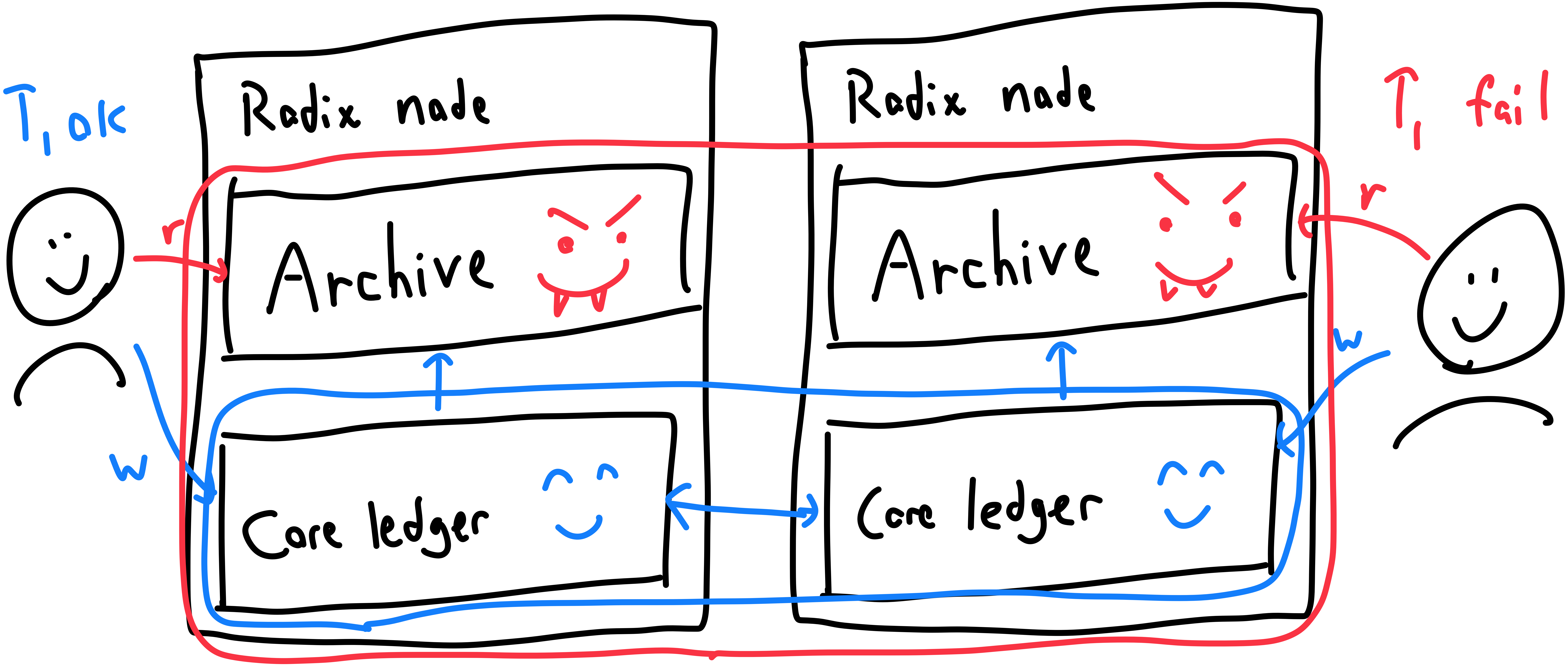
Time(s)	Node	Log of account #16
134	n1	264
138	n1	264, 474
138	n4	264, 267, 474,
139	n1	264, 474
⋮	⋮	⋮
340	n3	264, 267, 474, 812, 831, ...
⋮	⋮	⋮
592	n1	264, 474, 812, 831, ...

Cause never identified

Fixed by rewrite of txn log archive subsystem in 655dad3

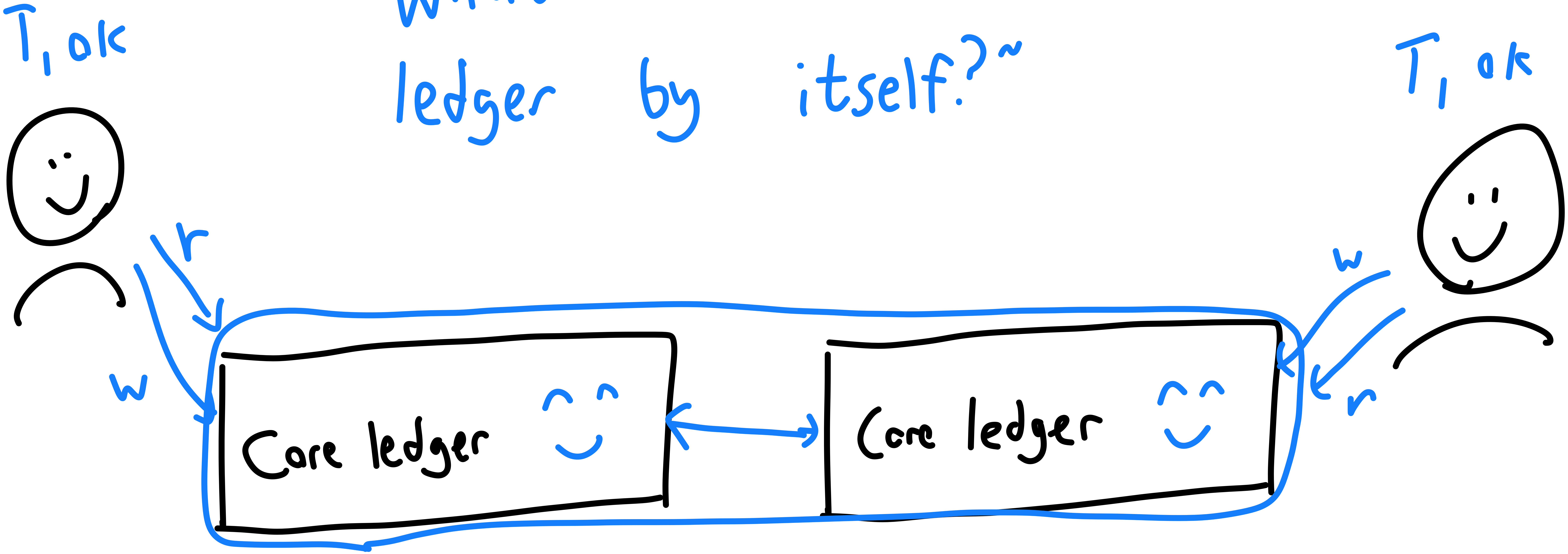

T1 ok
r
w
Radix nade
Archive
Core ledger
Radix nade
Archive
Core ledger
T1 fail
r
w


T1 ok
r
w
Radix node
Archive
Core ledger
Radix node
Archive
Core ledger
T1 fail
r
w
"Consensus"



Also consensus!

"What about the core ledger by itself?"



Time (s)	Node	Raw transaction log
359	n4	... T_1, T_2, T_3
359	n1	... T_1, T_2, T_3, T_4, T_5
359	n5	... T_1, T_2, T_3, T_4, T_5
359	n3	... T_1, T_2, T_3, T_4, T_5, T_6
360	n2	... T_1, T_2, T_3, T_4, T_5, T_6
362		(kill all nodes)
369	n5	... T_1, T_2
375	n5	... T_1, T_2, T_7, T_8, T_9

Time (s)	Node	Raw transaction log
359	n4	... T_1, T_2, T_3
359	n1	... T_1, T_2, T_3, T_4, T_5
359	n5	... T_1, T_2, T_3, T_4, T_5
359	n3	... T_1, T_2, T_3, T_4, T_5, T_6
360	n2	... T_1, T_2, T_3, T_4, T_5, T_6
362		(kill all nodes)
369	n5	... T_1, T_2
375	n5	... T_1, T_2, T_7, T_8, T_9

Lost!

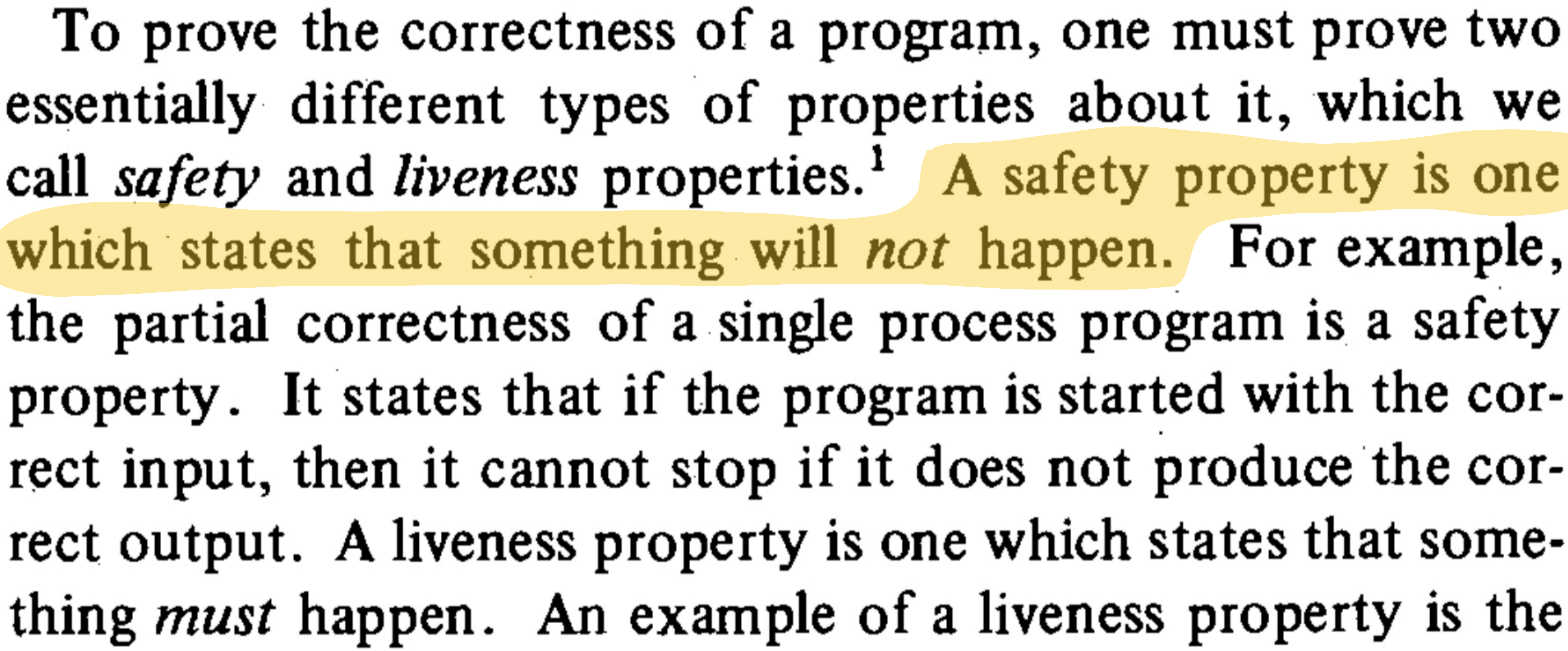
RDX works chose COMMIT_NO_SYNC for writes to BerkeleyDB

→ Fixed in 1.1.0 by using COMMIT_SYNC

№	Summary	Event Required	Reported Fixed
1	Indeterminate transactions during normal operation	None	1.1.0
2	High latencies during single faults	Single crash, partition, etc.	Unresolved
3	Non-monotonic reads	None	1.1.0
4	Missing & extra actions in transaction logs	None	1.1.0
5	Premature commits in development builds	None	350ac77
6	Committed transactions have status `FAILED`	None	1.1.0
7	Missing transactions in transaction logs	None	1.1.0
8	Contradictory transaction logs	None	1.1.0
9	Split-brain transaction loss	Single-node crash	1.1.0
10	Loss of committed transactions from raw log	All nodes crash	1.1.0
11	Intermediate balance reads	None	1.1.0
12	More committed transactions with status `FAILED`	Network partitions	1.1.0
13	More non-monotonic reads	Membership changes and crashes	1.1.0

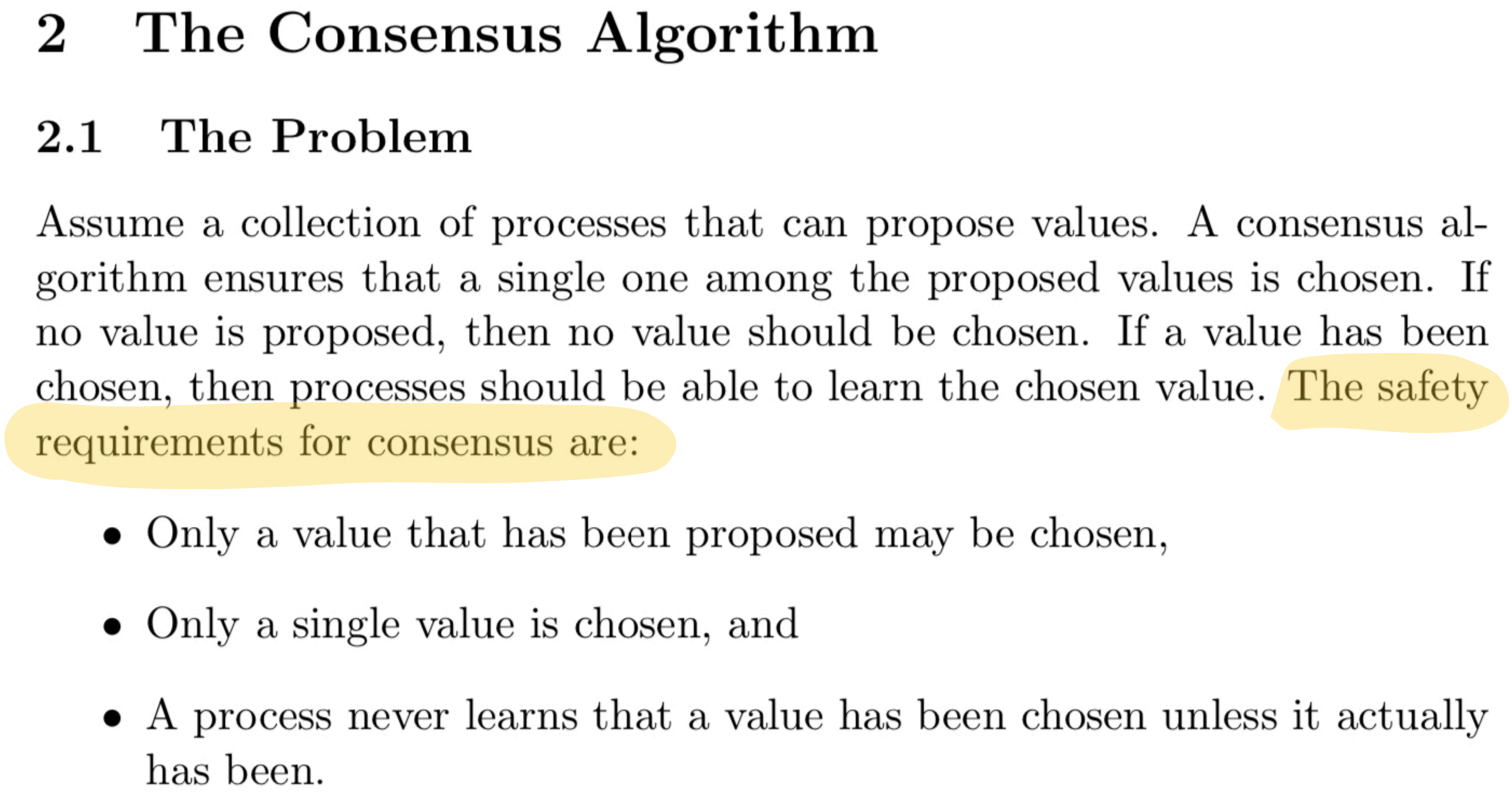
Safety

Lamport, 1977:



To prove the correctness of a program, one must prove two essentially different types of properties about it, which we call *safety* and *liveness* properties.[1] A safety property is one which states that something will *not* happen. For example, the partial correctness of a single process program is a safety property. It states that if the program is started with the correct input, then it cannot stop if it does not produce the correct output. A liveness property is one which states that something *must* happen. An example of a liveness property is the

Lamport 2001: Paxos Made Simple



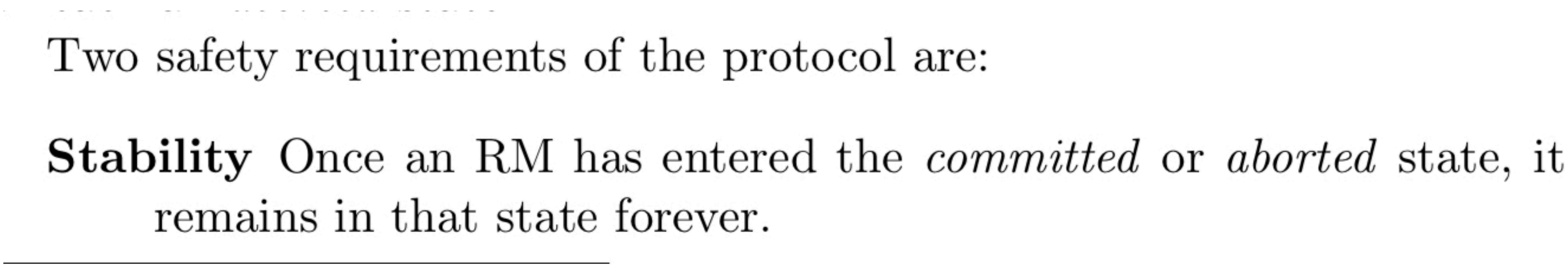
2 The Consensus Algorithm

2.1 The Problem

Assume a collection of processes that can propose values. A consensus algorithm ensures that a single one among the proposed values is chosen. If no value is proposed, then no value should be chosen. If a value has been chosen, then processes should be able to learn the chosen value. The safety requirements for consensus are:

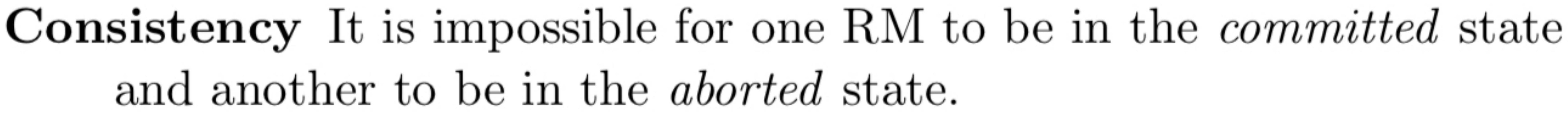
- Only a value that has been proposed may be chosen,
- Only a single value is chosen, and
- A process never learns that a value has been chosen unless it actually has been.

Gray & Lamport, 2004: Consensus on Txn Commit



Two safety requirements of the protocol are:

Stability Once an RM has entered the *committed* or *aborted* state, it remains in that state forever.



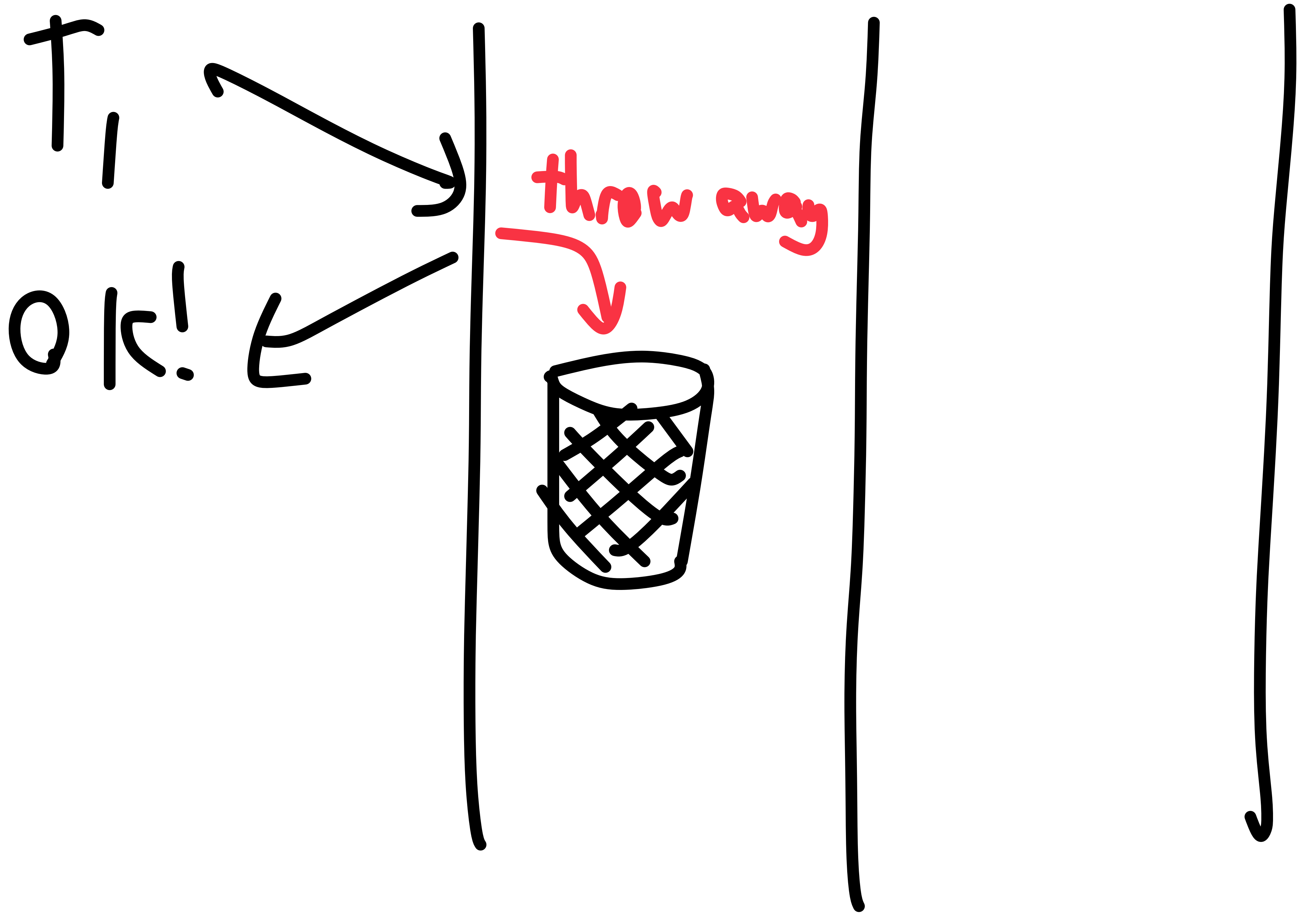
Consistency It is impossible for one RM to be in the *committed* state and another to be in the *aborted* state.

RDX Works:

"A safety violation is defined as two healthy consensus nodes disagreeing on what is the correct ledger state."

This is apparently what "safety" means in crypto!

"A safety violation is defined as two healthy consensus nodes disagreeing on what is the correct ledger state."



A [Radix DLT](https://radixdlt.com/) Olympia network is a distributed, byzantine-fault-tolerant ledger for cryptocurrencies based on delegated proof-of-stake. We evaluated the Radix DLT Olympia Node at version 1.0-beta.35.1, as well as the following 1.0.0 release and various development builds. We found two liveness and 11 safety errors, ranging from stale reads which violated per-server monotonicity, to aborted and intermediate reads, as well as the partial or total loss of committed transactions. At least some of these issues affected users of the public Radix mainnet. As of (TODO: date) the most prominent safety issues had been fixed in unreleased development builds; liveness issues and some less critical safety violations remain unresolved. This work was funded by [Radix Tokens (Jersey) Limited](https://radixdlt.com) with technology consultation and support provided by the team at RDX Works Ltd. This work and conducted in accordance with the [Jepsen ethics policy](https://jepsen.io/ethics.html), and was conducted with the RDX Works team that has developed the Radix Olympia Node.

What do crypto/DeFi people expect ledgers to do, exactly?

Strict-1SR? Read Uncommitted? Weaker?

↑ Goal???

↗ Actual

WHAI is GOING ON?

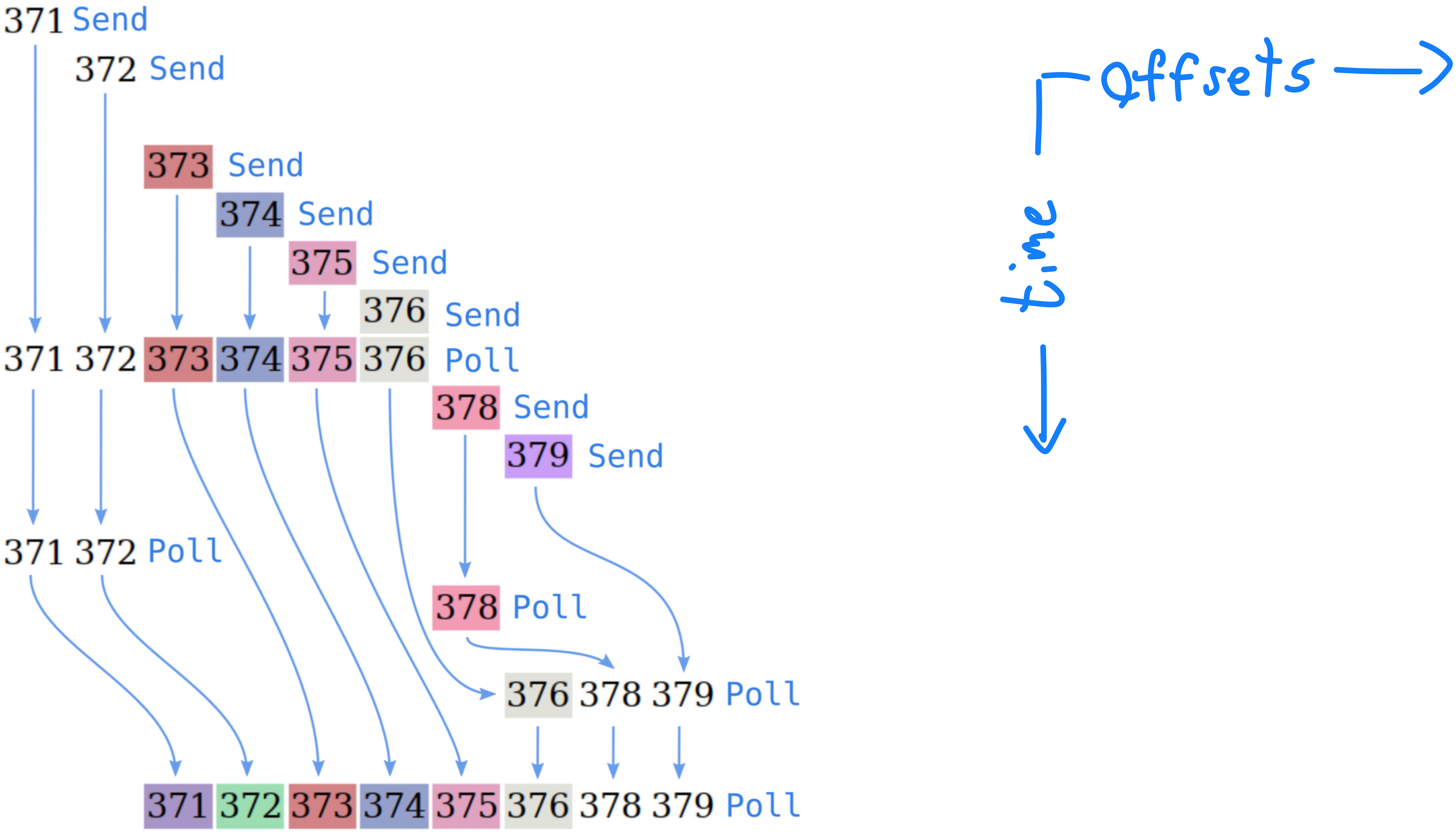
Redpanda

21.10.1 ... 21.11.2

- Distributed Kafka-compatible streaming system
- Append-only, totally ordered logs
- Internal Raft consensus system, instead of Zookeeper

#3003 Incansistent Offsets

w/ process crashes or network partitions...



371 Send

372 Send

373 Send

374 Send

375 Send

376 Send

371 372 373 374 375 376 Poll

378 Send

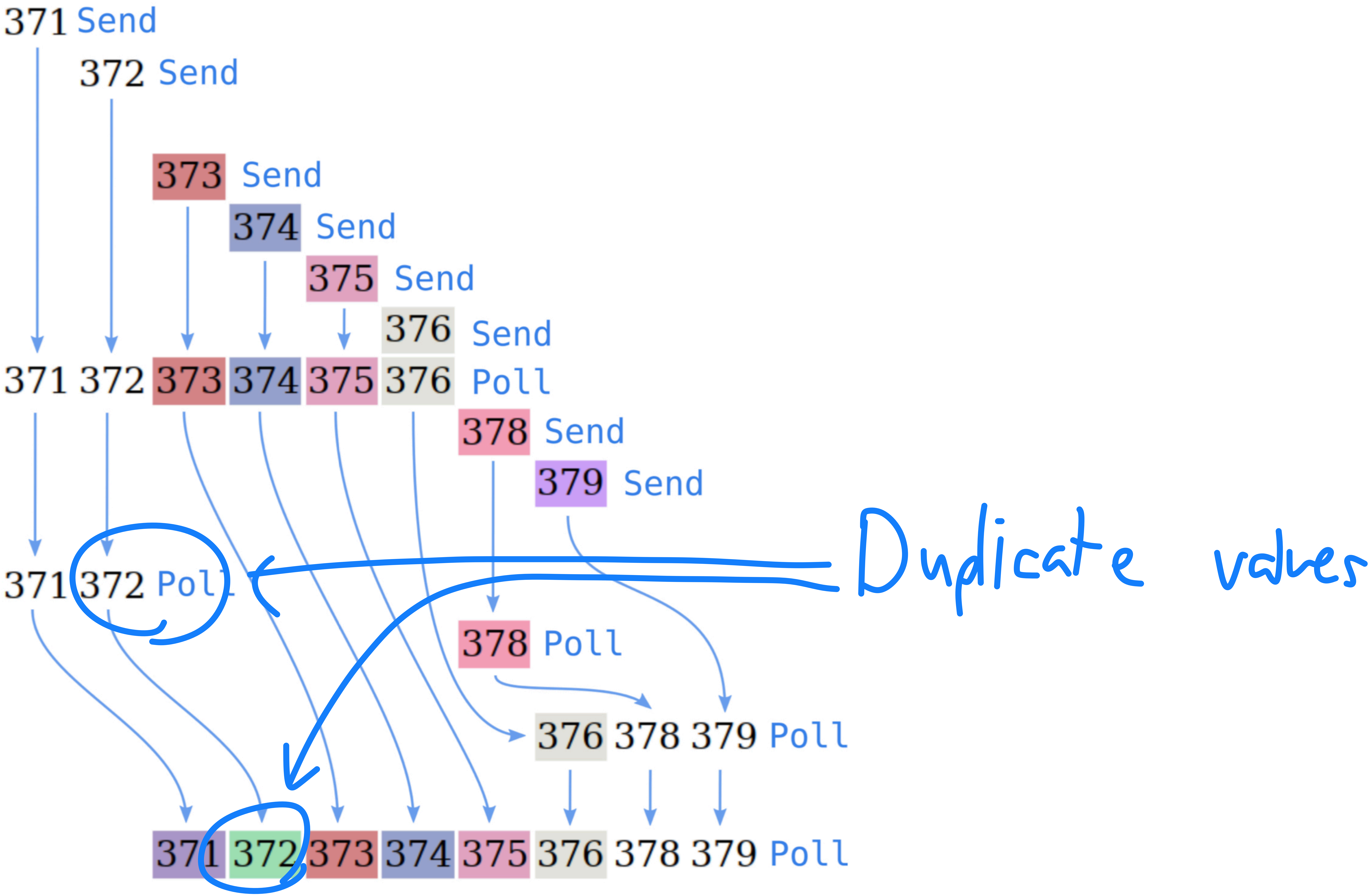
379 Send

371 372 Poll

378 Poll

376 378 379 Poll

371 372 373 374 375 376 378 379 Poll



371 Send

372 Send

373 Send

374 Send

375 Send

376 Send

371 372 373 374 375 376 Poll

378 Send

379 Send

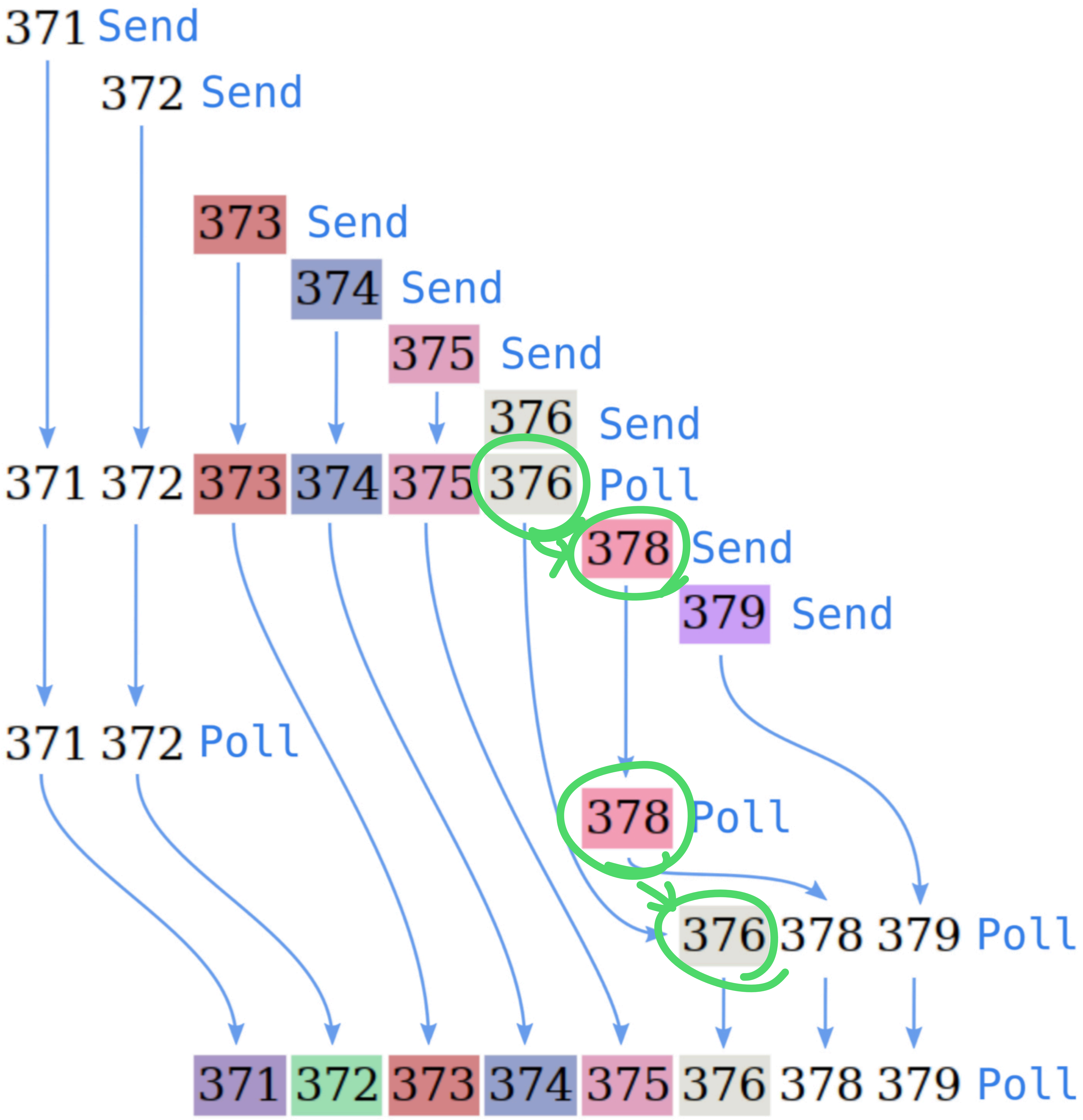
371 372 Poll

Duplicate values

378 Poll

376 378 379 Poll

371 372 373 374 375 376 378 379 Poll



Order inversion

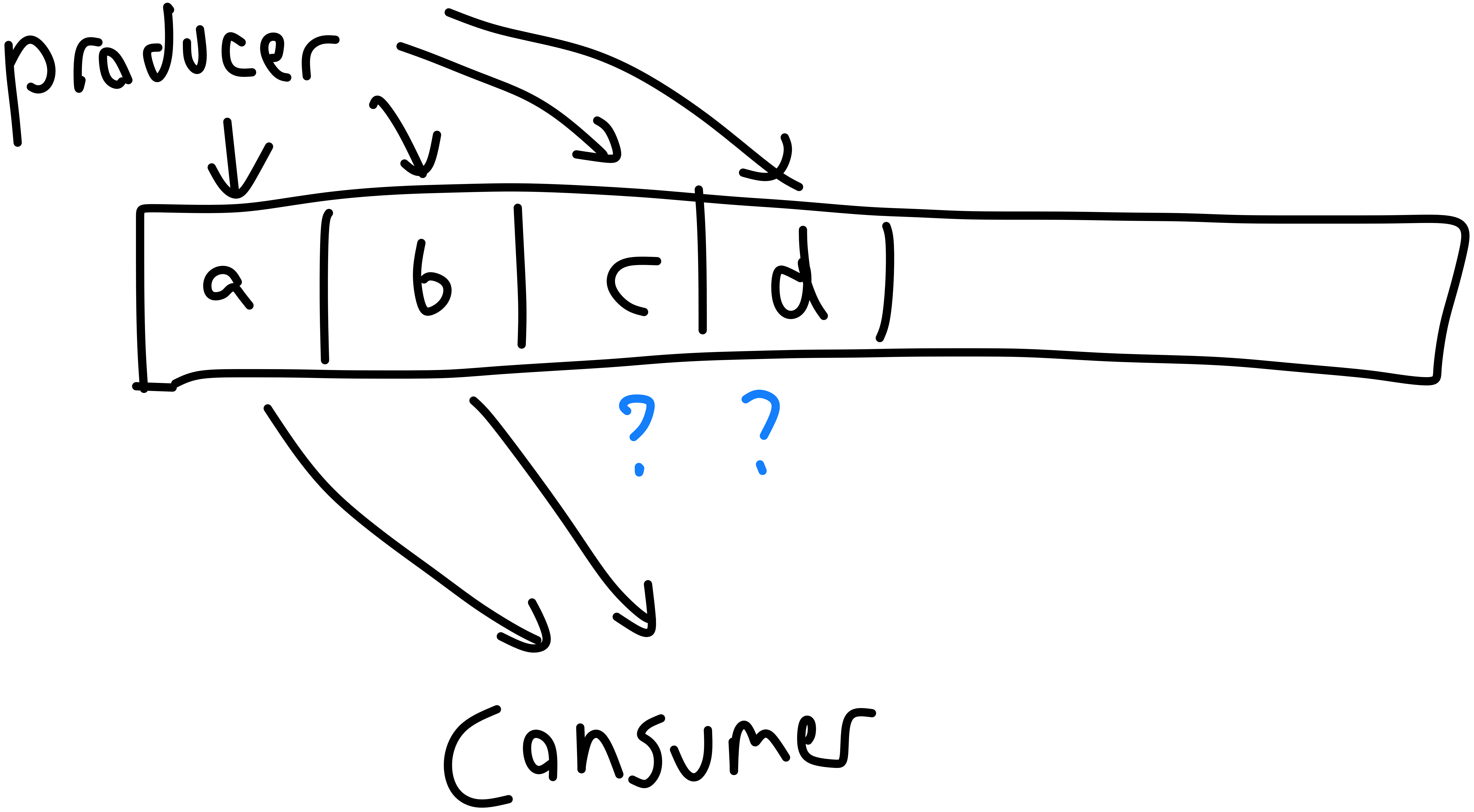
#3003 Inconsistent Offsets

Caused by Raft state machine applying uncommitted log entries

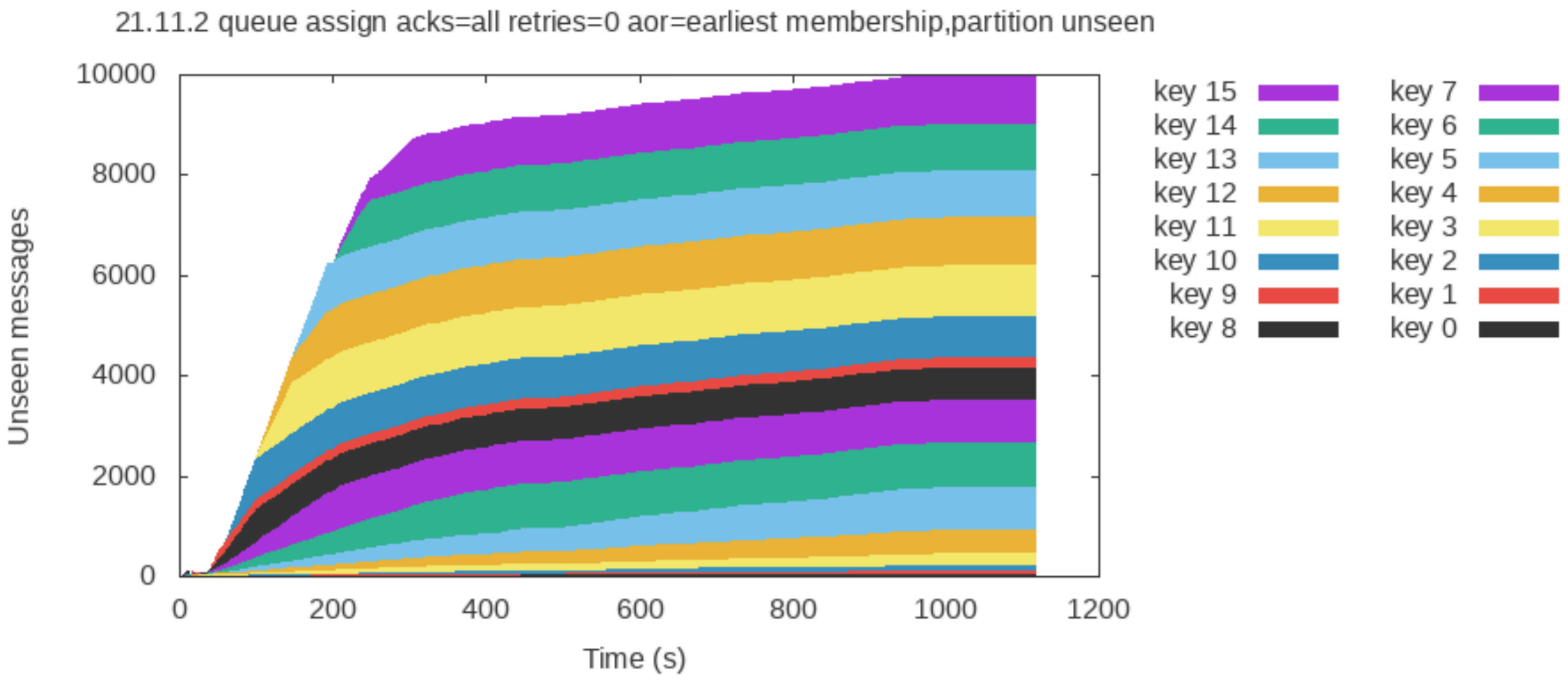
→ Fixed in 21.10.3 by waiting for commit pointer to advance first

#6 Last/Stale Messages

w/process pause



#6 Lost/Stale Messages

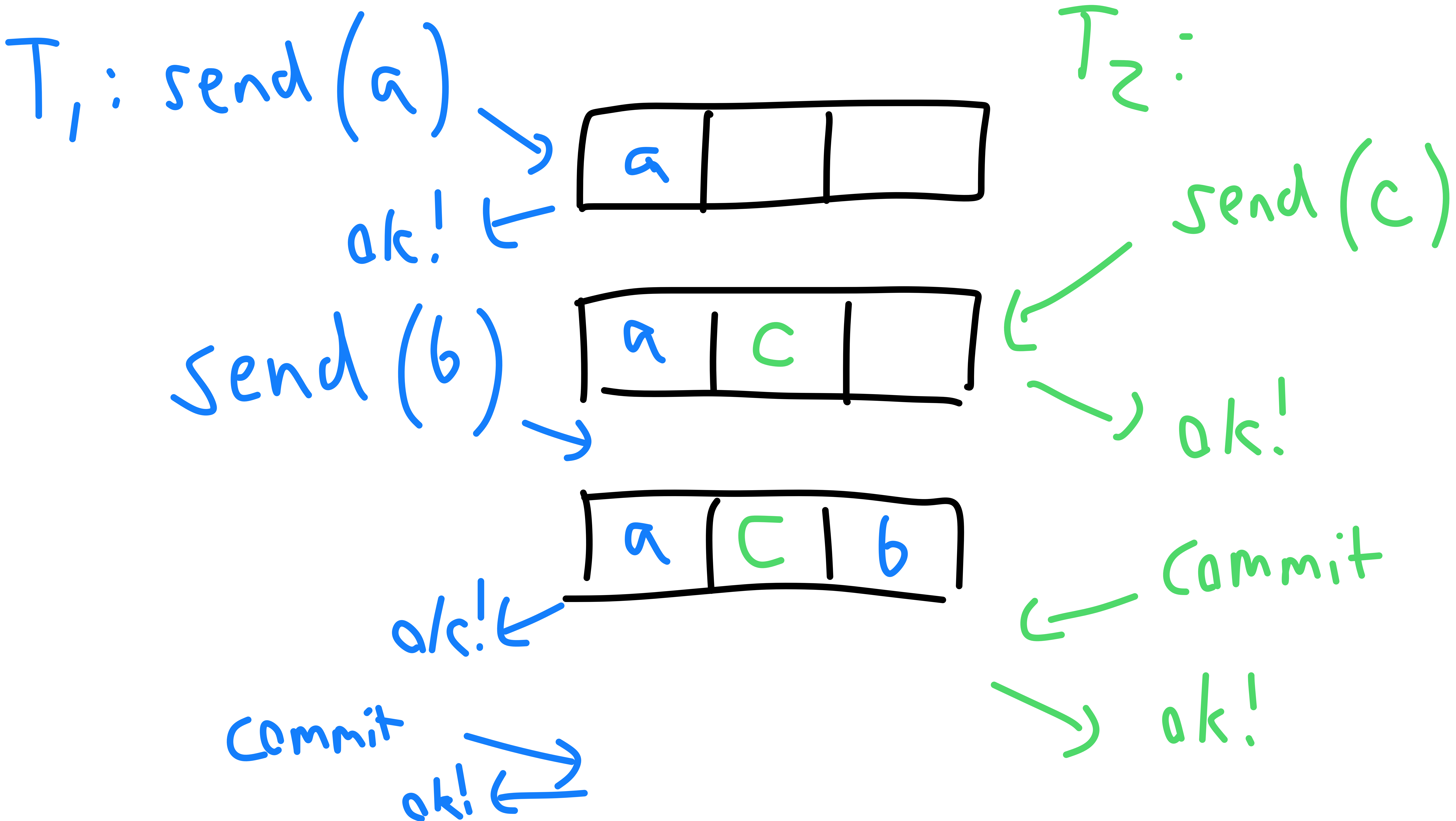


#6 Lost/Stale Messages

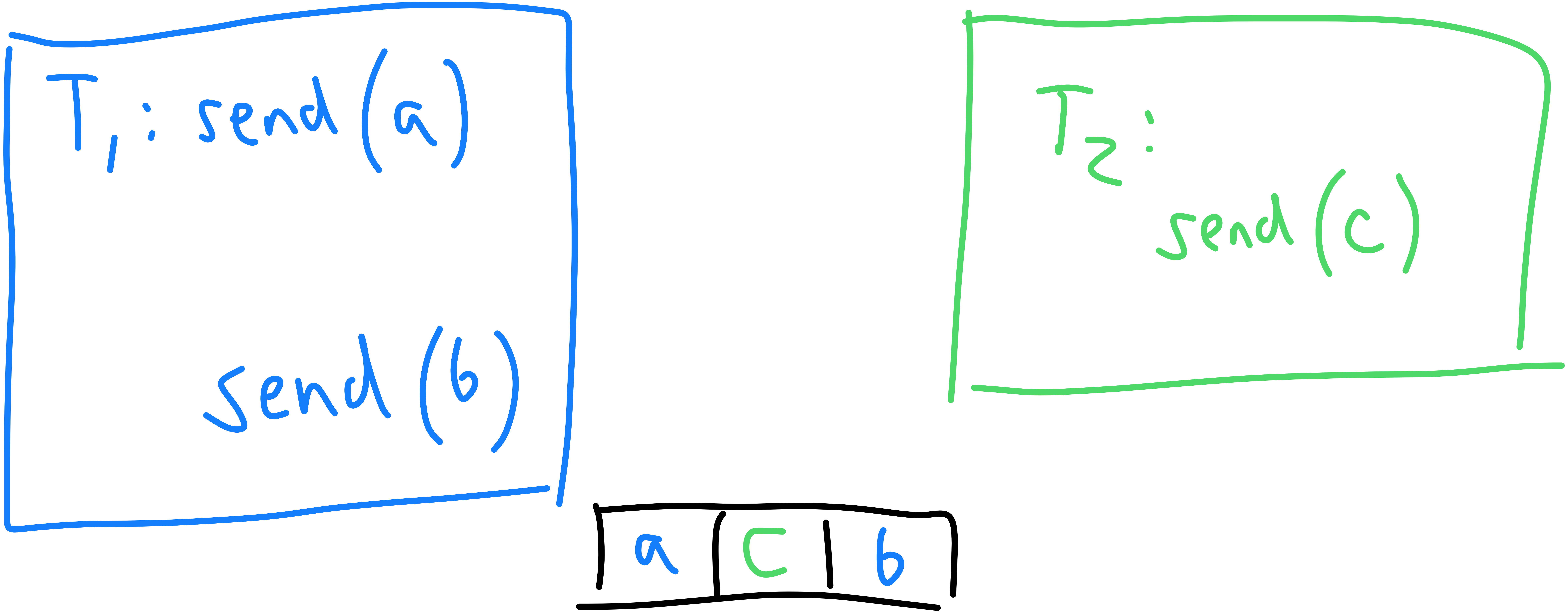
Still on disk... just not being delivered...

Redpanda still investigating!

#8 Write Cycles



#8 Write Cycles



T_1: send(a)

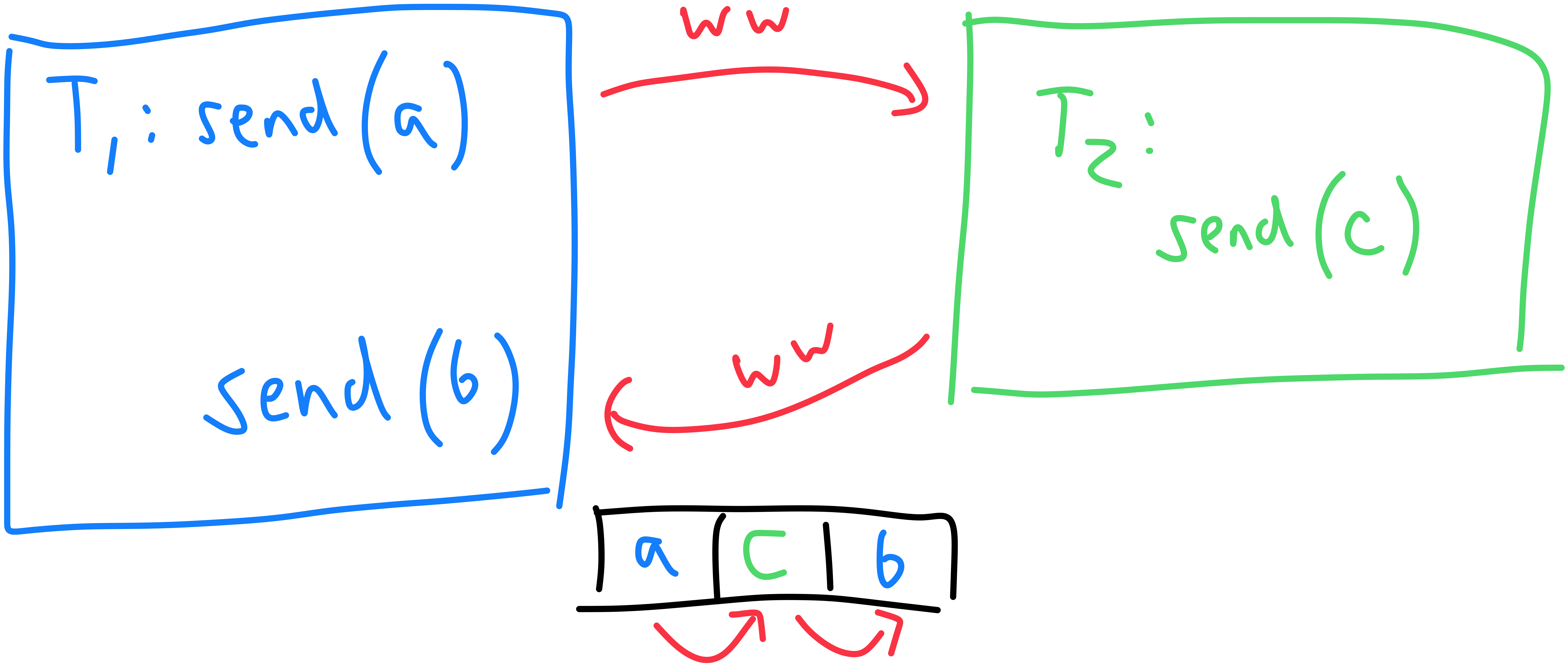
send(b)

T_2:

send(c)

a	c	b

#8 Write Cycles



#8 Write Cycles

G0 happens constantly in healthy Redpanda & Kafka clusters...

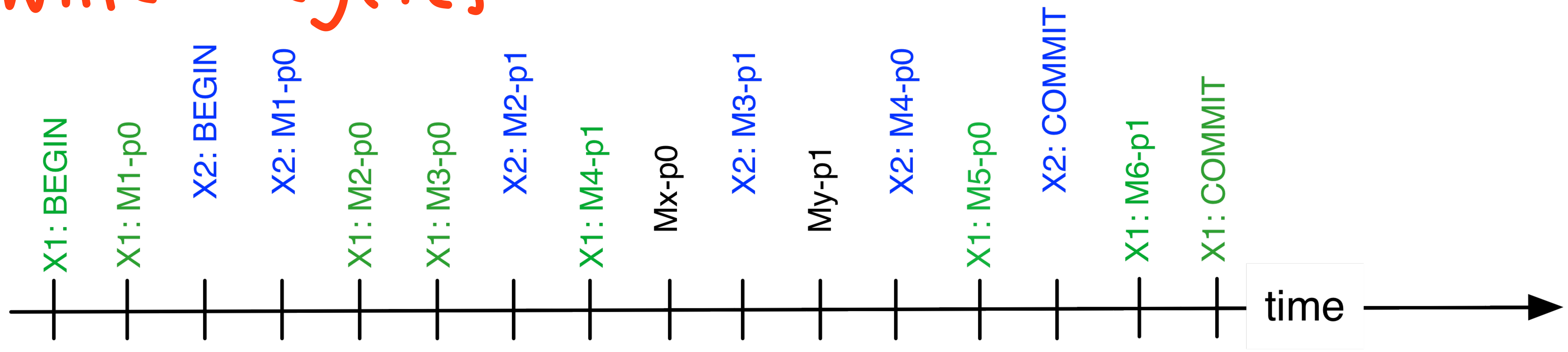
Forbidden by Adya read-uncommitted!

#8 Write Cycles

G0 happens constantly in healthy Redpanda & Kafka clusters...

Forbidden by Adya read-uncommitted!

#8 Write Cycles



X1: BEGIN, X1: M1-p0, X2: BEGIN, X2: M1-p0, X1: M2-p0, X1: M3-p0, X2: M2-p1, X1: M4-p1, Mx-p0, X2: M3-p1, My-p1, X2: M4-p0, X1: M5-p0, X2: COMMIT, X1: M6-p1, X1: COMMIT

time

Commit order: X2 < X0

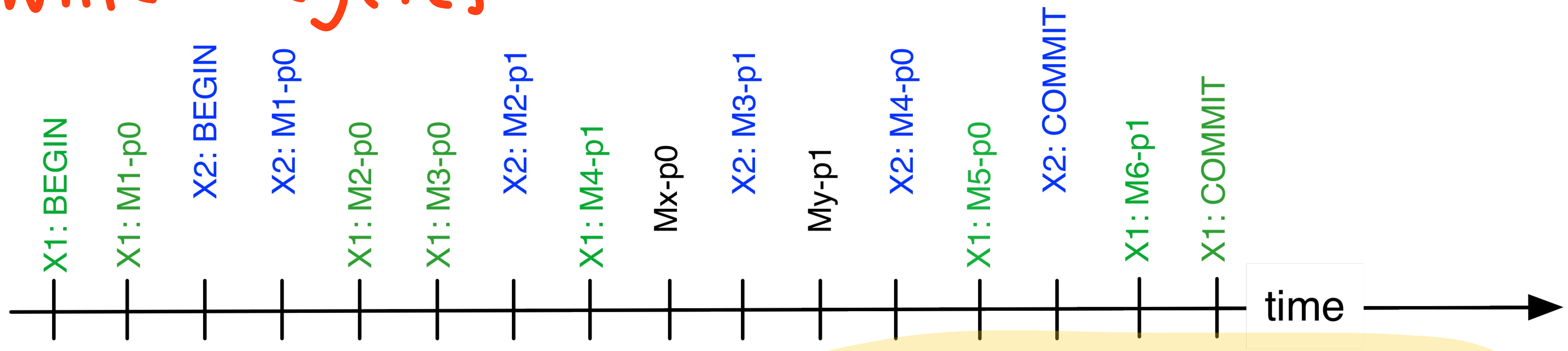
Since X2 is committed first, each partition will expose messages from X2 before X1.

Consumer processing order

p0: Mx-p0, X2: M1-p0, X2: M4-p0, X1: M1-p0, X1: M2-p0, X1: M3-p0, X1: M5-p0

p1: My-p1, X2: M2-p1, X2: M3-p1, X1: M4-p1, X1: M6-p1

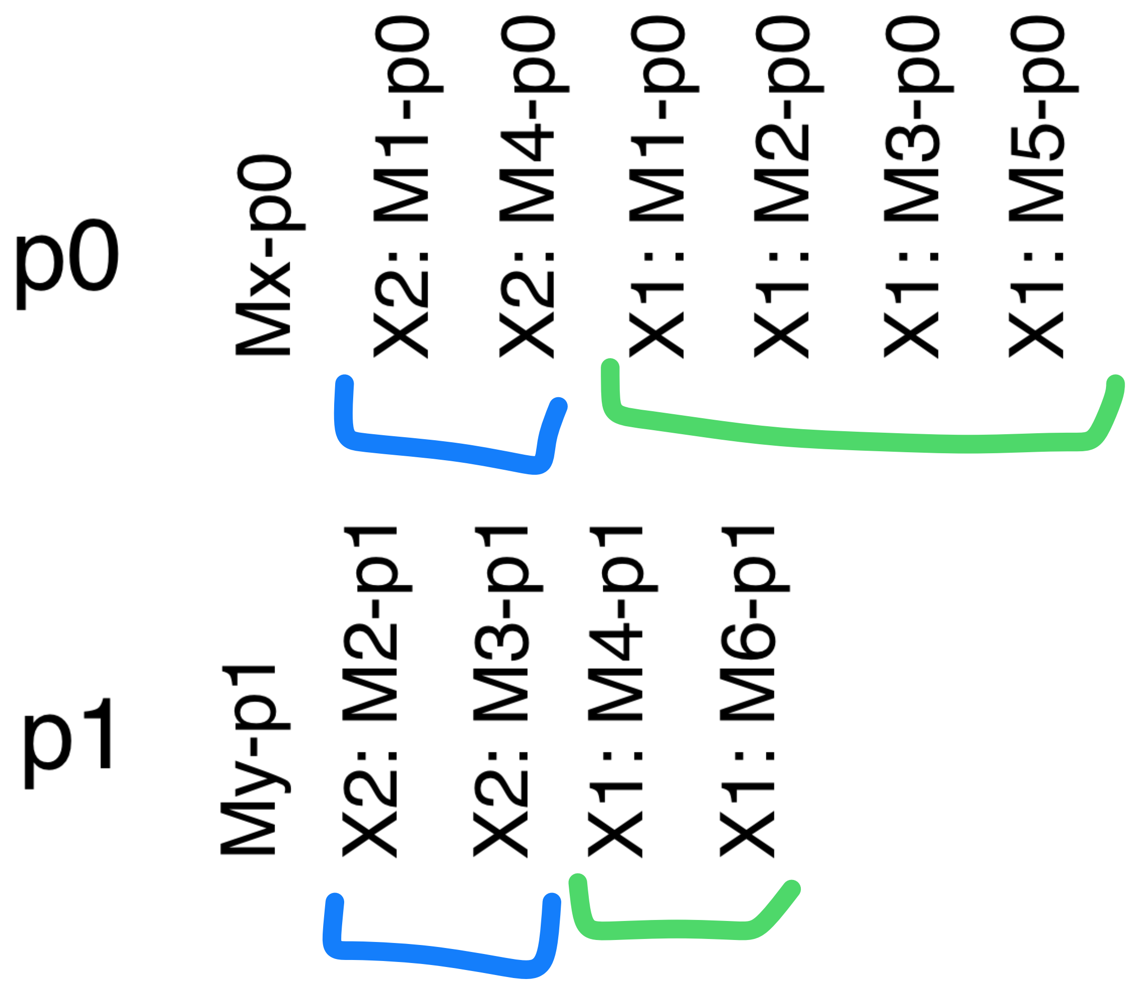
#8 Write Cycles



Commit order: X2 < X0

Since X2 is committed first, each partition will expose messages from X2 before X1.

Consumer processing order



#8 Write Cycles

Liiiiieeees

"We have no strong evidence that applications can benefit from the commit order option, we opted for not implementing it"

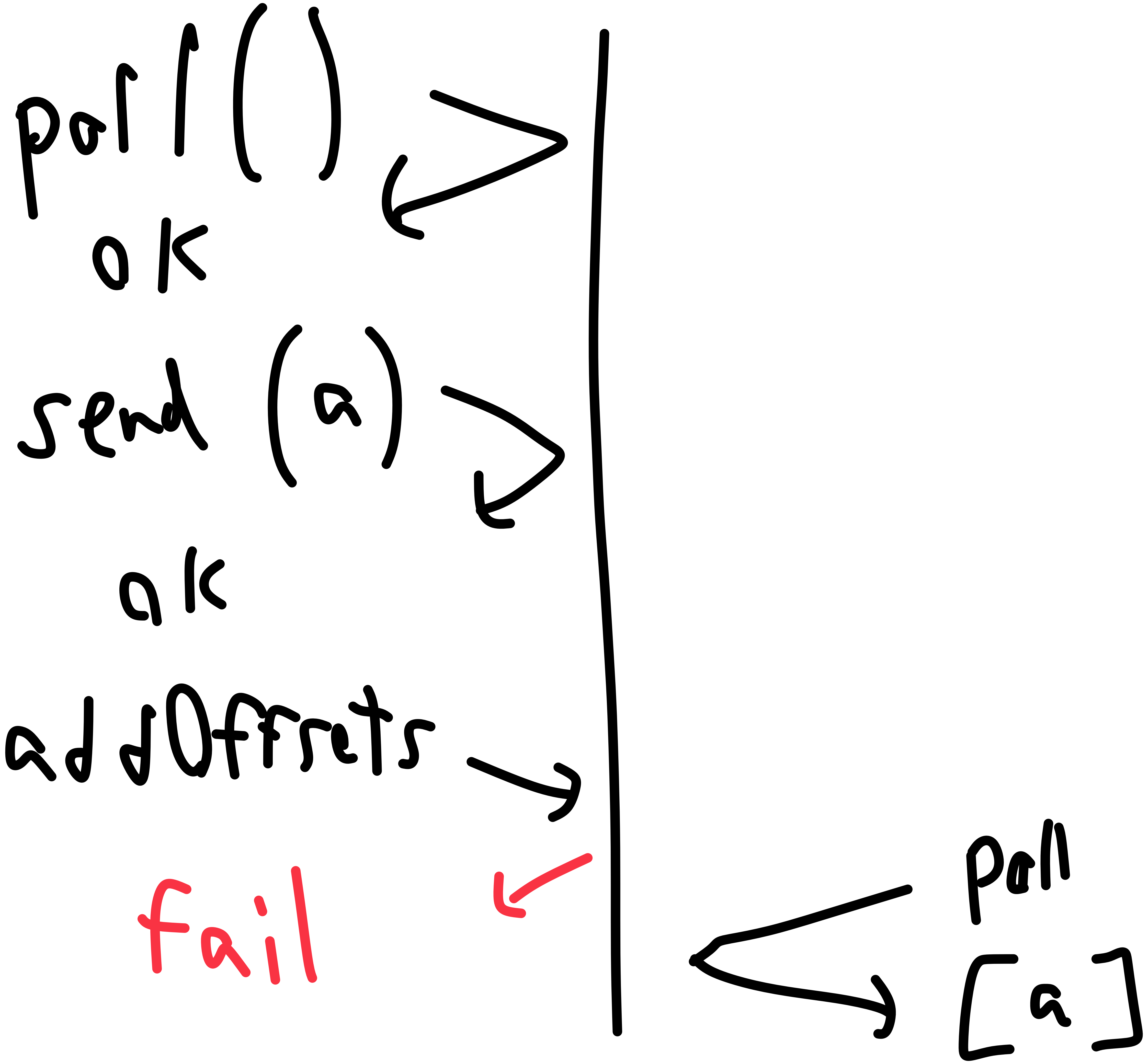
-Exactly Once Delivery and Transactional Messaging in Kafka

#8 Write Cycles

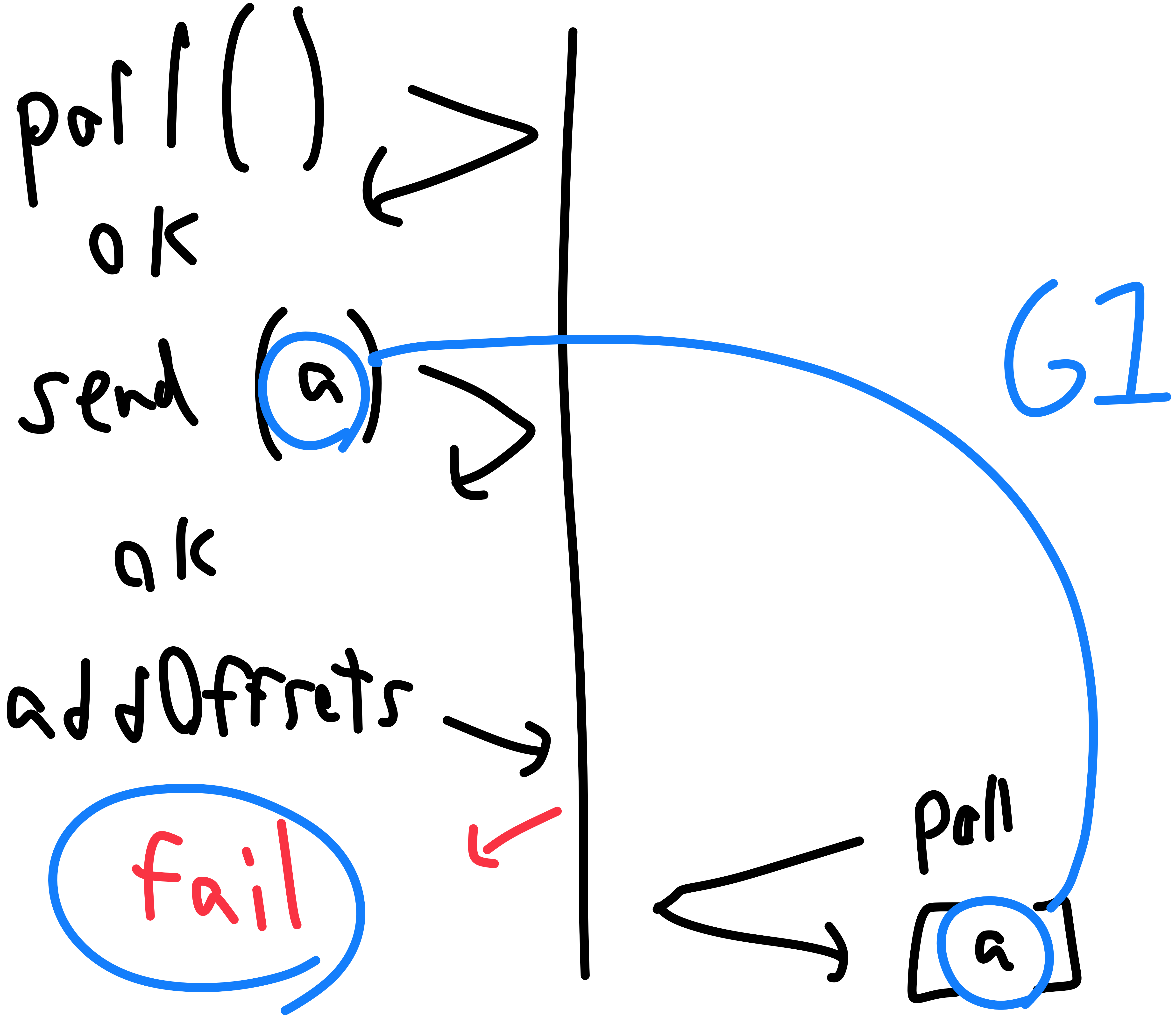
This is just how Kafka/ Redpanda txns work

→ Needs documentation

#3036 Aborted Reads & Circular Information Flow



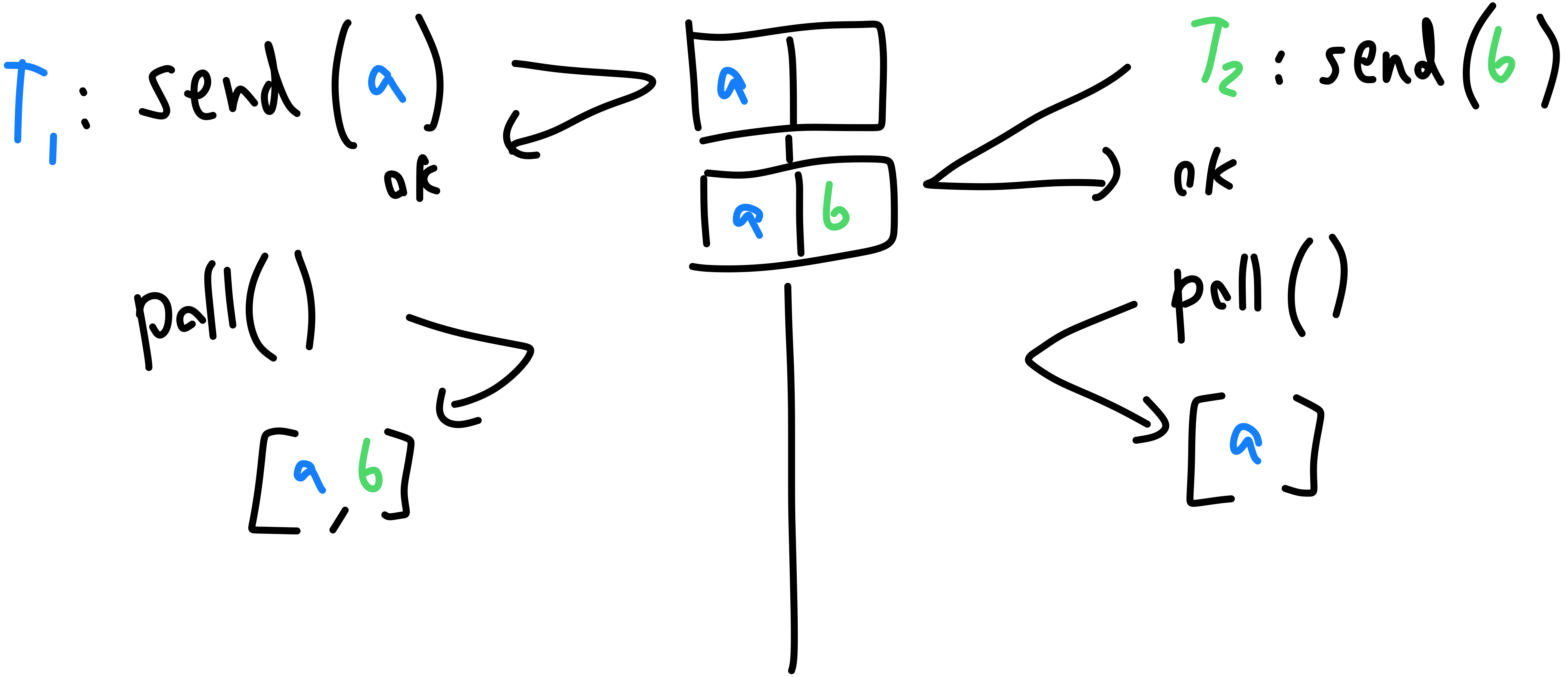
#3036 Aborted Reads & Circular Information Flow



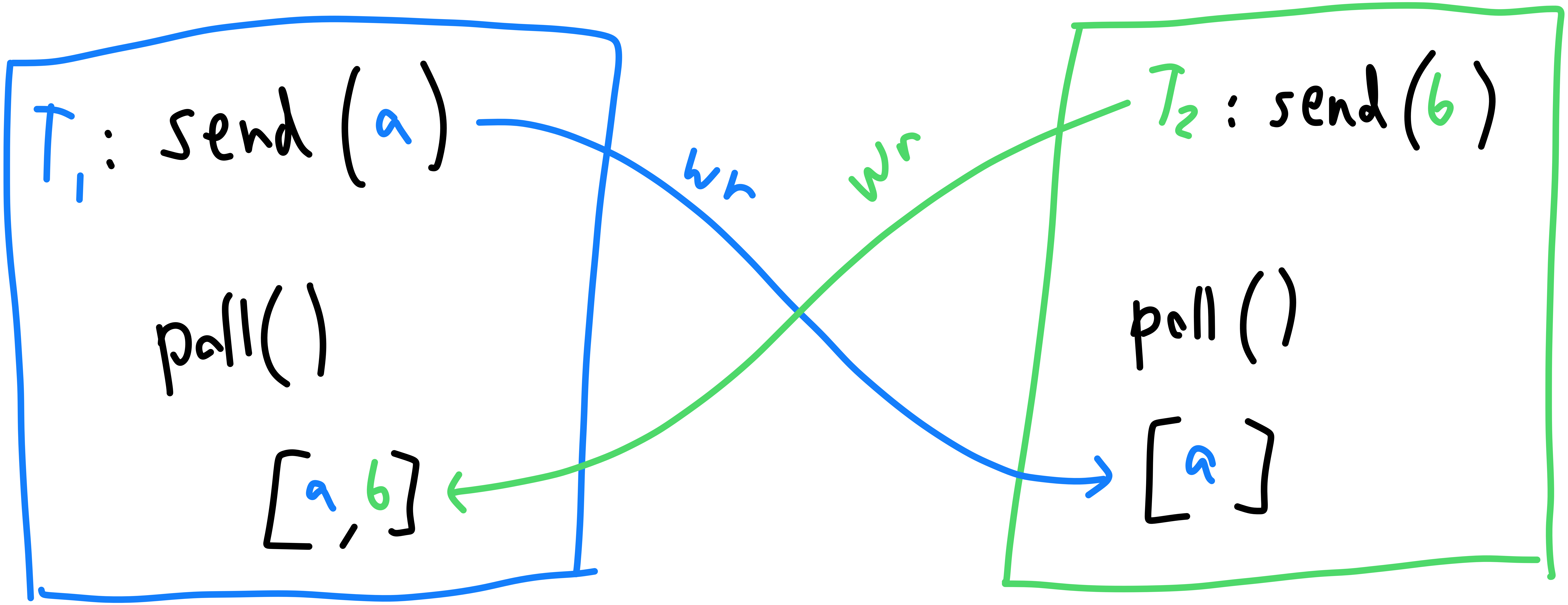
G1a: Aborted Read

Happened often in healthy clusters!

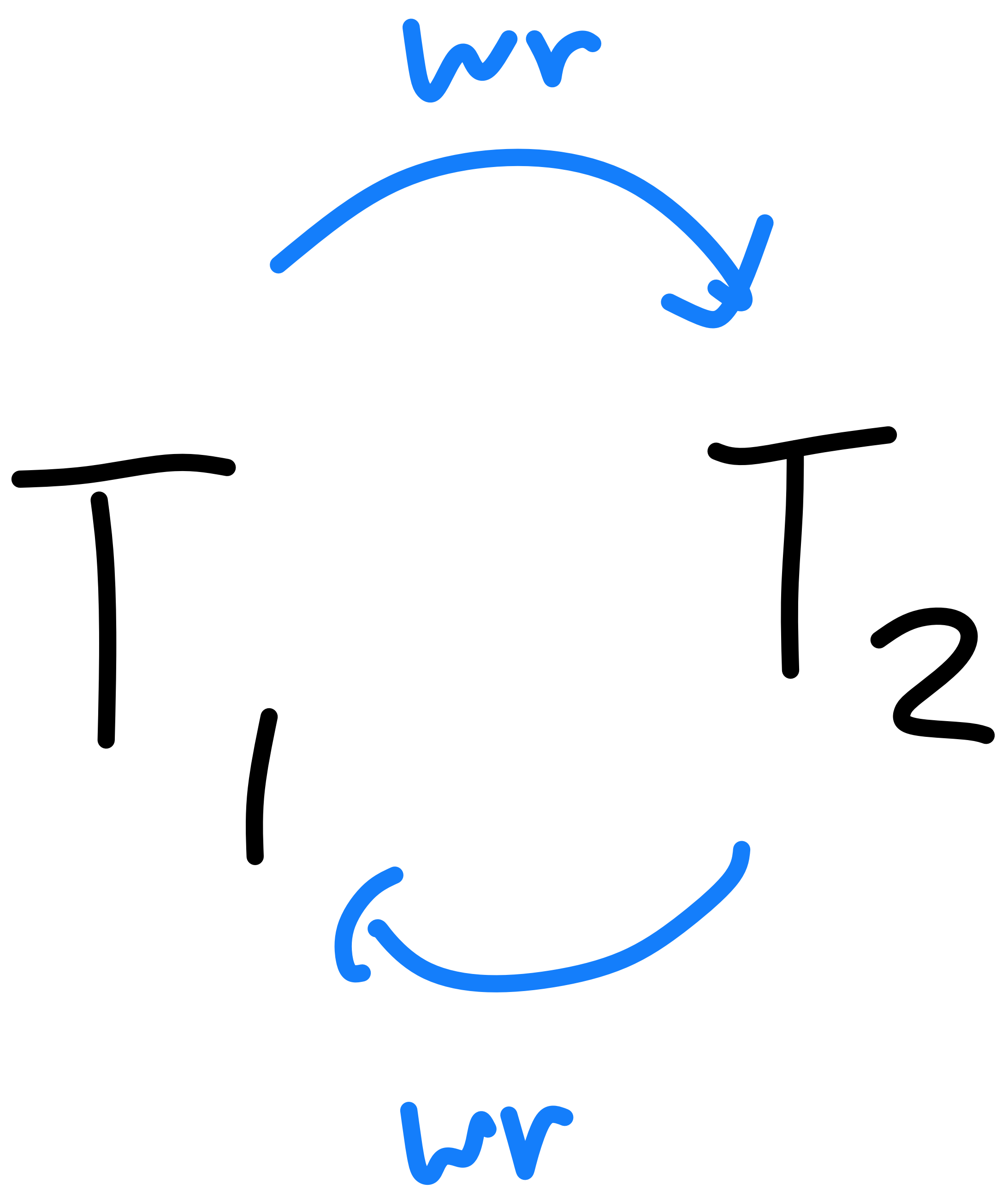
#3036 Aborted Reads & Circular Information Flow



#3036 Aborted Reads & Circular Information Flow



#3036 Aborted Reads & Circular Information Flow



G1c: Circular Information Flow

Illegal in Adya Read Committed!

#3036 Aborted Reads & Circular Information Flow

- Off-by-one error allowed LSO to advance beyond committed offsets
- #3232 accidentally aborted more txns than necessary

Fixed in 21.10.2

#3616-6 Lost Transactional Writes

```
beginTxn()
send(a)       -> offset 2
commitTxn() -> OK
```

#3616-6 Last Transactional Writes

beginTxn()

send(a) → offset 2

commitTxn() → ok

poll() →

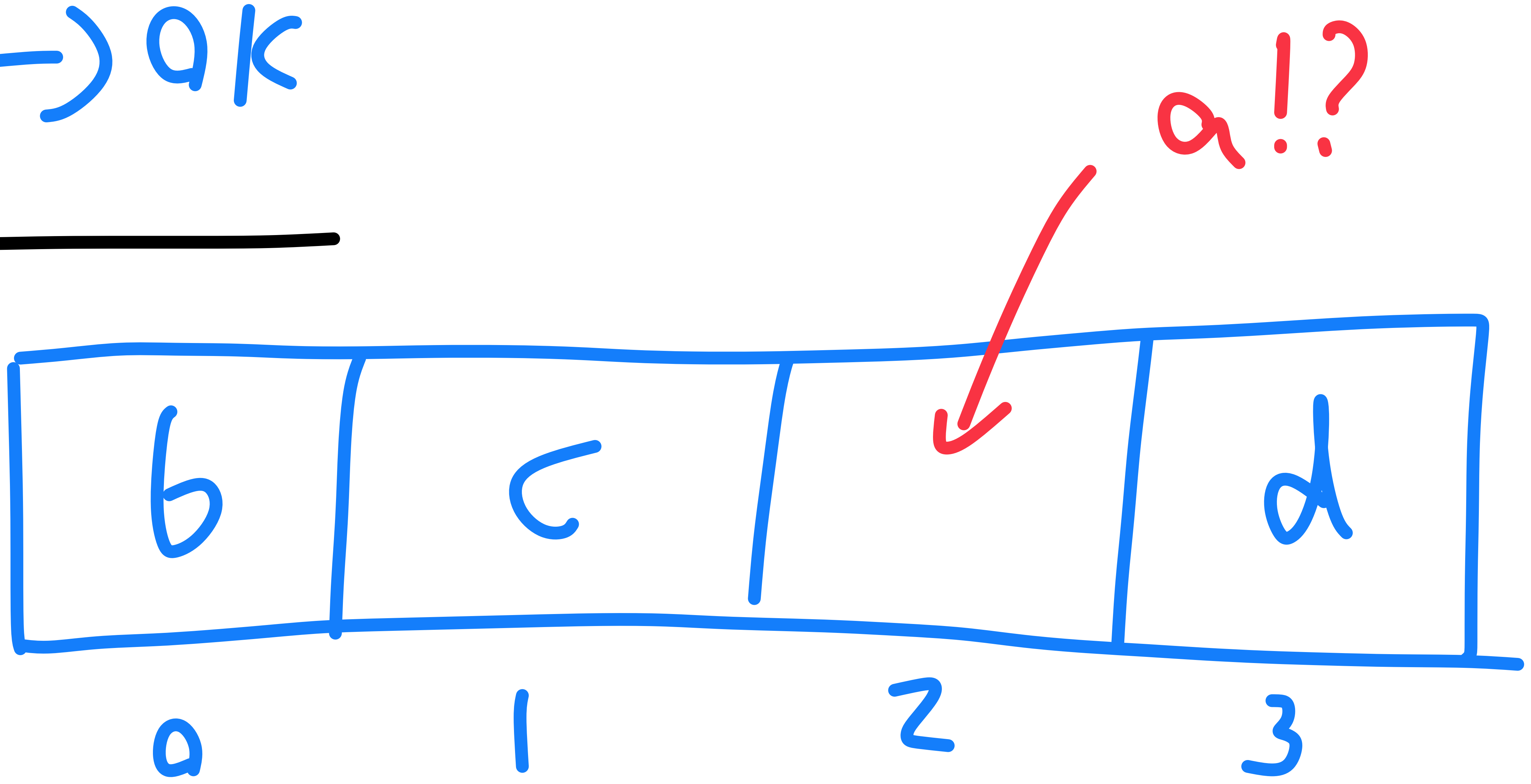
b	c		d
0	1	2	3

#3616-6 Last Transactional Writes

```
beginTxn()
send(a)       -> offset 2
commitTxn() -> OK
```

---

poll() ->



#3616-6 Lost Transactional Writes

Happened in healthy clusters

on 21.11.2

#3616-6 Lost Transactional Writes

- Node could lose, then regain leadership during critical section

#3616-6 Lost Transactional Writes

Fixed January 21, 2022

↳ Released in 21.11.15

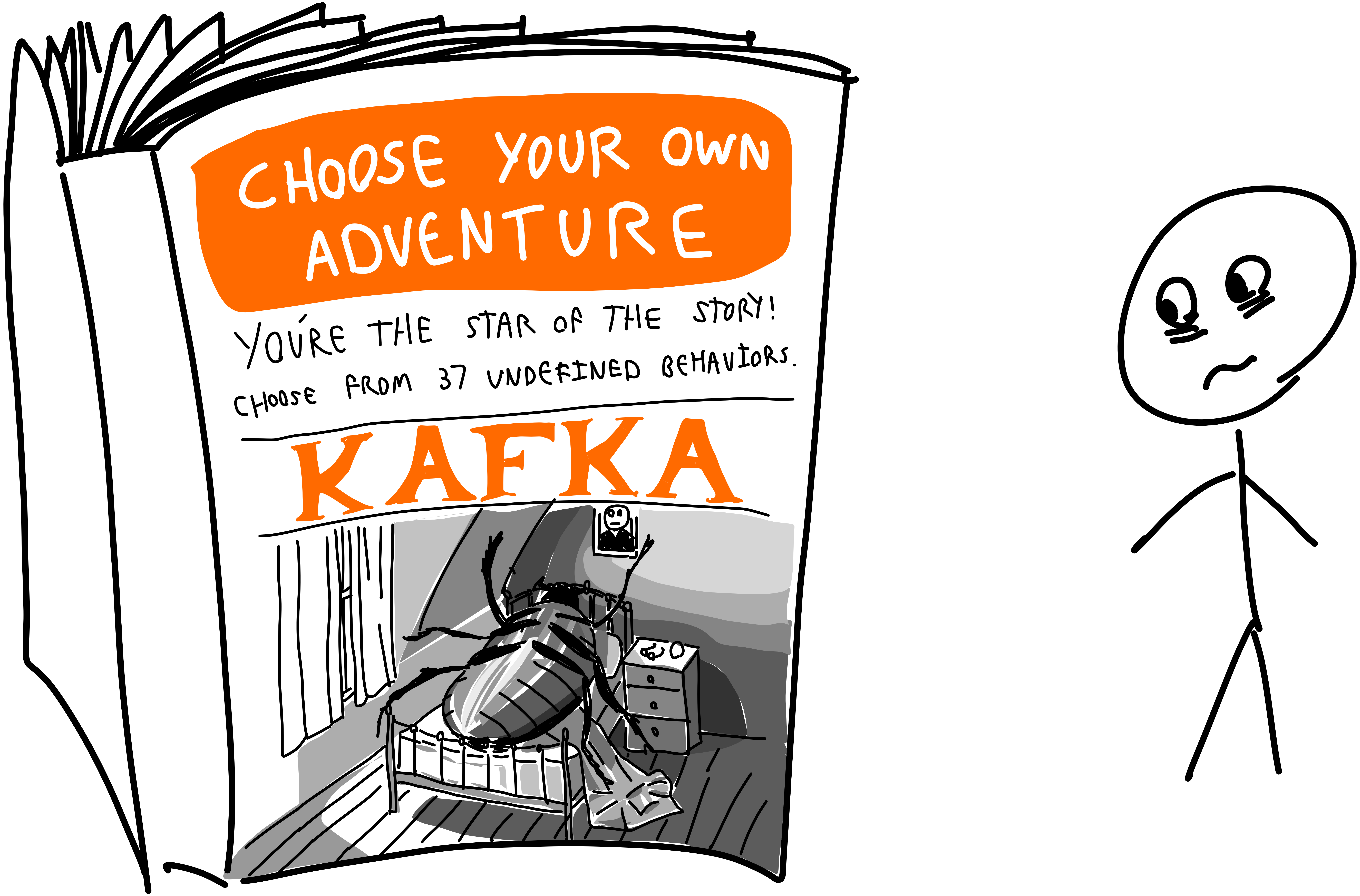
№	Summary	Event Required	Fixed In
1	Duplicate writes by default	Pause, crash, or partition	22.1.1*
3039	Duplicate writes with idempotence	Pause, crash, or partition	21.10.3
3335	Assert failure deallocating partitions	Membership change	Unresolved
3336	Assert failure involving partition IDs	Crash	22.1.1*
3003	Inconsistent offsets	Crash or partition	21.10.3
6	Lost/stale messages	Pause, crash, or partition	Unresolved
KAFKA-13574	Aborted read with NotLeaderOrFollower	Membership change & pause	Unresolved
8	Write cycles	None	Unresolved
3036	Aborted read & circular information flow	None	21.10.2
10	Internal non-monotonic polls	None	Unresolved
3616-a	Aborted read with InvalidTxnState	Pause or crash	21.11.15
3616-b	Lost transactional writes	None	21.11.15

Most frequent issues resolved in 21.10.3 — but some serious problems in 21.11.2

Txn Documentation

- Absent
- Incomplete
- Vague
- Misleading
- Wrong

- Largely explained in terms of implementation
- Go read 67 pages of design doc?


CHOOSE YOUR OWN ADVENTURE
YOU'RE THE STAR OF THE STORY!
CHOOSE FROM 37 UNDEFINED BEHAVIORS.
KAFKA

Discussion

Simple tests → Surprising Results

1. End-to-end tests
2. With fault injection
3. Generative/property tests
4. Sophisticated safety checkers

→ Elle

Transactions!

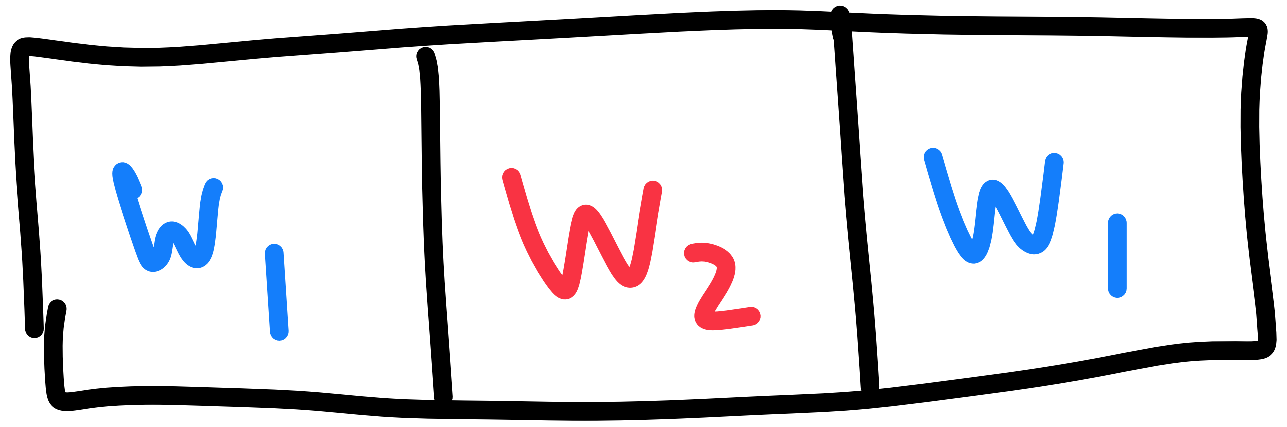
Not just SQL any more!

New Domains, New Expectations?

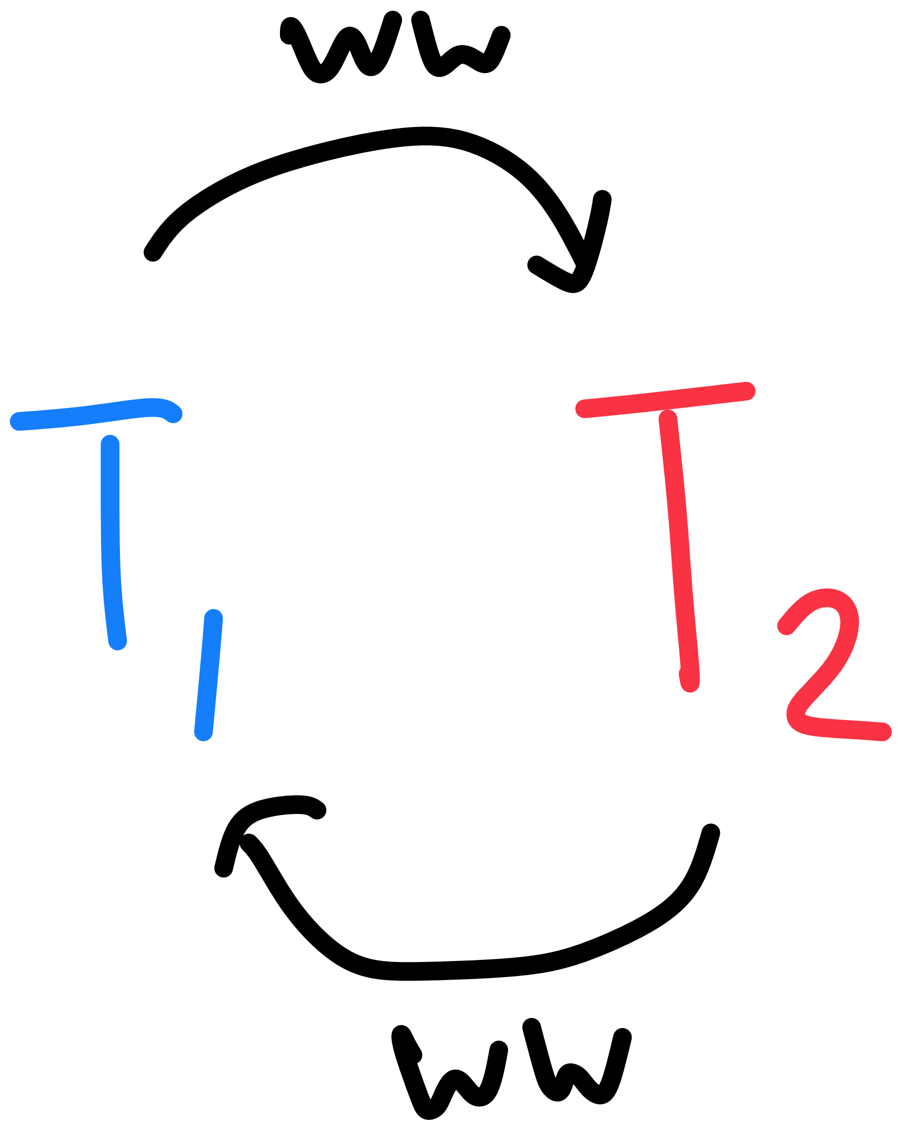
- Stream Processing
- Permissionless Ledgers

  "Blockchain"

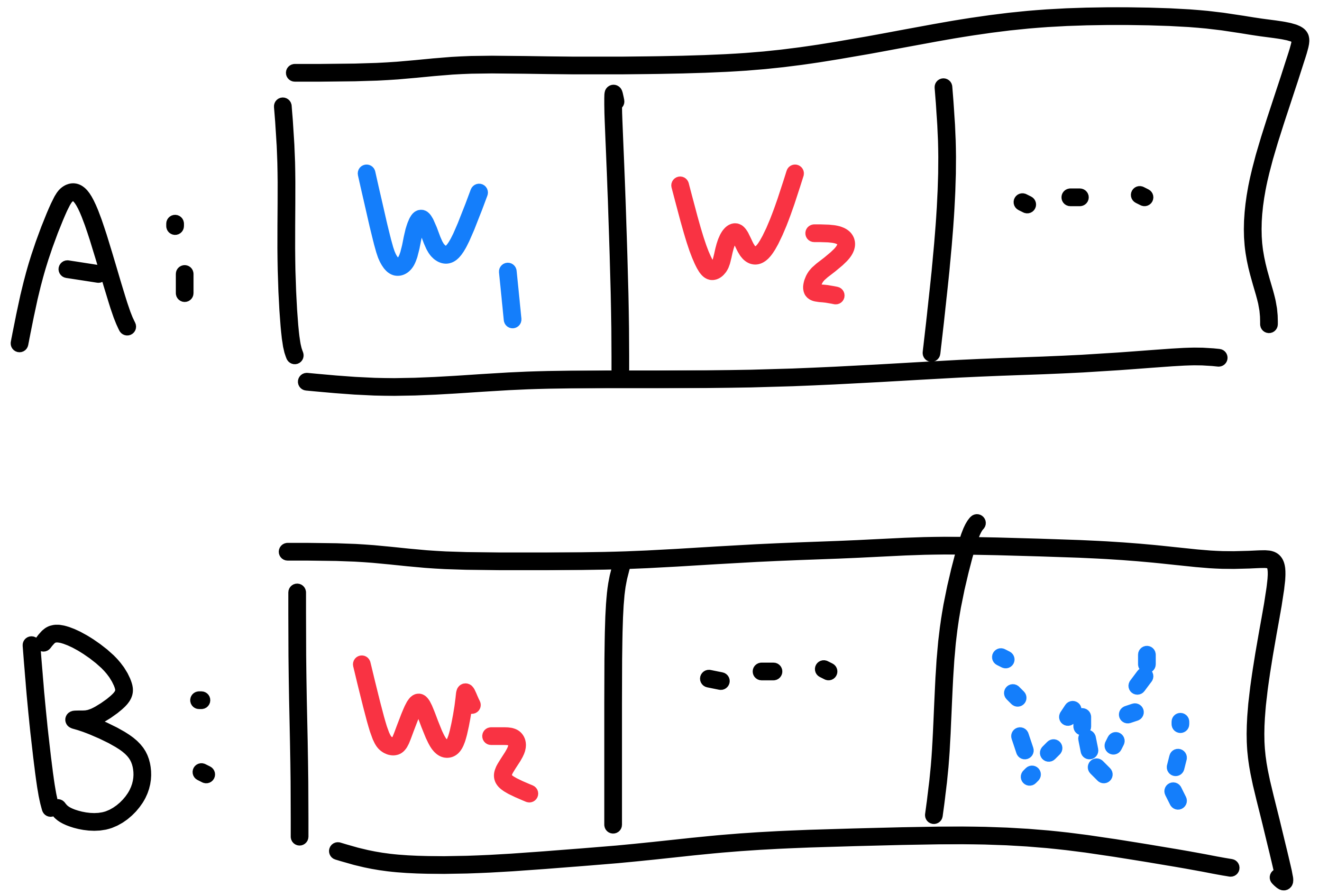
Analogues from Adya et al?



Redpanda

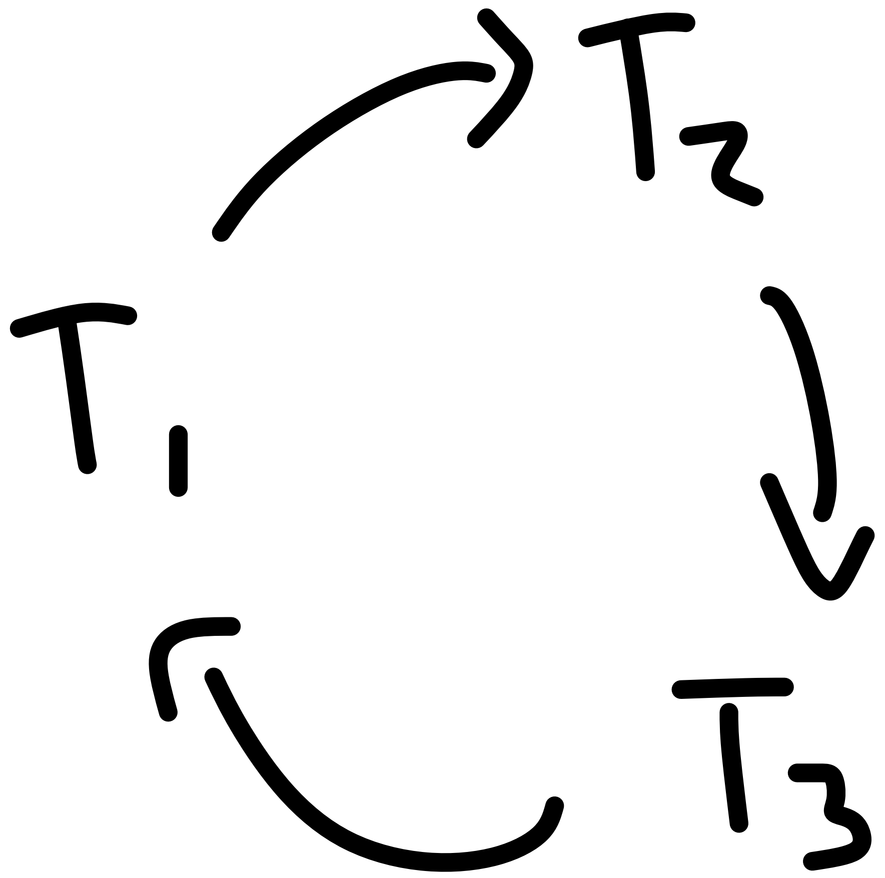


Adya G0

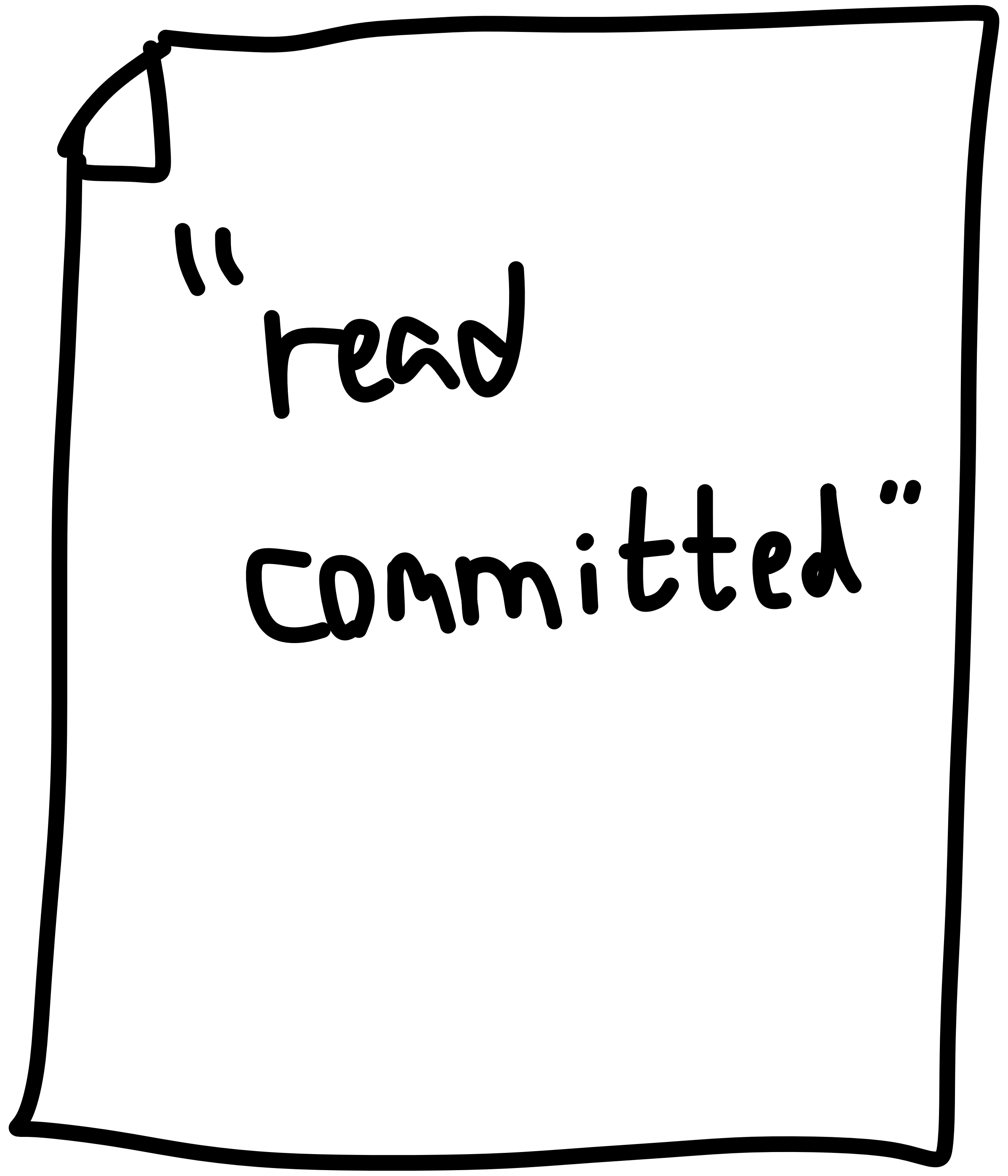


Radix

Formalisms



Docs



Intuition

Formalisms Docs Intuition

Meaningful
Invariant
Violations

"Real-World
Impact"

Users

- Read literature
- Read docs carefully
- Check application/use case safety
- Share expectations & invariant violations

Vendors

- Read literature & user reports
- Formalize safety model
- Communicate it through docs!
- Test against invariants!

Academics

- User research → new, abstract formalisms
  - "Adya for Kafka?" "... for blockchain?"
- Work on new formalisms <u>is</u> being done!
- How do we connect w/ vendors & users?

Thanks

Funded by

Redpanda Data

Radix Tokens (Jersey) Limited

- Russell Harvey
- Matthew Hine
- Joshua Primera
- Shambu Pujar
- Piers Ridyard
- Sergiy Yevtushenko

- Camilo Aguilar
- Travis Bischel
- Bob Dever
- Juan Castilla
- Alexander Gallego
- Dhruv Gupta
- Michal Maslanka
- Denis Rystsov
- John Spray
- Coral Waters
- David Wang
- Noah Watkins
- Allison Daly

https://jepsen.io