Jeppsen

a fsyncing feeling

Kyle Kingsbury

@aphyr

I break databases!

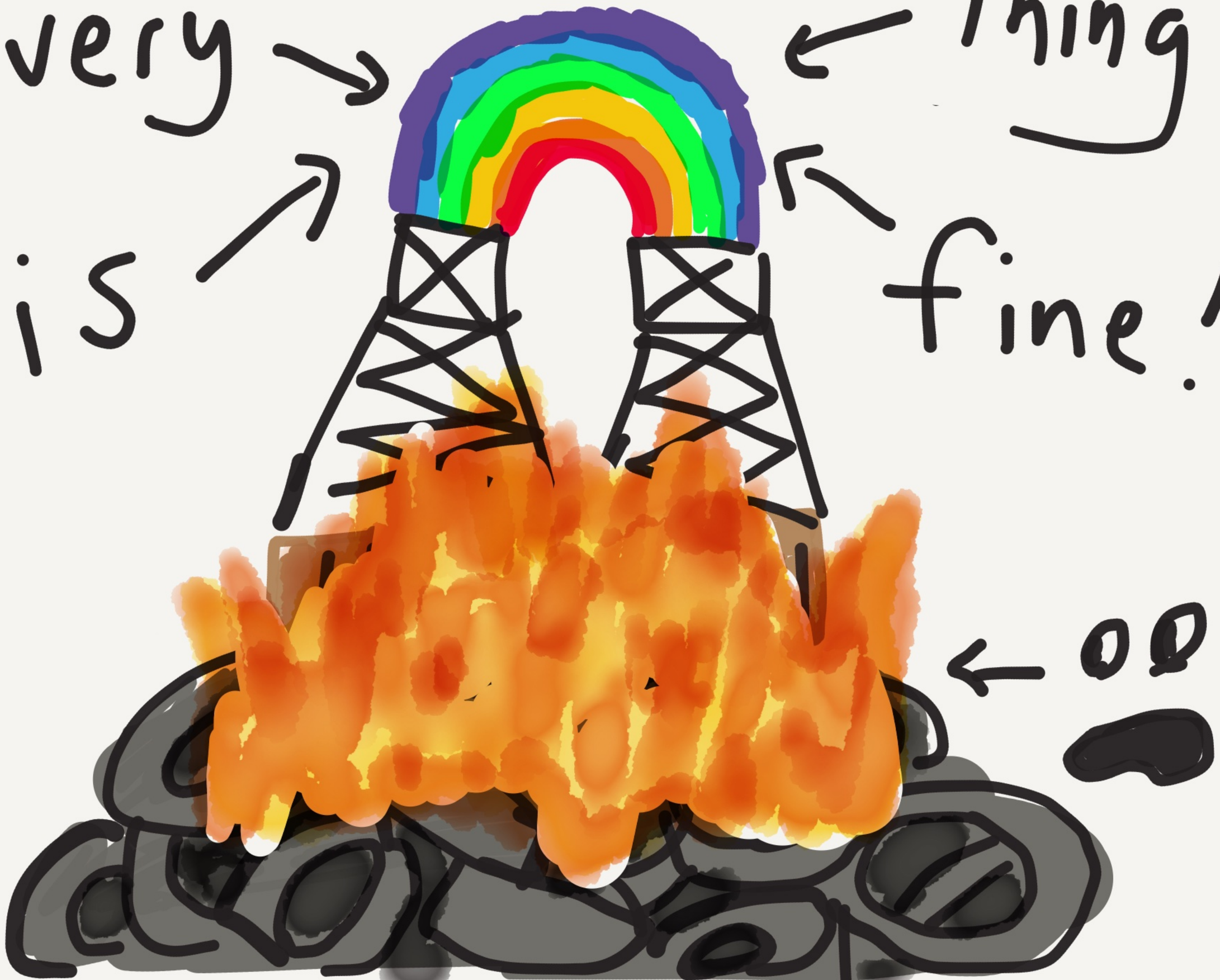
Public API {

} API code

Ruby {

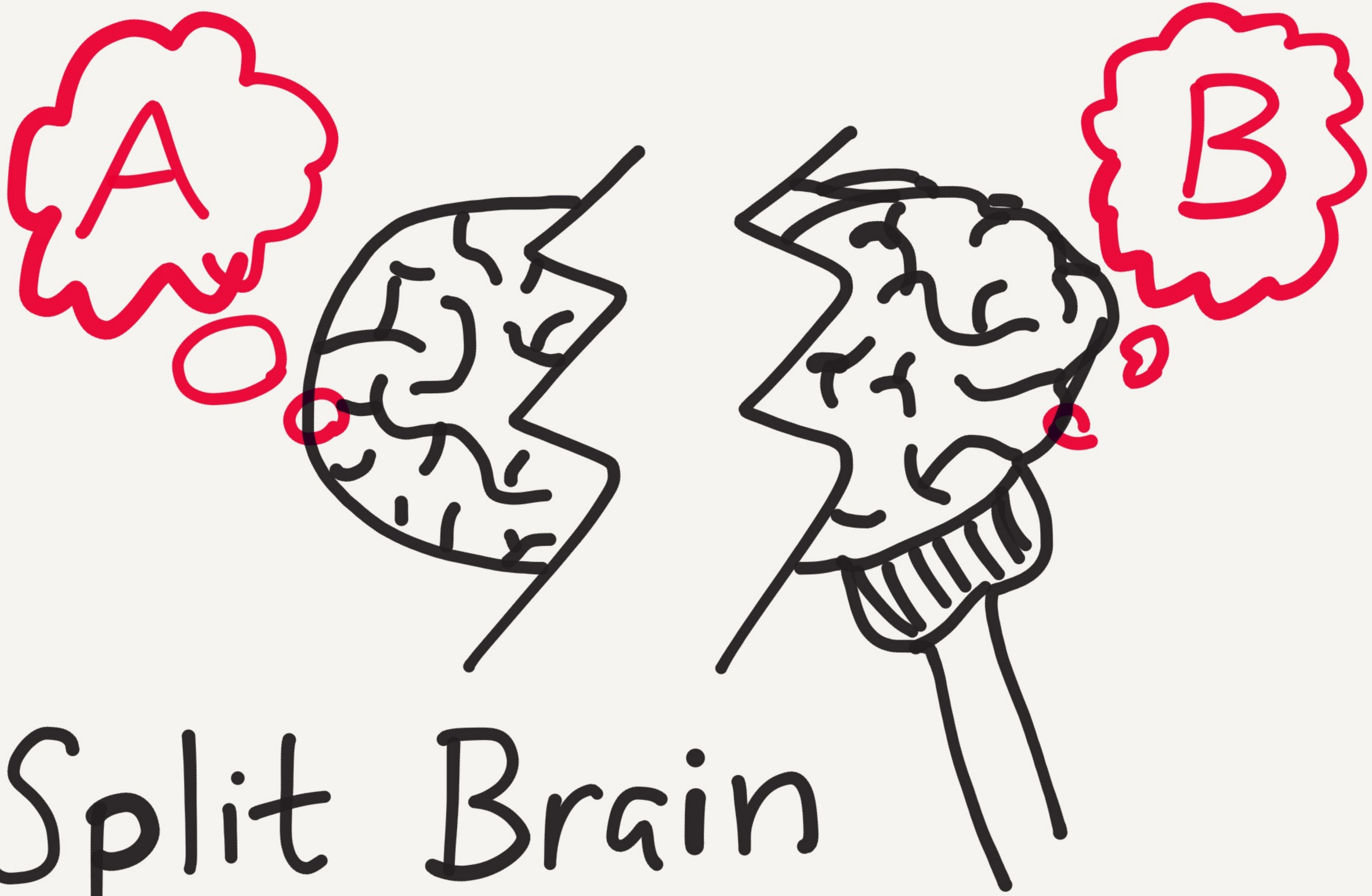
DBs

Every thing is fine!



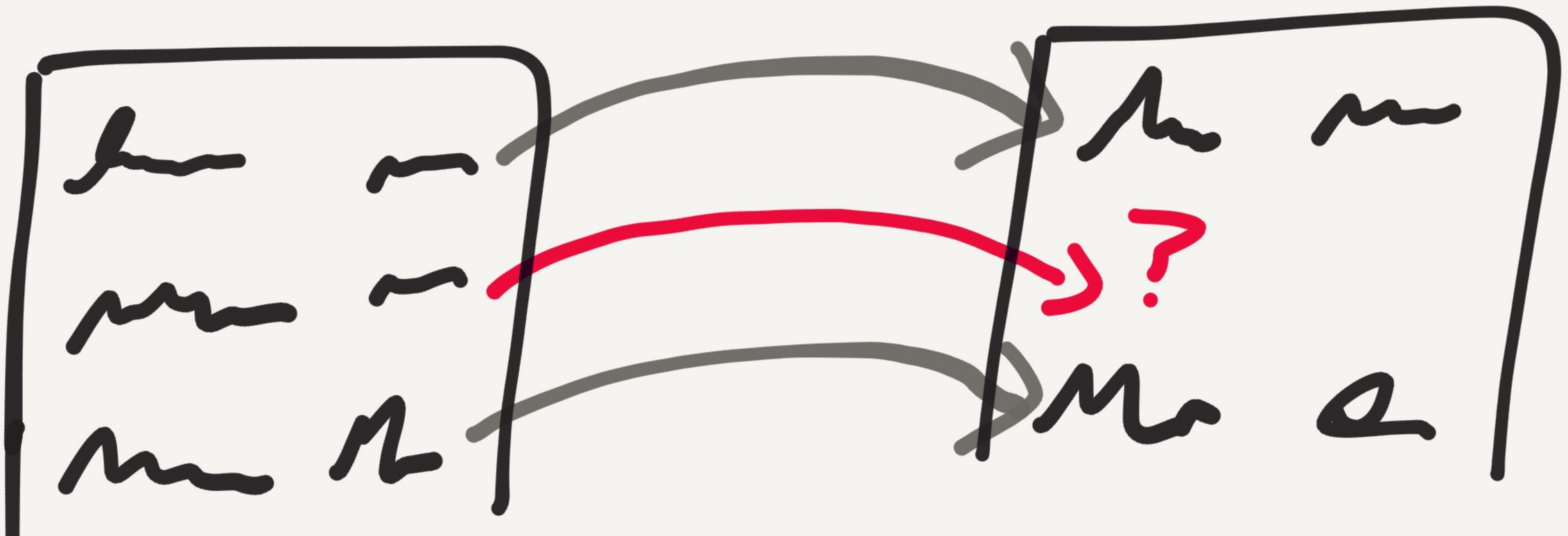
Databases!
Queues!
Discovery!

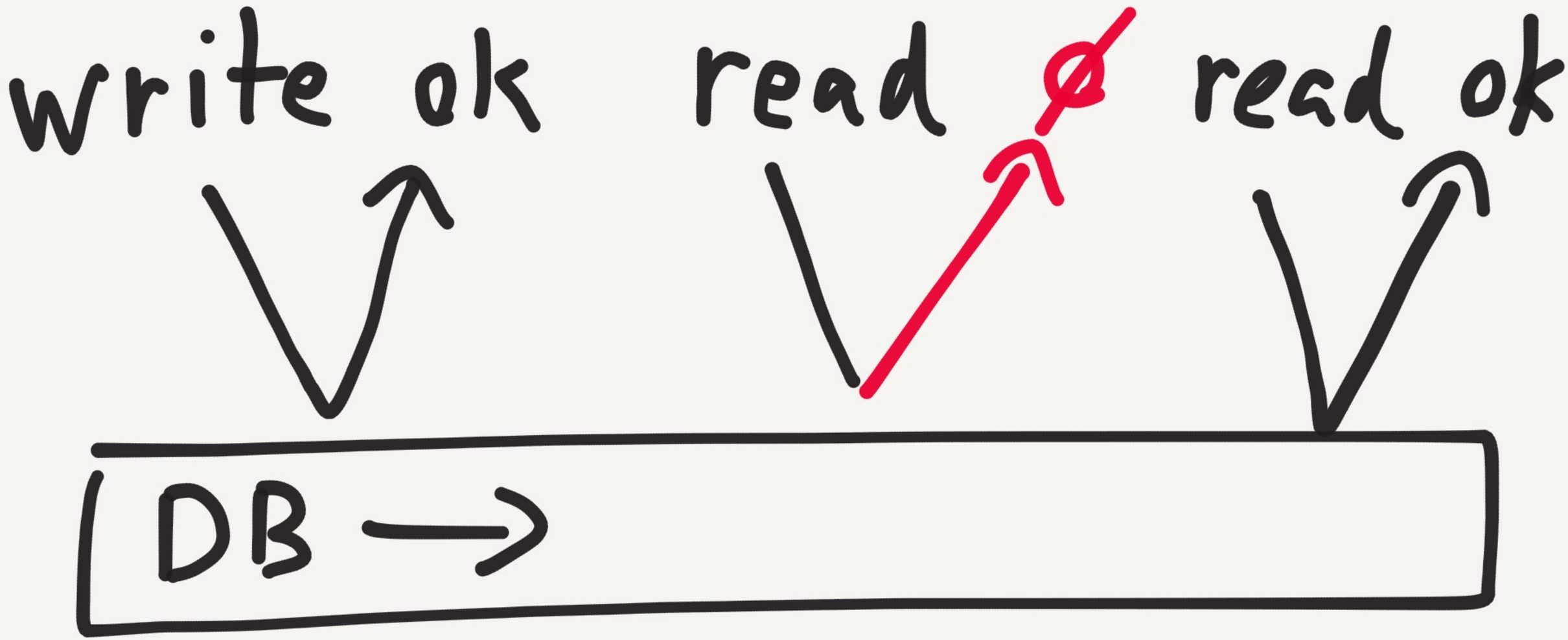
THE HORROR



Split Brain

Broken Foreign Keys

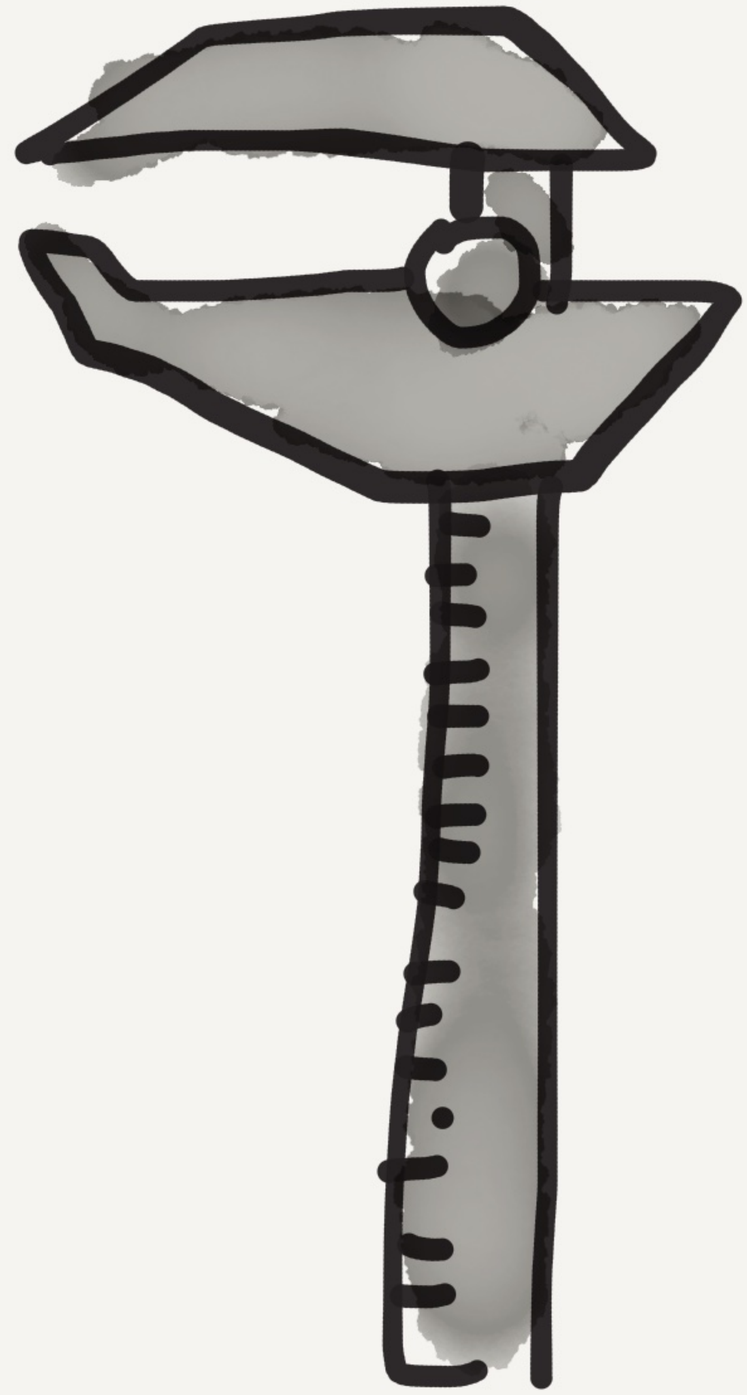




Anomalies

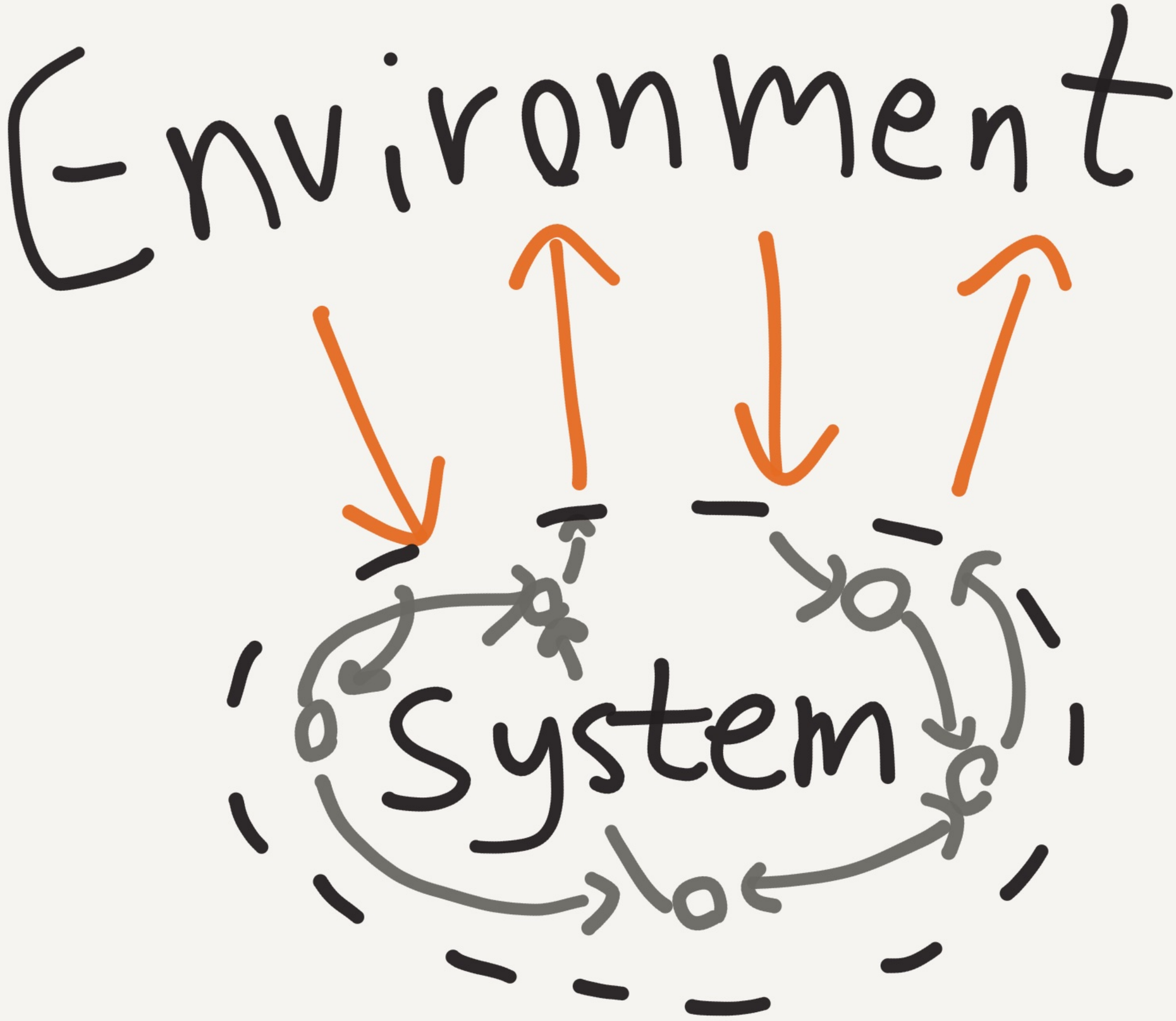
How do you
know if a
system is safe?

Measure your Systems

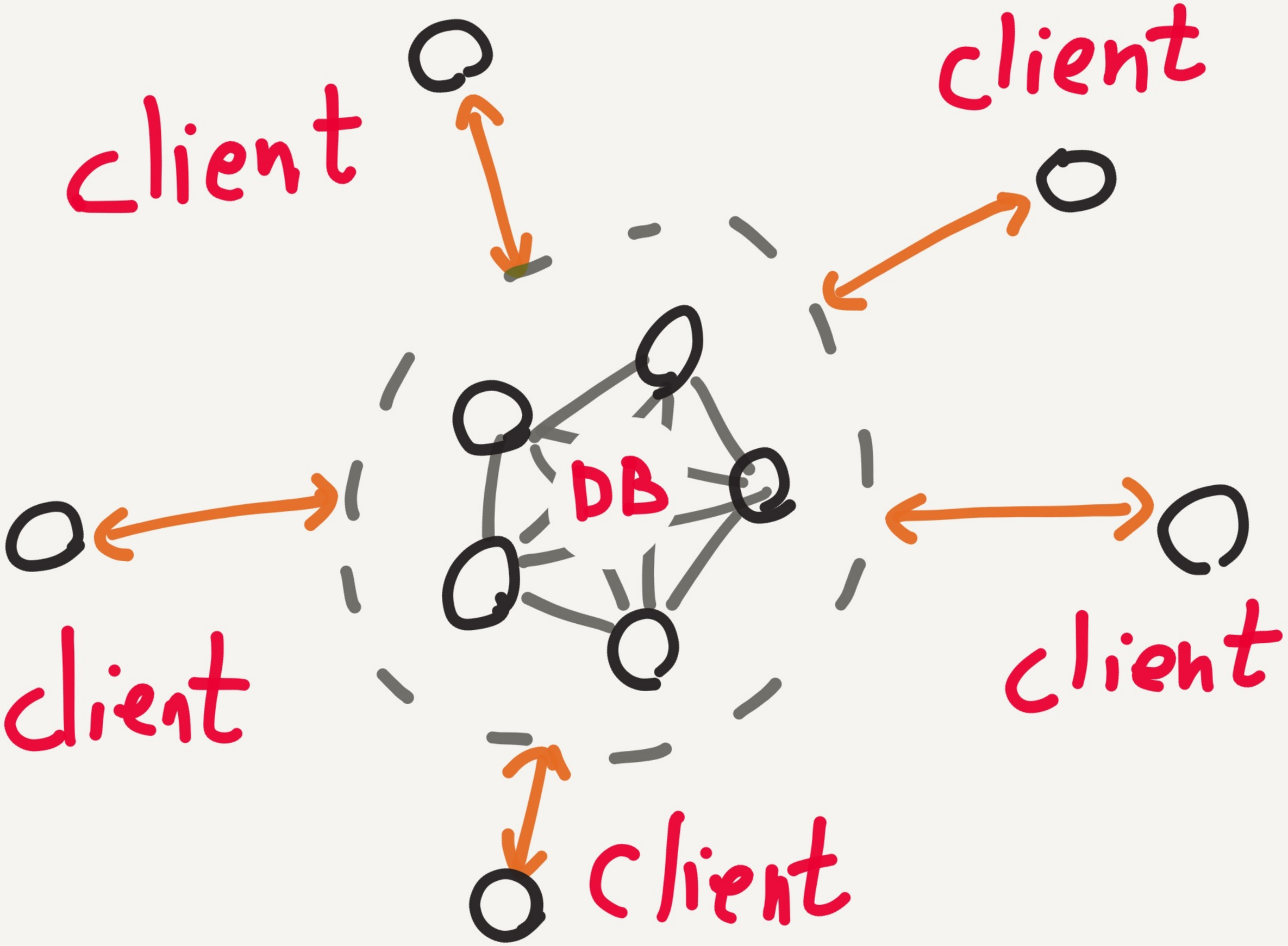


Jepsen

github.com/aphyr/jepsen


Environment
System

-INVARIANTS-

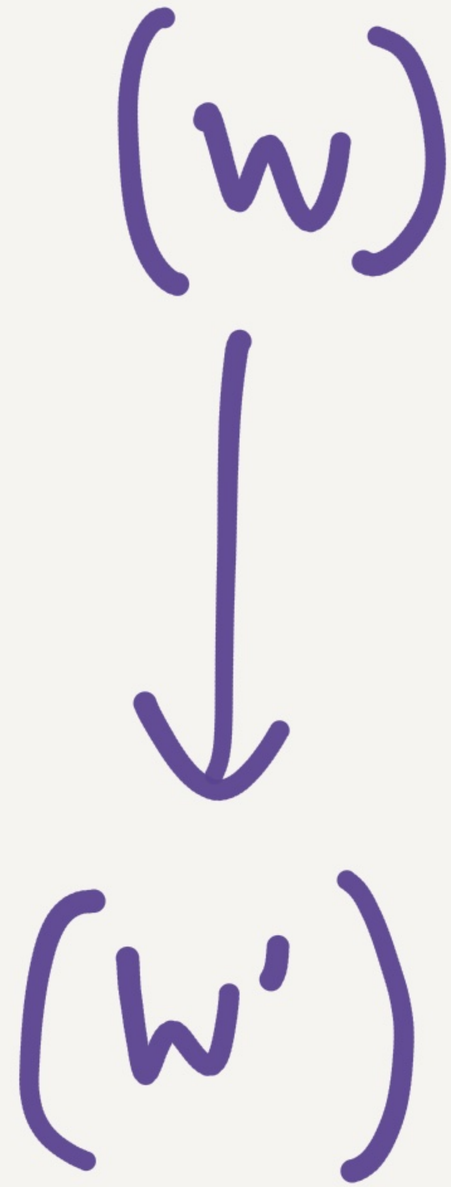

client
client
DB
client
client
client

client: w—w' r—r' w—w'

DB:

client : w —— w' w —— - - -

Clients Generate random operations (w) and apply them to the system



invoke ok

invoke fail

invoke ? ? info ? ? ?

invoke ok

invoke fail

invoke ok

invoke info

invoke ok

invoke fail

invoke ok

invoke info

invoke ok

invoke fail

invoke ok

invoke info

invoke ok

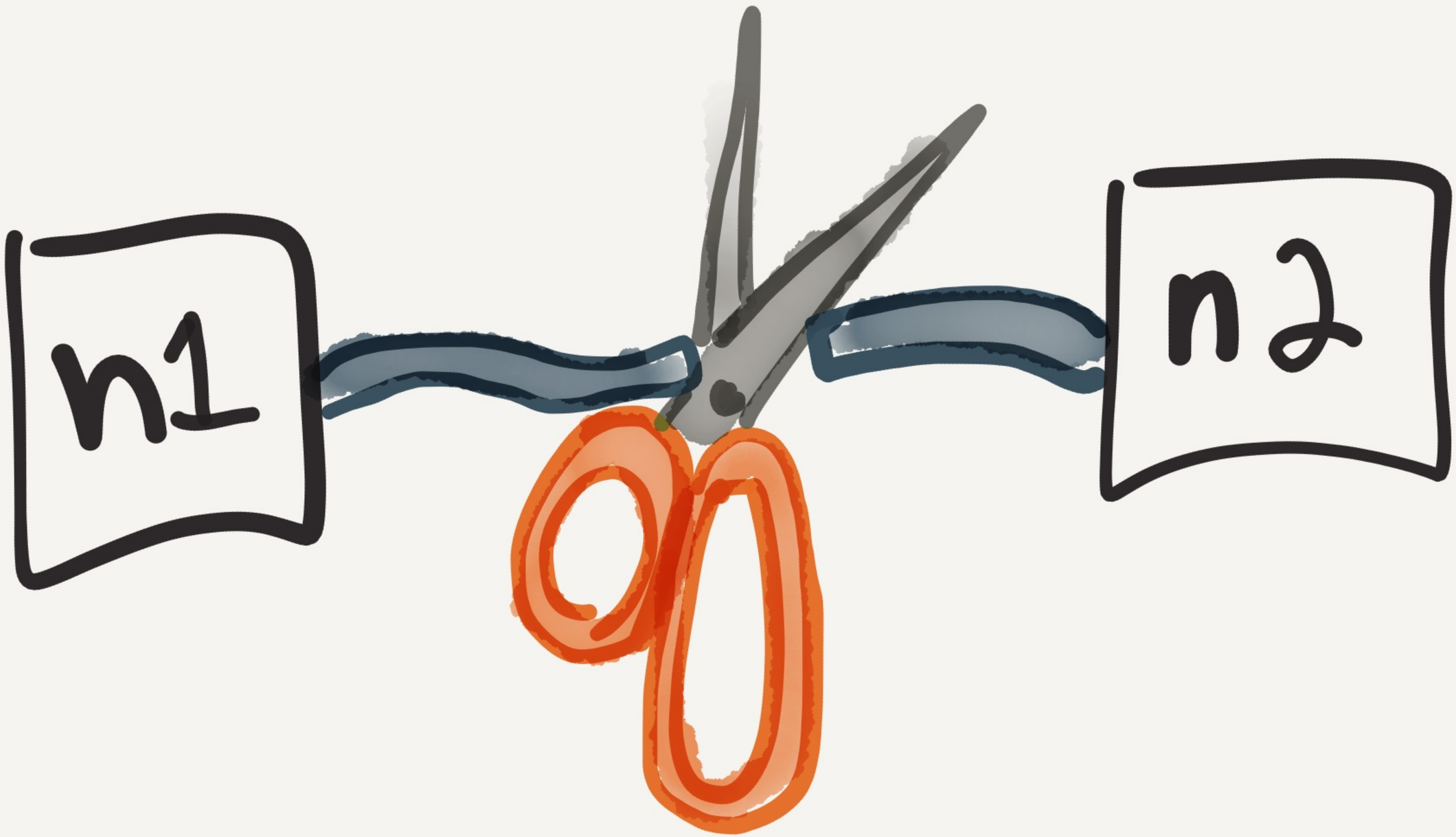
invoke fail

invoke ok

invoke ok

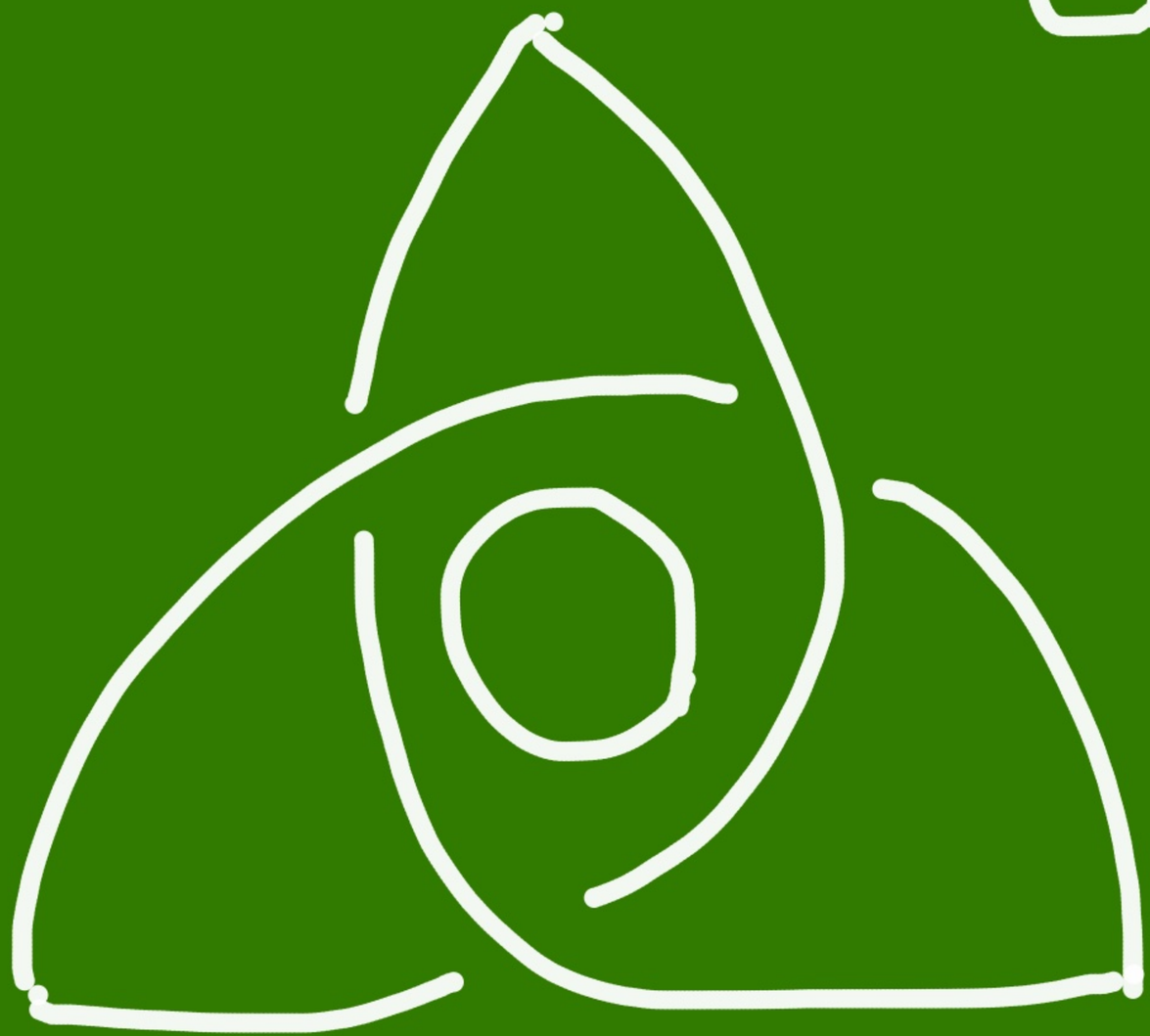
invoke info

1. Generate ops
2. Record history
3. Verify history is consistent w/ model



Partitions!

0.10.2

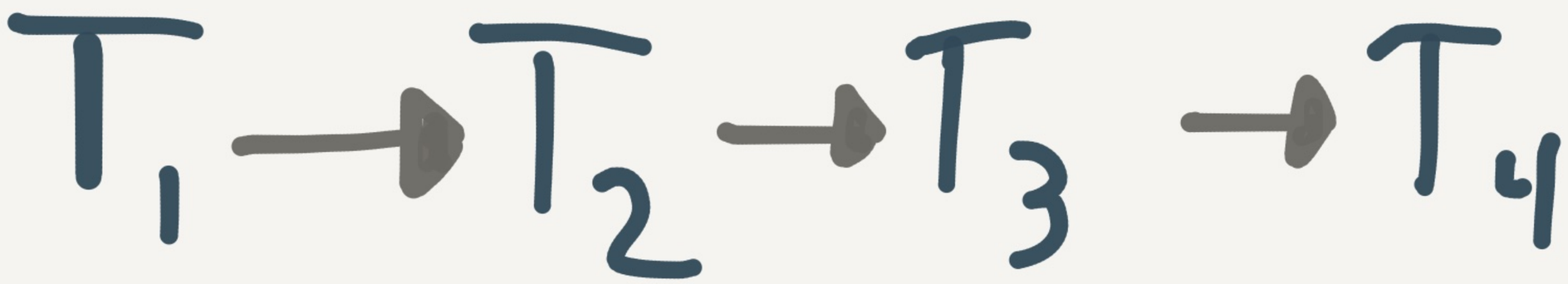


Tendermint

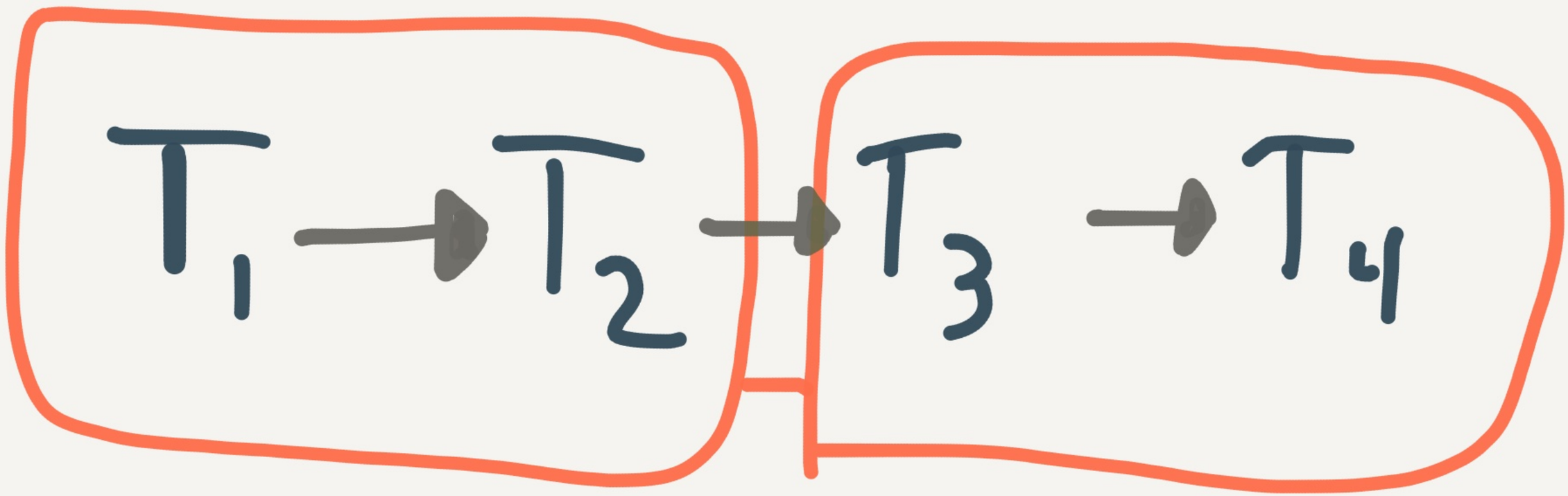
- Transaction replication
- Pluggable FSMs
- Byzantine fault tolerant

"Raft Meets Blockchain"

Like e.g. Raft, forms a linearizable order of txns



Like Raft, provides a linearizable order of txns



blockchain!

Byzantine Fault Tolerant!

shun!

shun!

shun!

shun!

Pluggable State Machines

validator

validator

state

state

merkle eyes

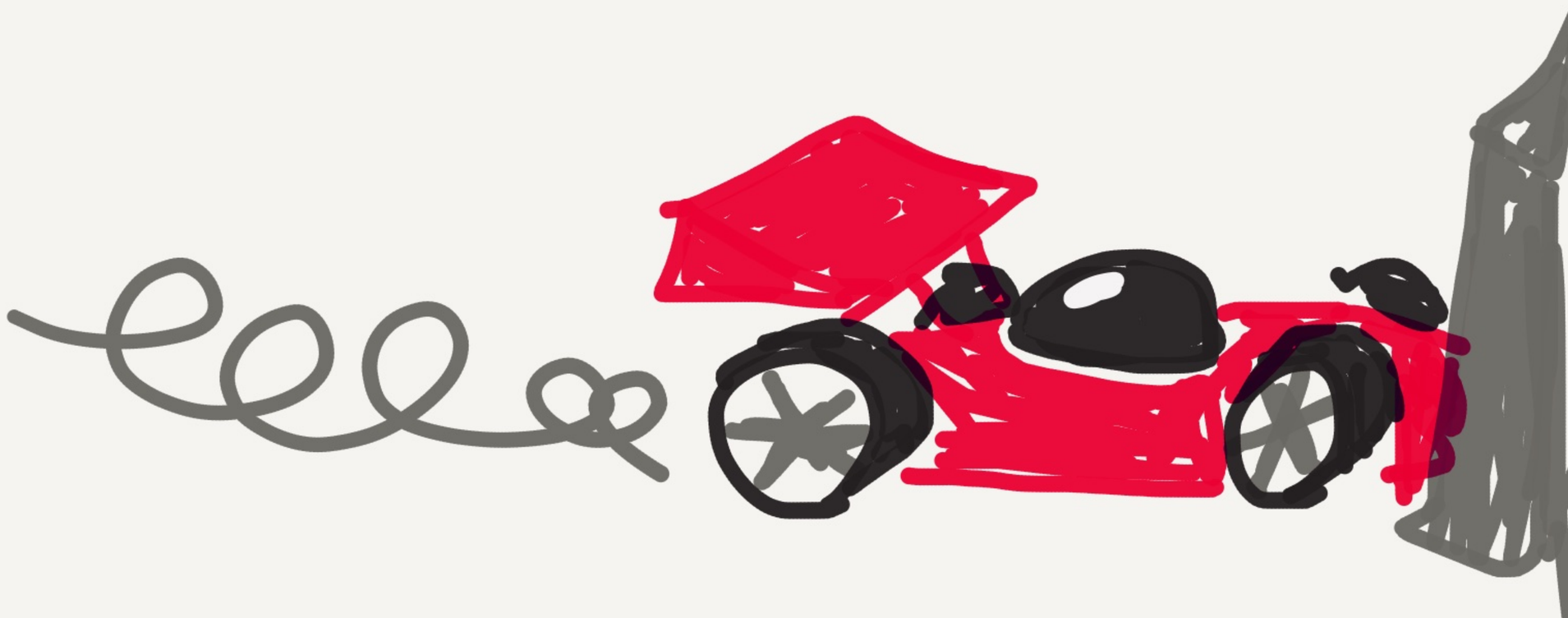
cas-register

- read(2)
- write(5)
- cas(1,4)

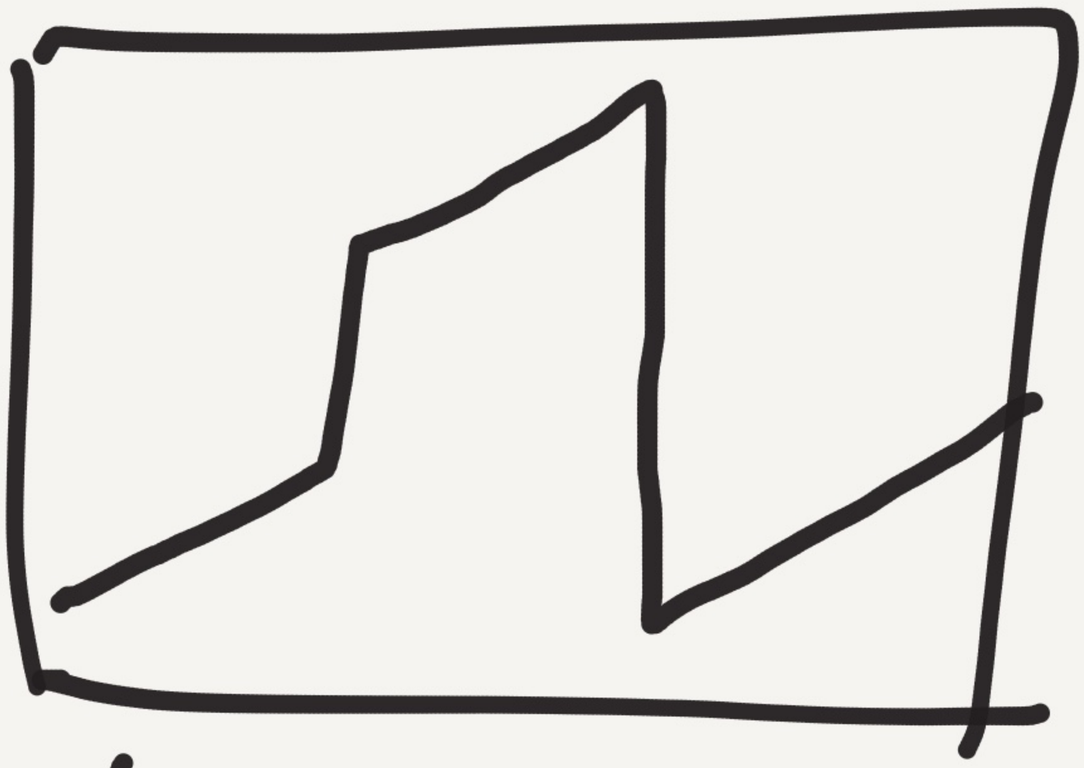
set

- add(6)
  - [read({1,2,3})
    cas({1,2,3}, {1,2,3,6})]
- read({1,2,3, ...})

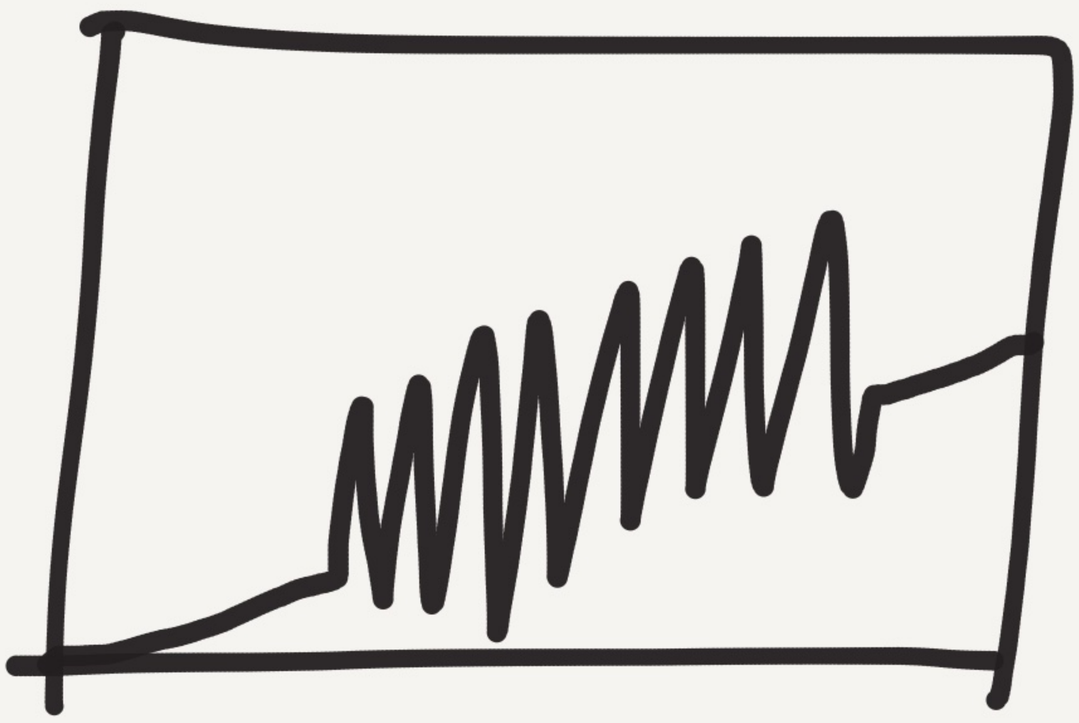
Merkleeyes Race



clock skew

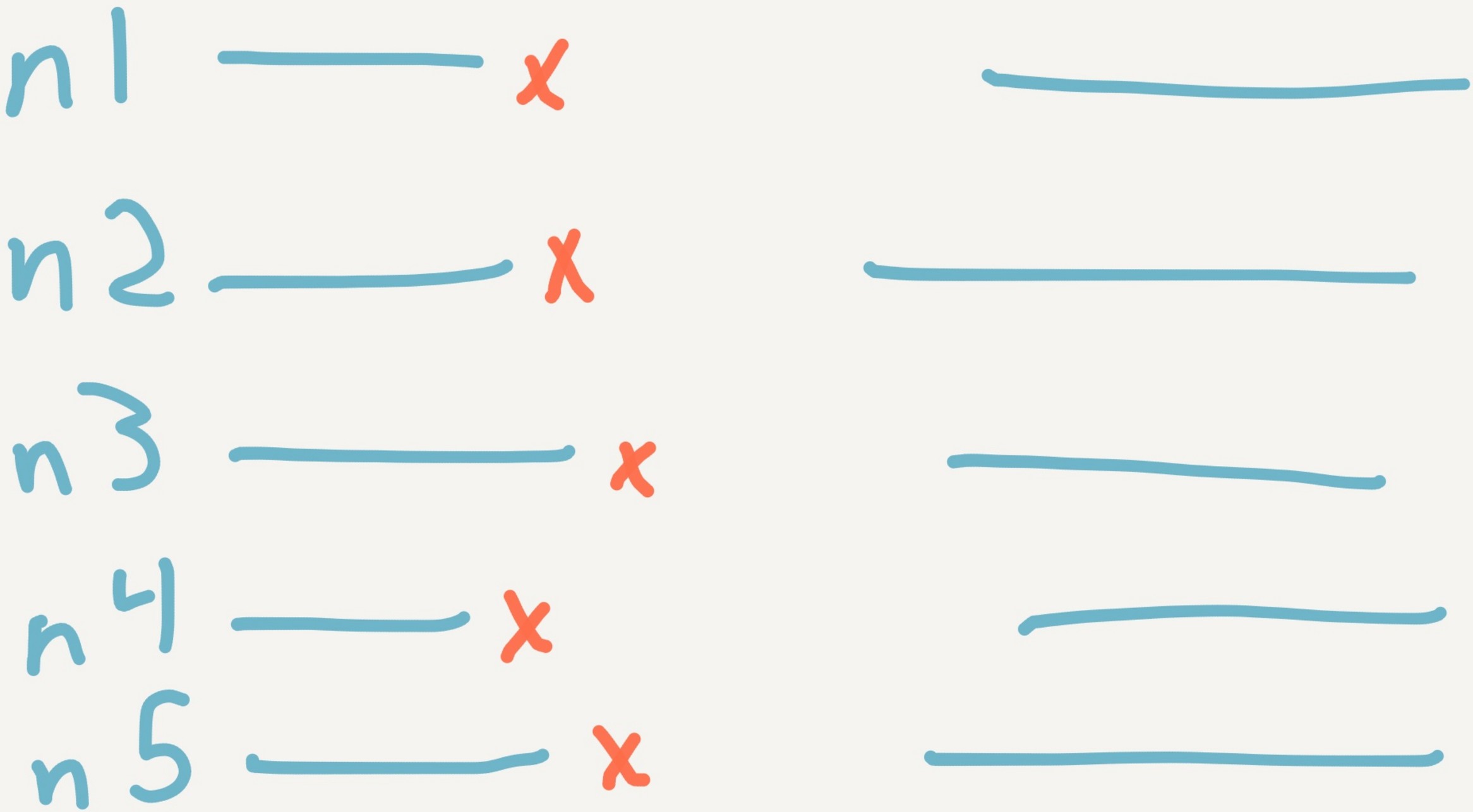


bump

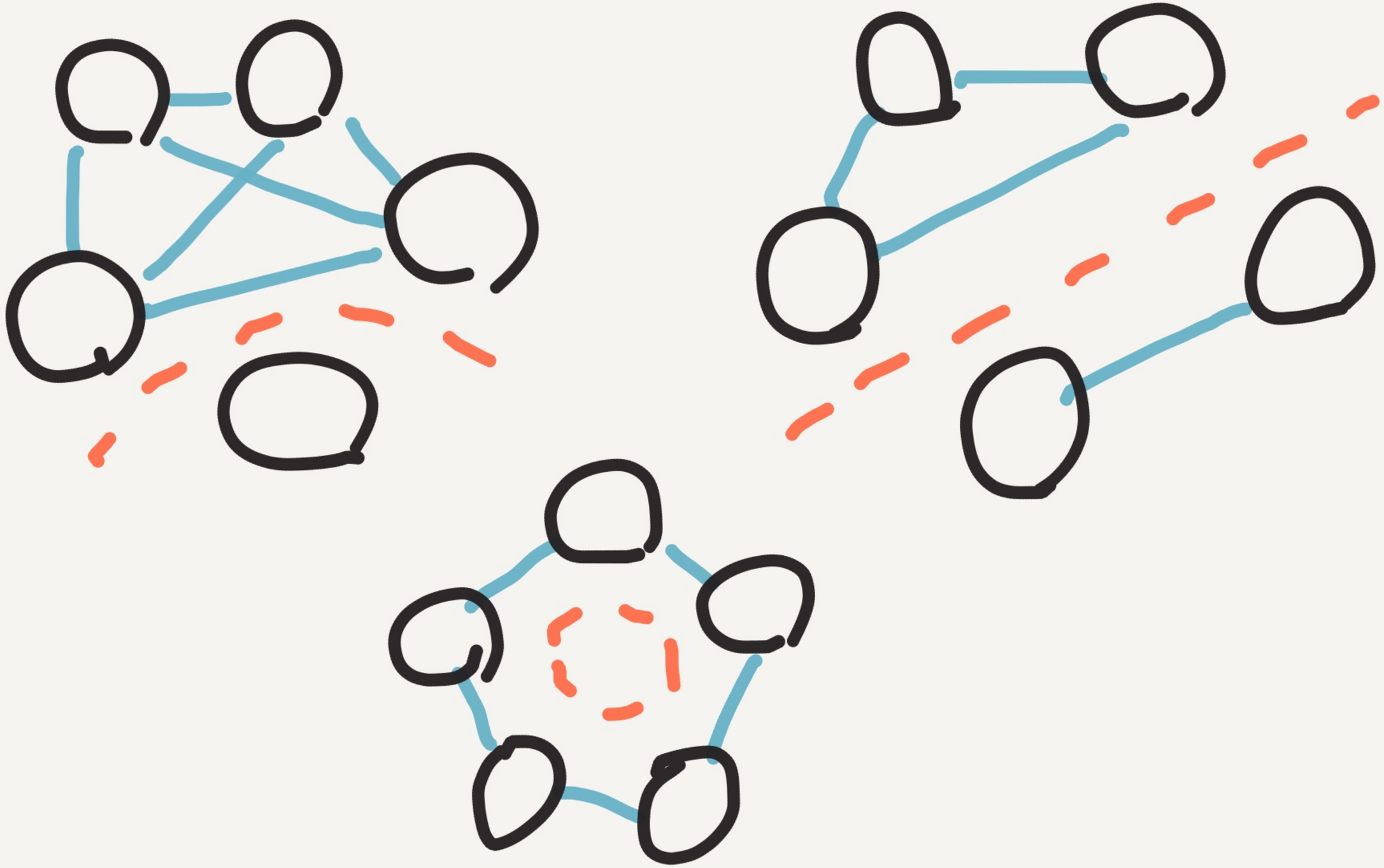


strobe

crashes



partitions



Duplicate Validators

$$7A\ B^{2}$$

$$7A^{1}\ C^{2}$$

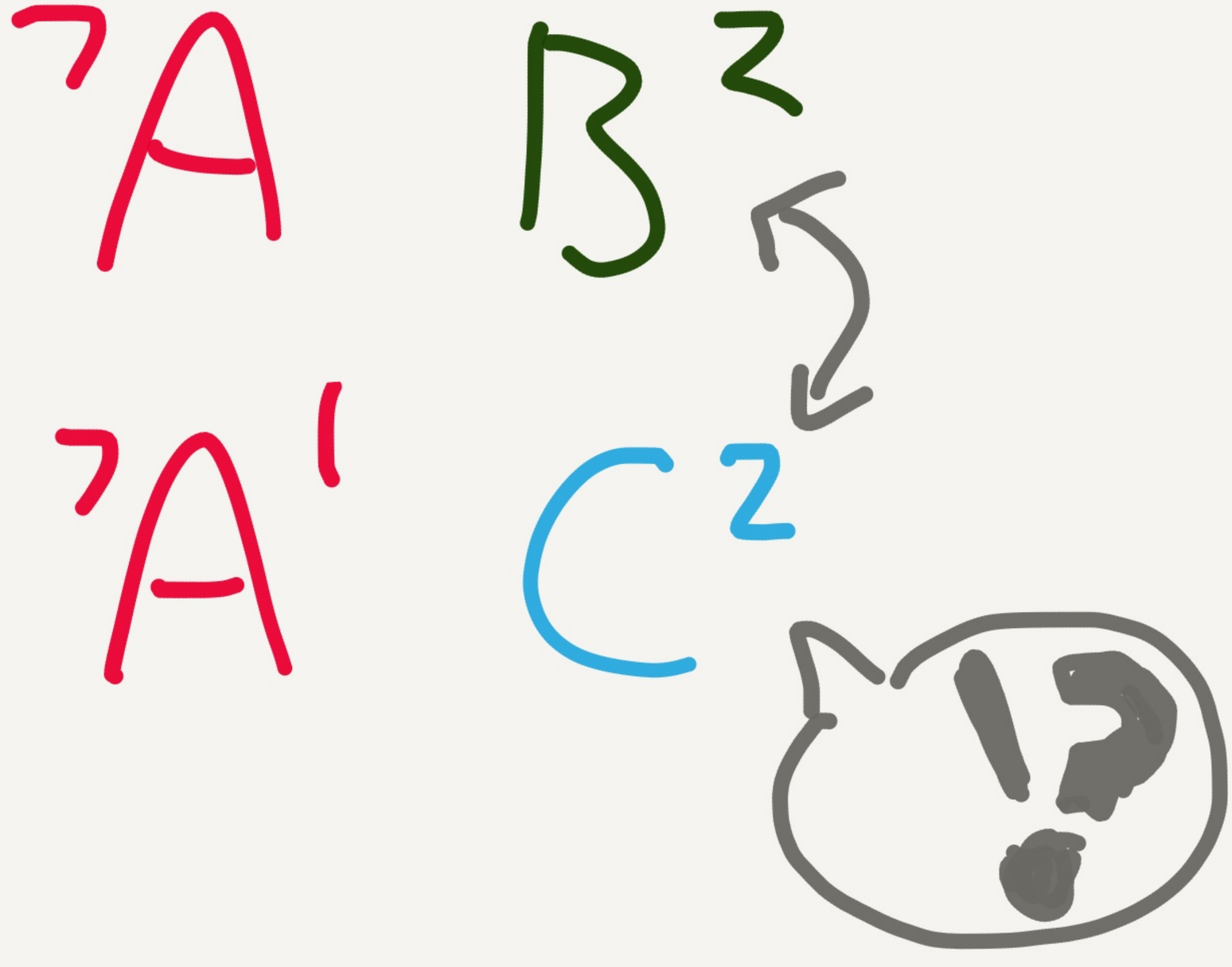
$$7 + 2 + 2 = 11$$

Duplicate Validators

$^{7}A\ B^{2}\qquad 9/11$

$^{7}A^{1}\ C^{2}\qquad 9/11$

Duplicate Validators



Lost Documents

iff

$\geq 1/3$

of votes

file truncation

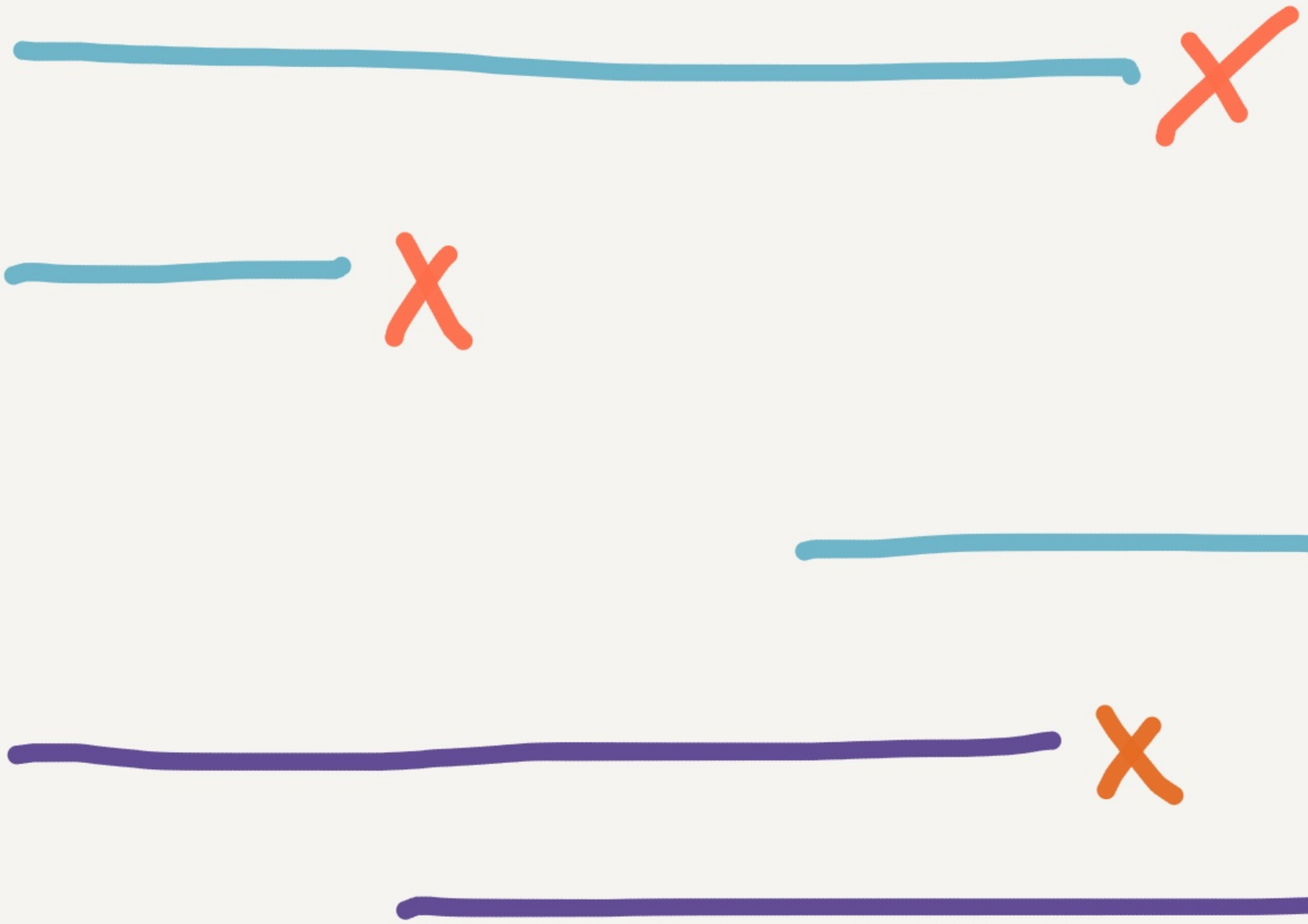
Merkleeyes crash!

(goleveldb bug)

Tendermint

Doesn't fsync to disk!

Dynamic Reconfig



Data corruption & crashes are to be expected — Tendermint is still in beta

Team working to fix these issnes

hazelcast

3.8.3

In-memory

Sets

Lists

IDs

Semaphores

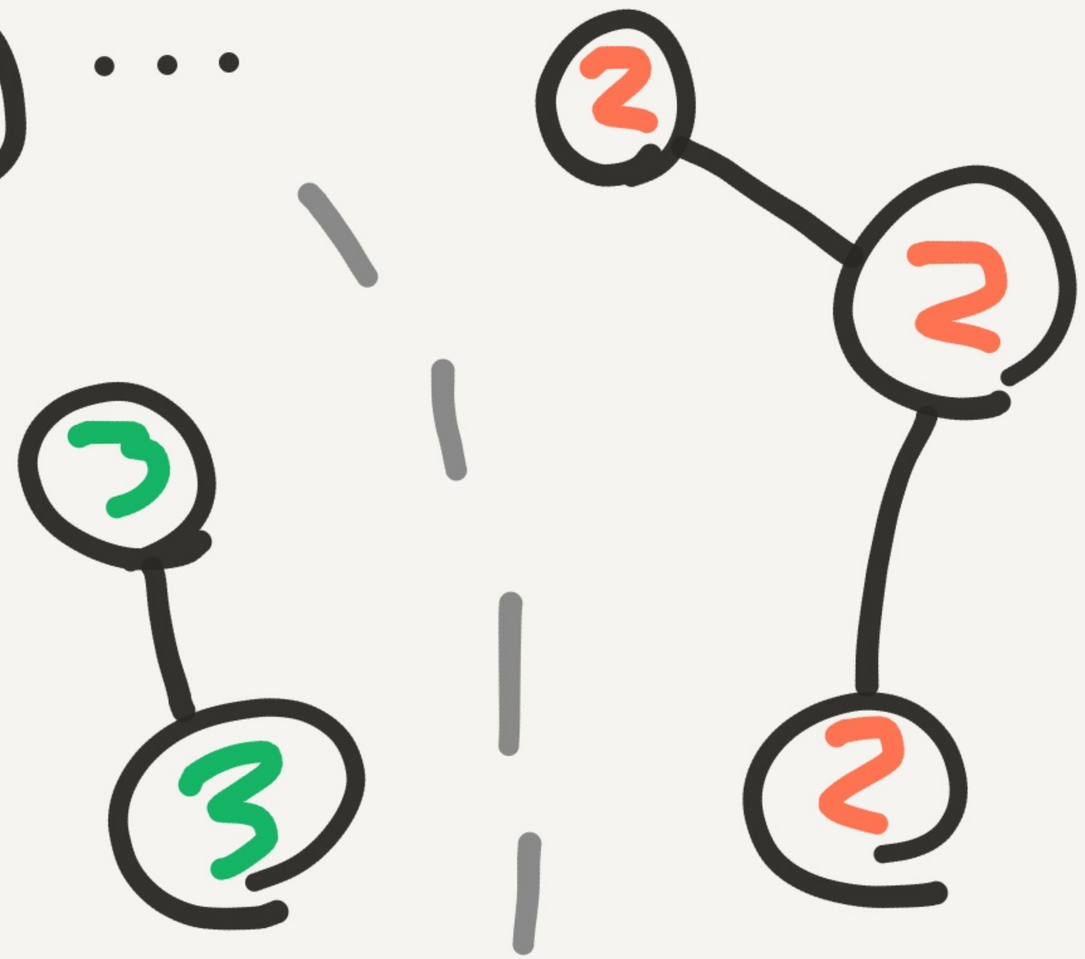
Queues

Maps

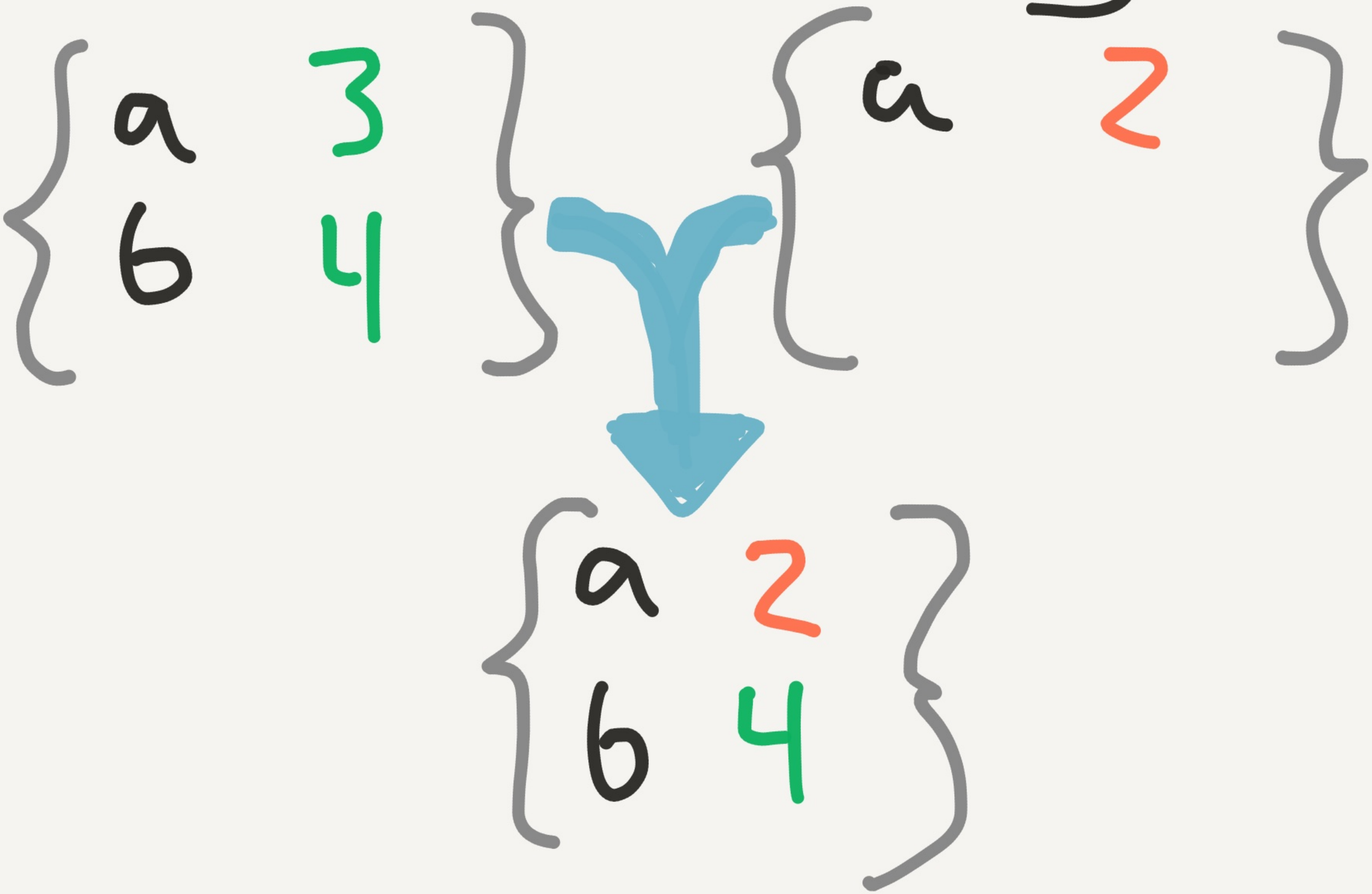
Locks

Atomic Longs

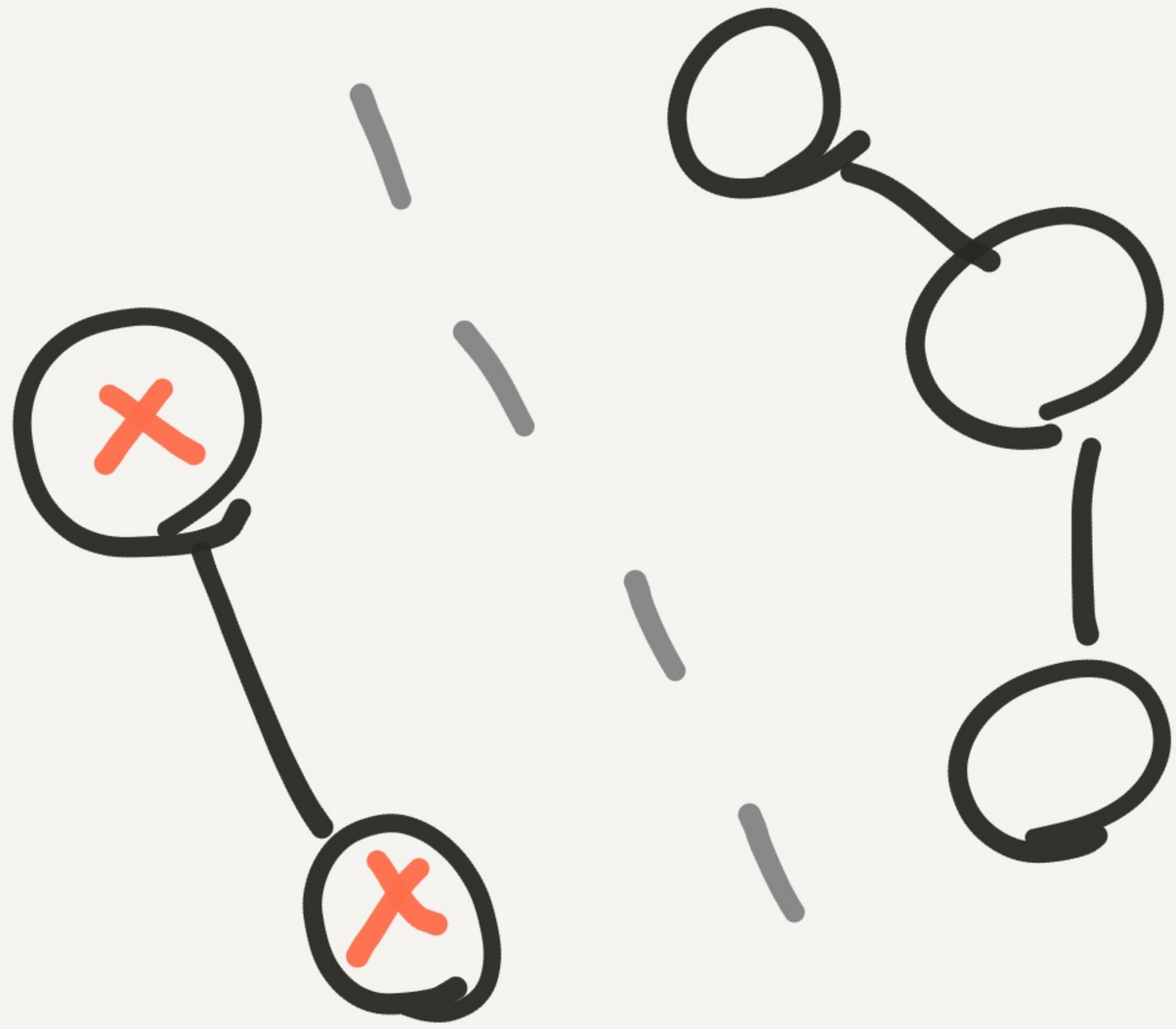
But in a partition, components run independently...



Maps have merge



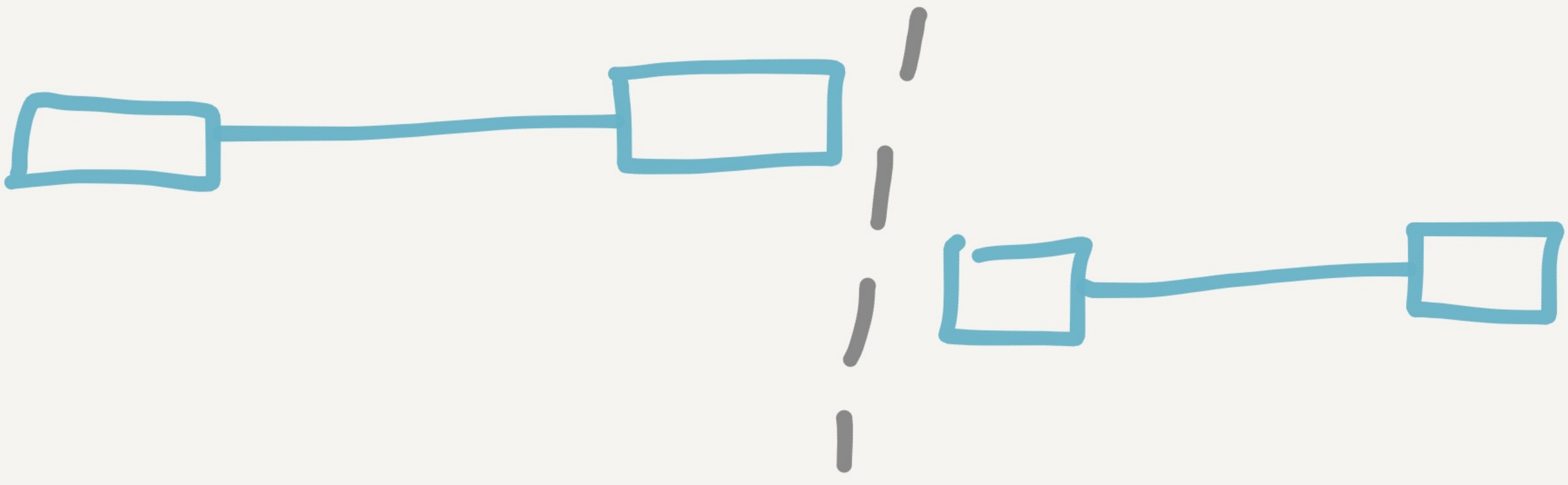
Split-Brain Protection

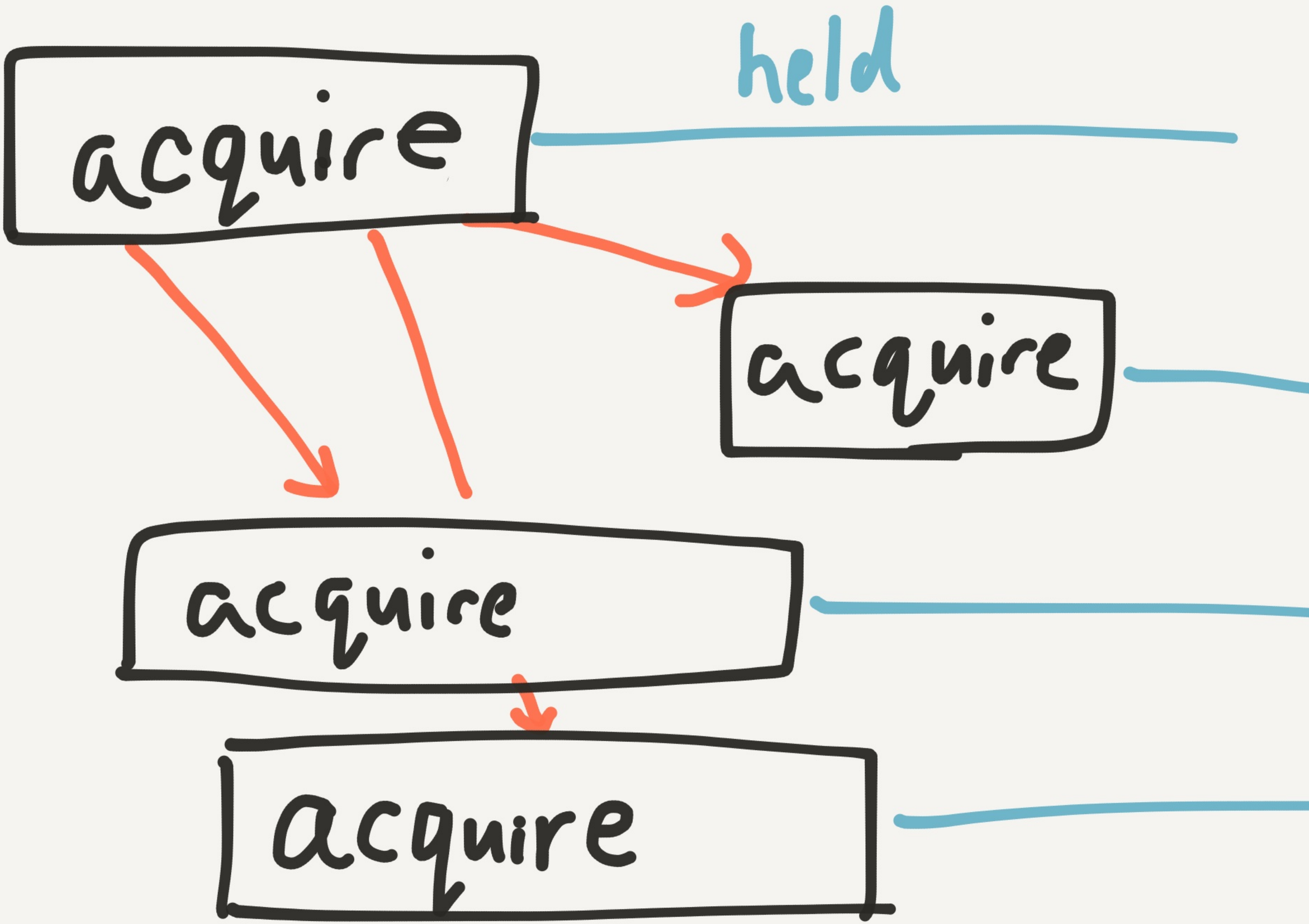


(10^{+} seconds)

Locks

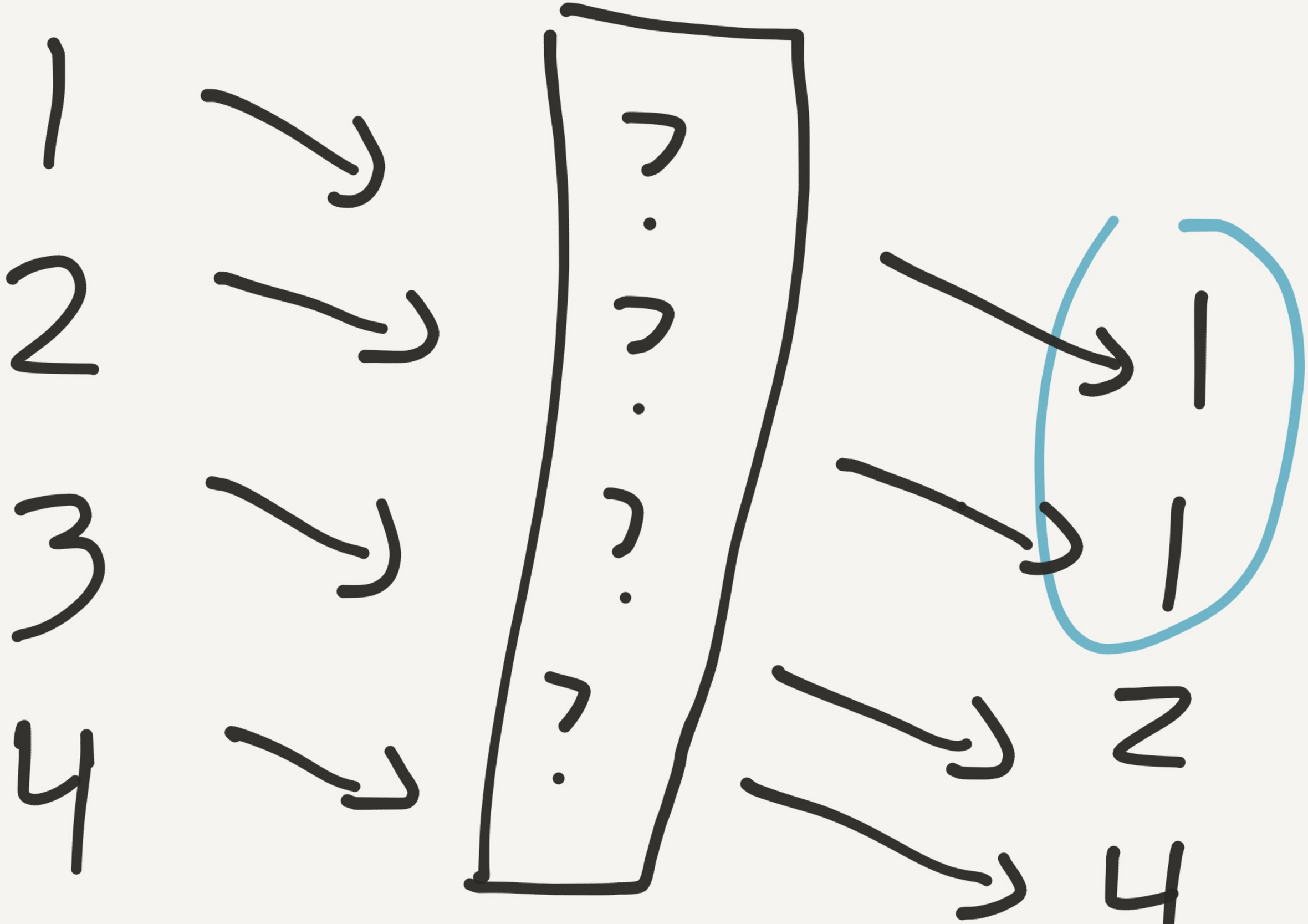
"Guaranteed to be exec-
-uted by only one thread
in the cluster"




held
acquire
acquire
acquire
acquire

Queues

"...you can add an item in one machine and remove it from another one"


1
2
3
4
?
?
?
?
?
1
1
2
4

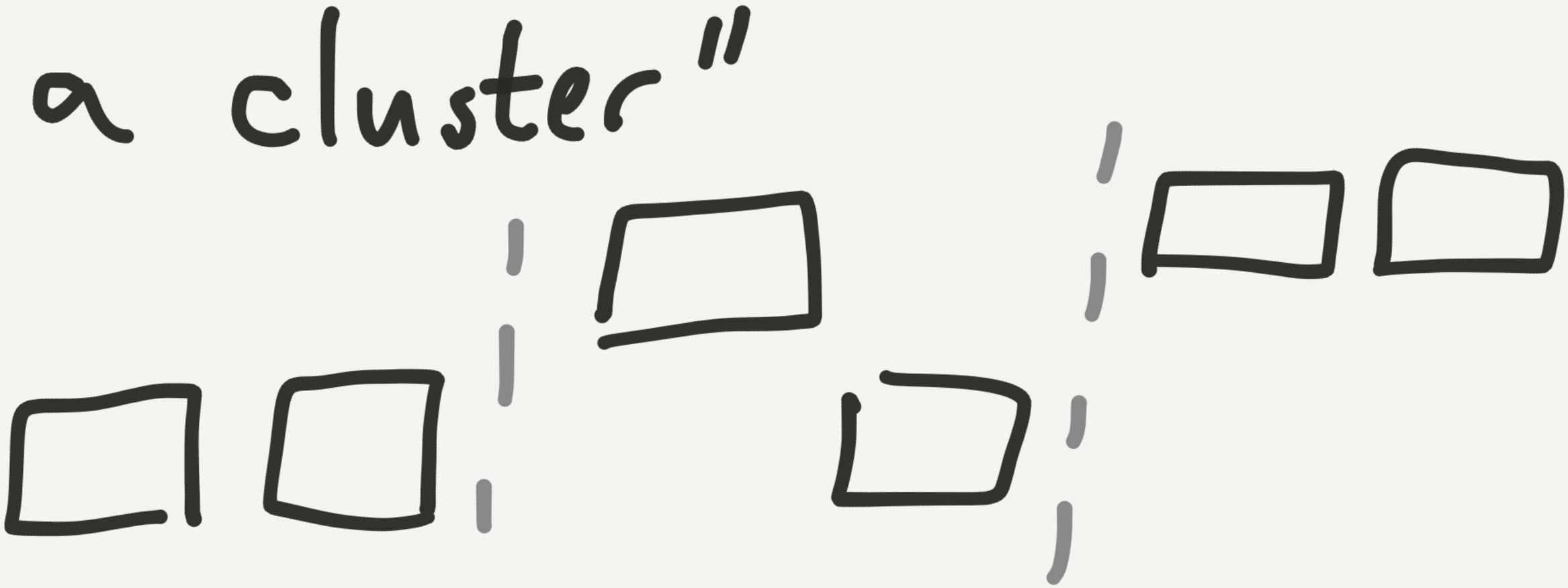
~2% duplicated

~2% lost

worse with default timeouts

Atomic Reference

"Guaranteed atomic compare and set across a cluster"



increment

2+1=3

"still 2?"
"yes!"

2

3

"still 2?"
"nope!"

increment

2+1=3

increment

2+1=3

"still 2?"
"yes!"

2

3

3

increment

2+1=3

increment

2+1=3

"still 2?"
"yes!"

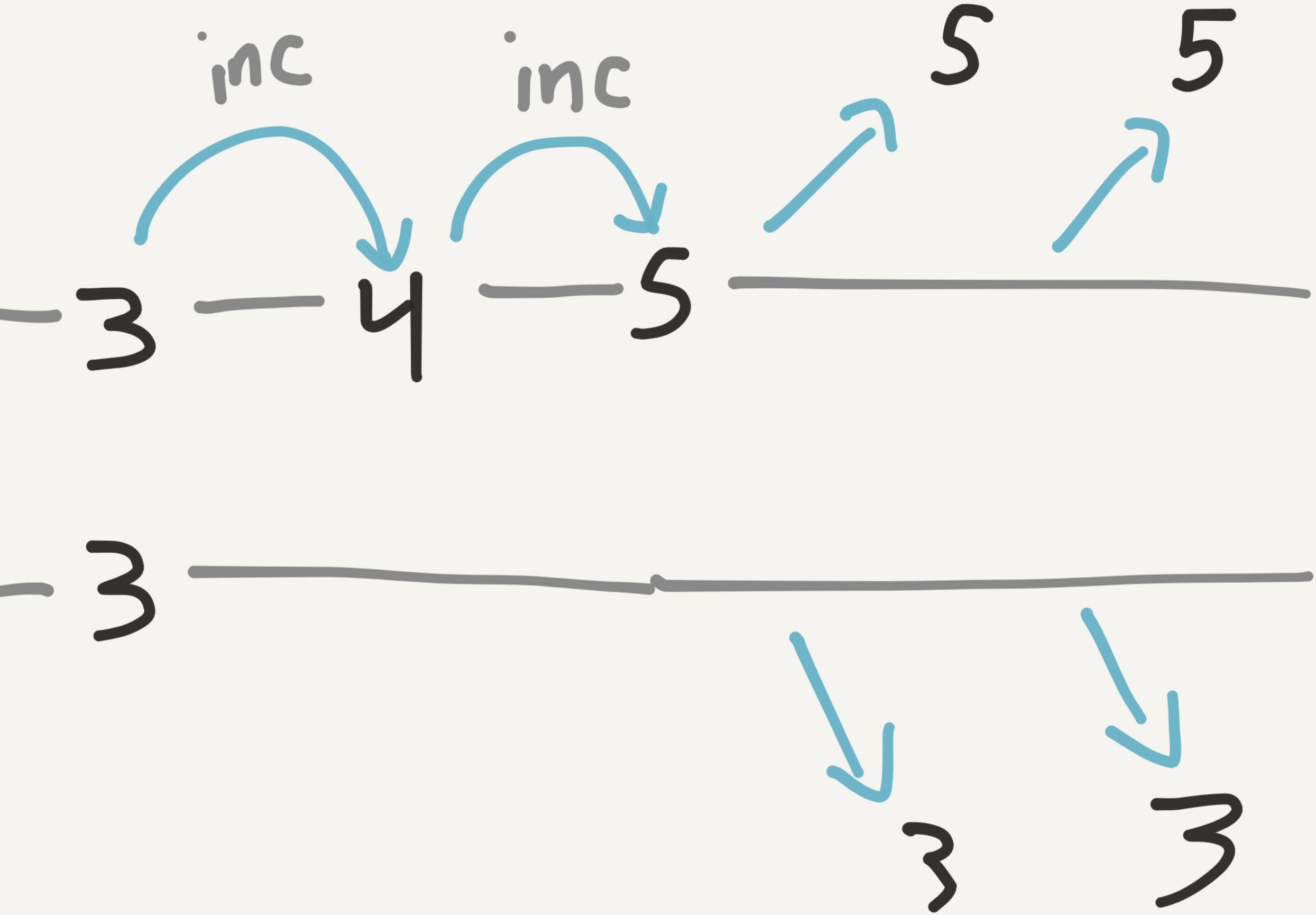
2 — 3

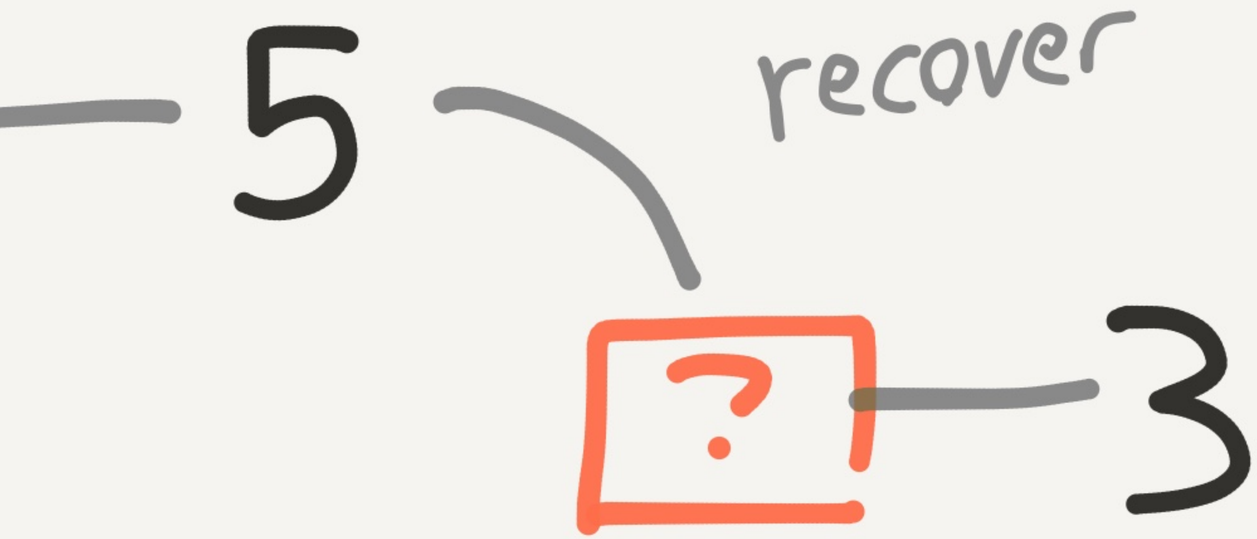
— 2

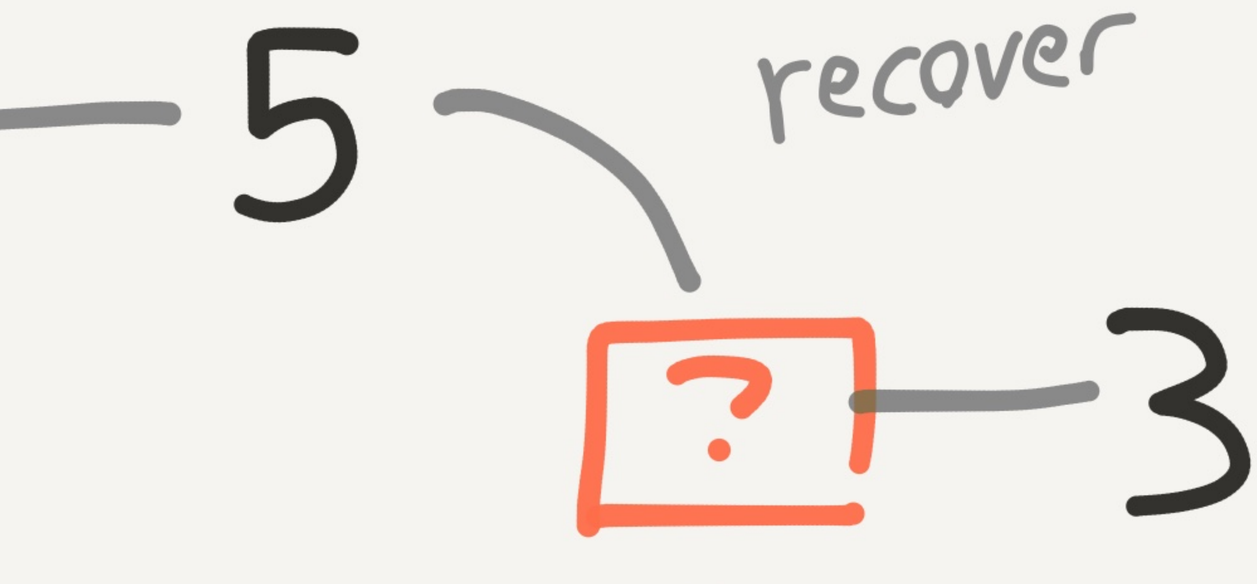
increment

2+1=3

2 — 3


inc
inc
5
5
3
4
5
3
3


5
recover
?
3
3


5
recover
?
3
3

5

recover

?

3

3

LOOK AT
your
LIFE

5

recover

?

3

3

LOOK at your CHOICES

LOOK AT
your
LIFE

ID Generator

- Same deal

~91,000 dup ids in 30s

834,000 generated

Maps

- putIfAbsent(k, v)
- replace(k, v, v')

Lost updates!

Even with quorum protection

However...

w/ custom merge function, can build safe CRDTs

Recommendations

- Choose replication algo appropriate for datatype
  - Flake IDs, PN-Counters, Paxos

- Don't rely on Hz for safety!
  - BagriDB txn IDs
  - OrientDB cluster state (used to be txns!)

DISTRIBUTED

LOCKS

aren't

REAL

AEROSPIKE

3.99.0.3

Distributed, sharded

{ Document }

store

3.5.4:

- Trivial split-brain
- Lost updates
- Stale & Dirty reads

2 Years of fundamental redesign work...

4.0: Linearizable Mode

- Custom Consensus protocol

- Reduced loss in AP mode

(start of test)

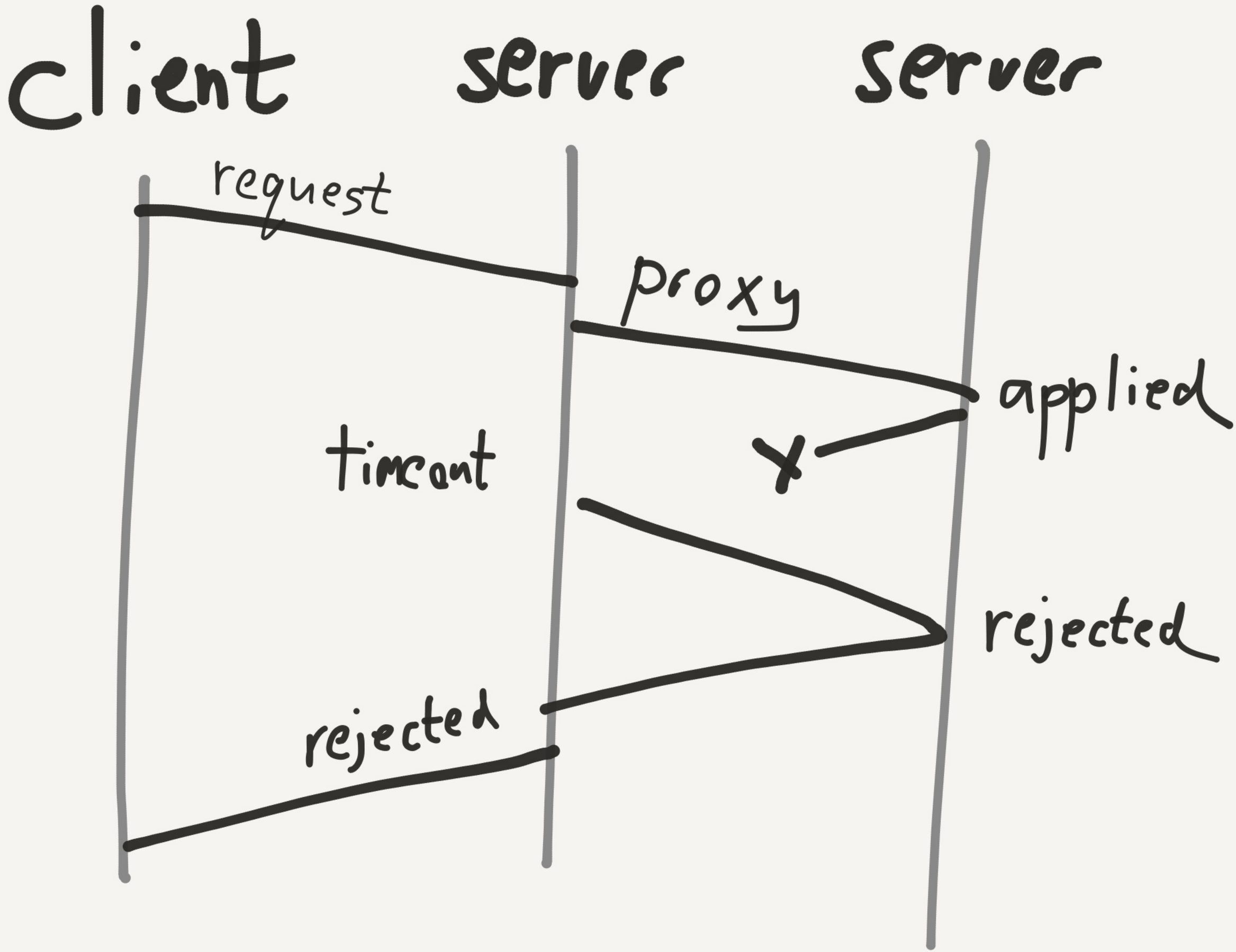
read 2

fail

write 2

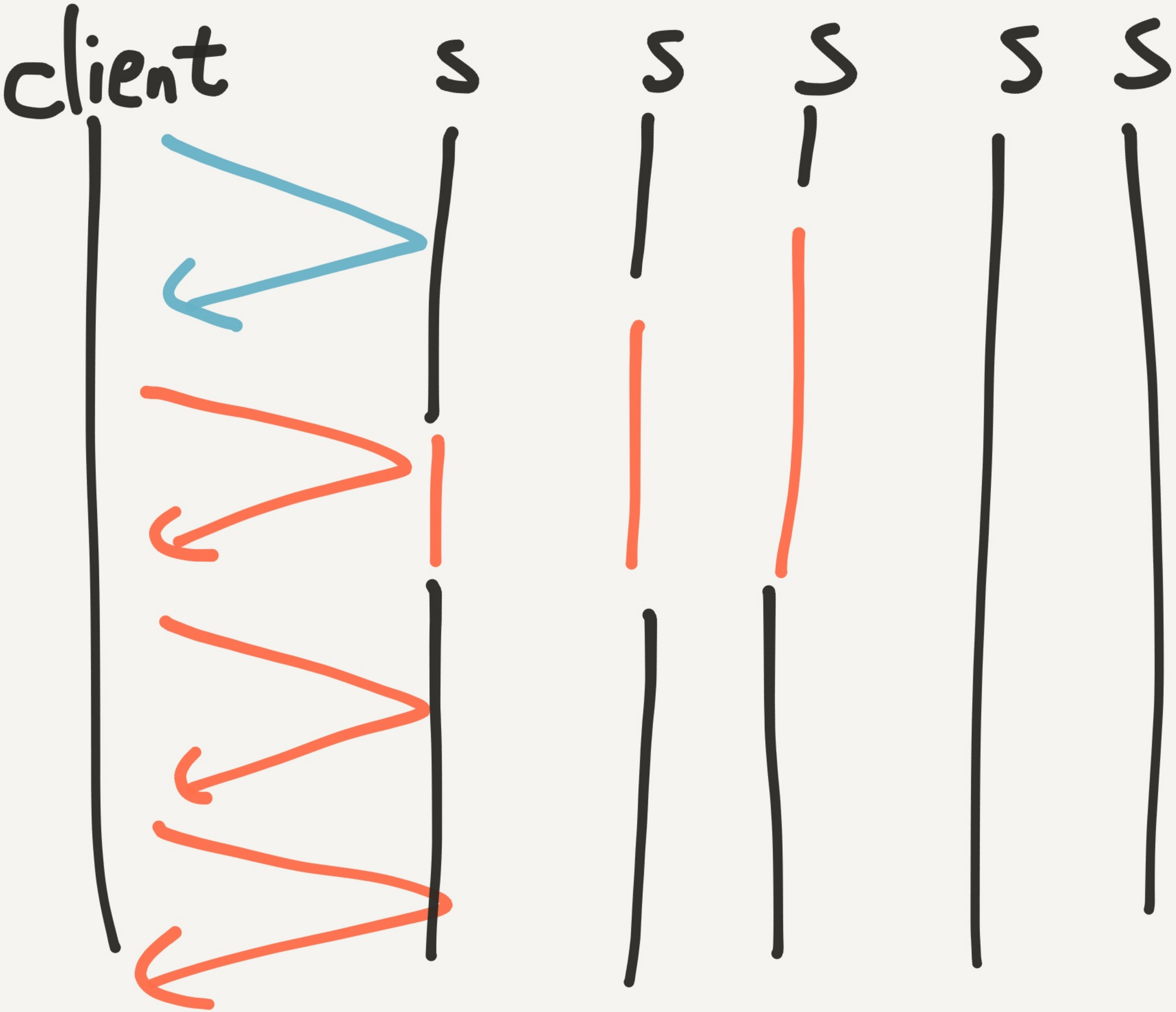
ok

read 2


client
server
server
request
proxy
applied
timeout
rejected
rejected

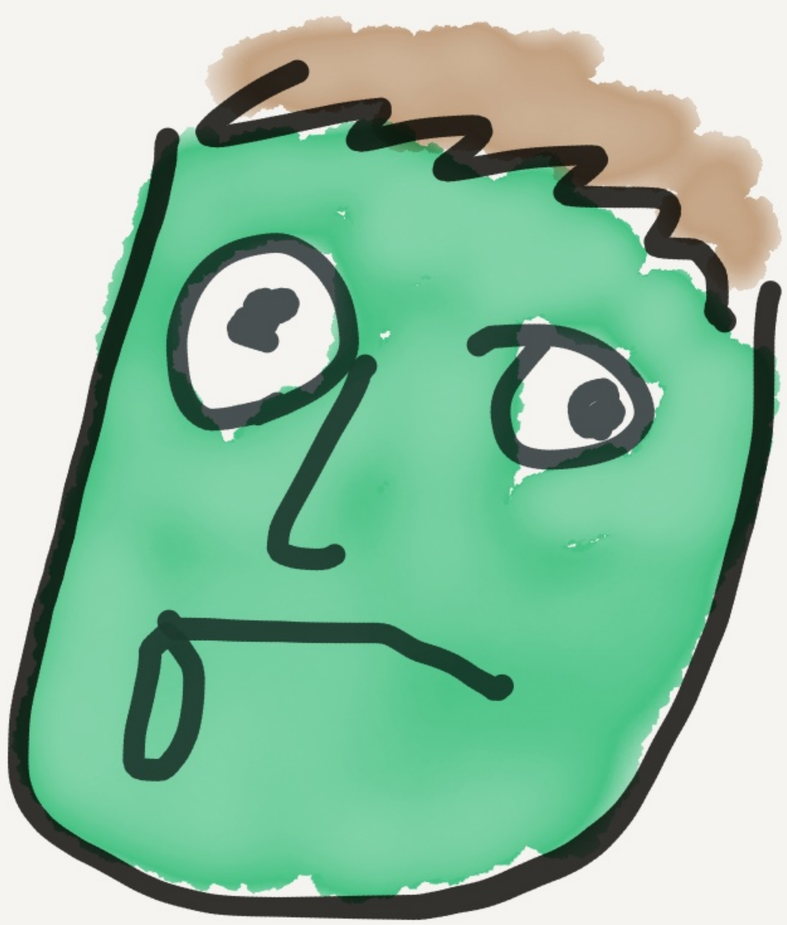
fixed in 3.99.1.5

Node crashes


client
S
S
S
S
S

[press E to revive]

Dead partitions may not be all there...



AS didn't flush writes to disk before acknowledging

(fixed in 3.99.2.1)

Clock Skew can lead to Lost updates

clock

regime	wall-clock	counter

increment on election

node local

for mult. updates in same clock

fits in cache line

regime	wall-clock	counter

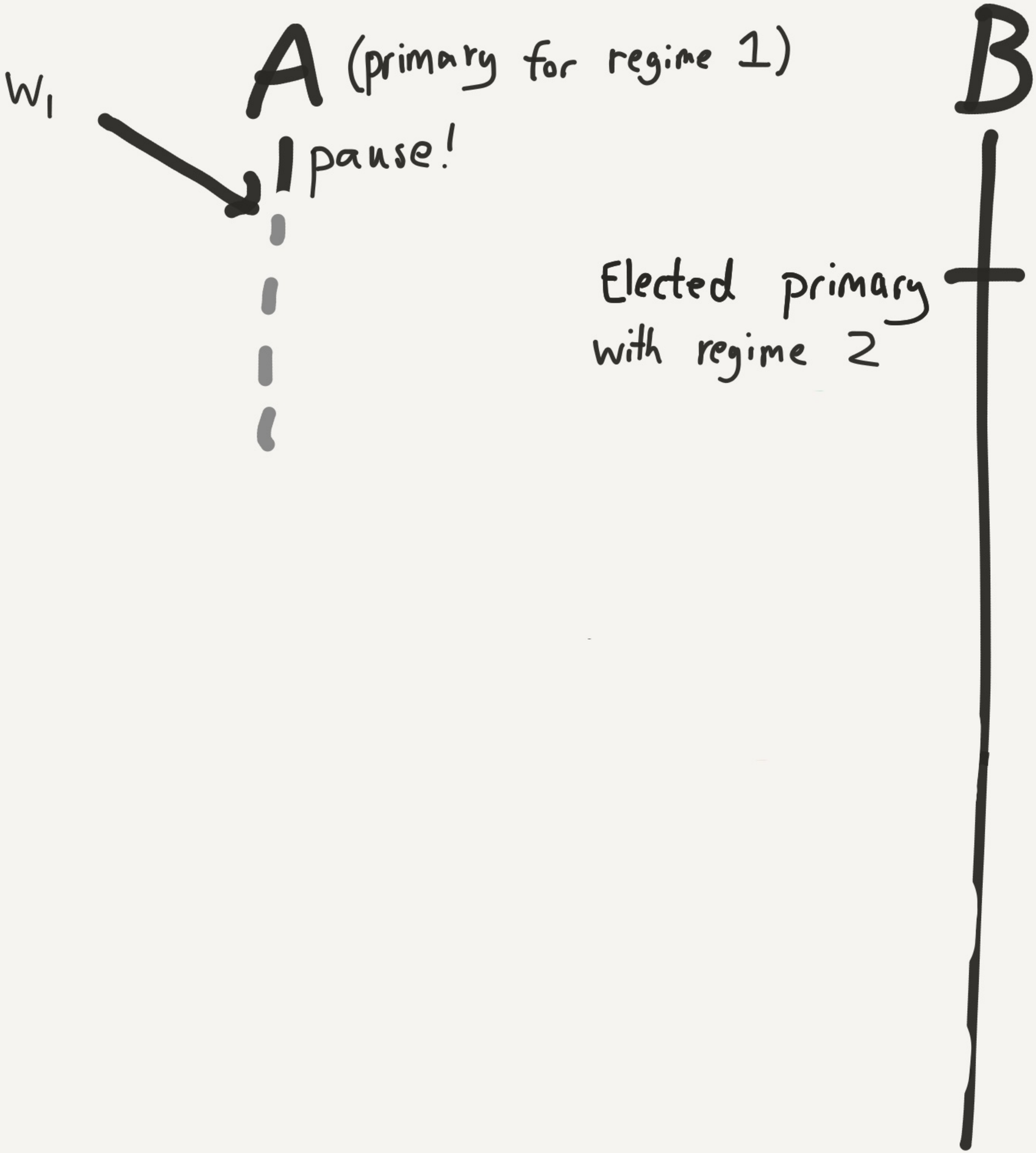
6 bits

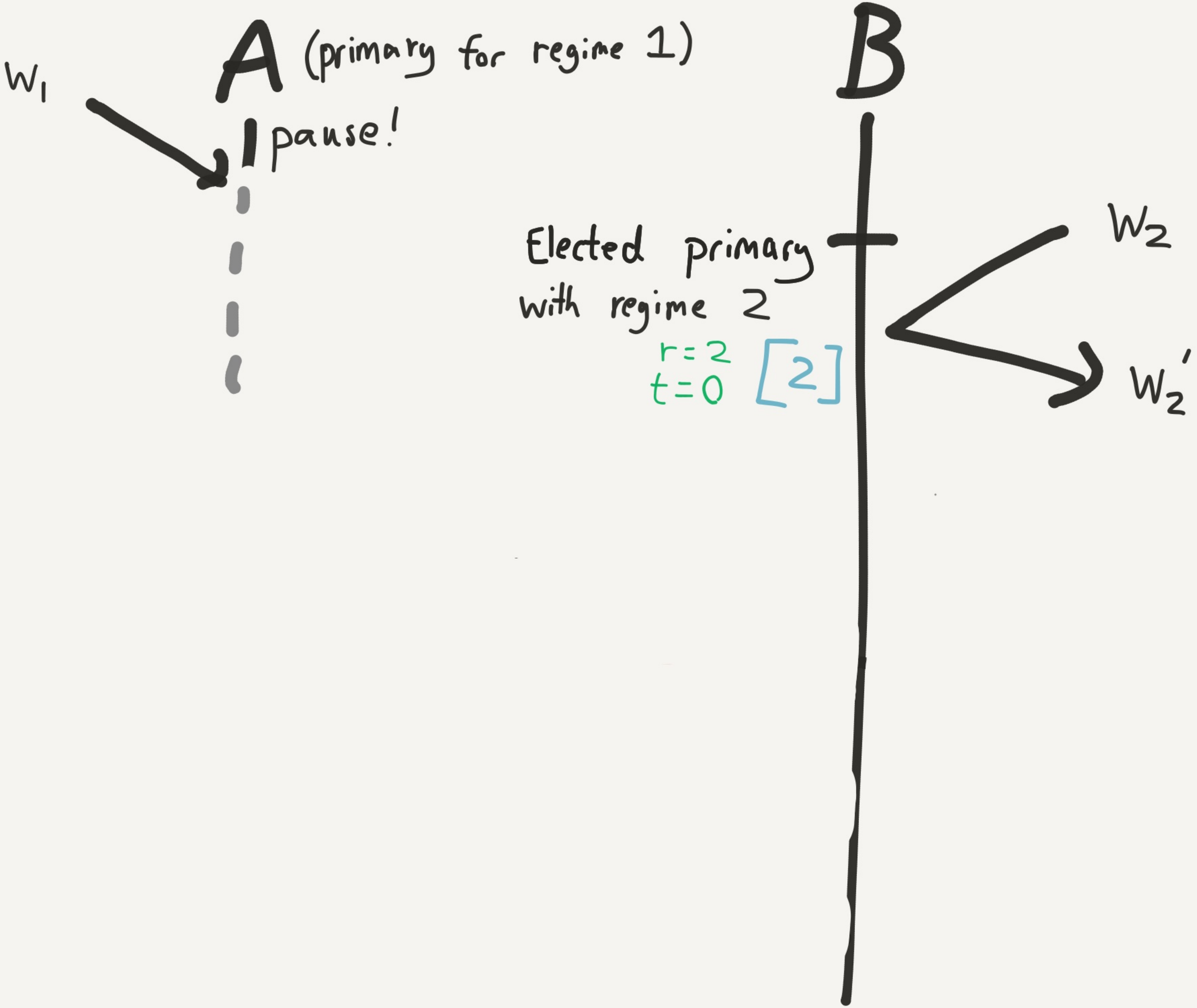
W_1

A (primary for regime 1)

pause!

B


A (primary for regime 1)
B
W_1
pause!
Elected primary
with regime 2


A (primary for regime 1)
B
W_1
pause!
Elected primary with regime 2
r=2
t=0
[2]
W_2
W_2'

A (primary for regime 1)

B

W_1

pause!

Elected primary with regime 2

r=2
t=0 [2]

W_2

W_2'

Wake up process W_1

[1] r=1
t=27

replicate W_1

[2] r=2
t=0

A (primary for regime 1)

B

W_1

pause!

Elected primary with regime 2

r=2
t=0 [2]

W_2

W_2'

Wake up process W_1

[1] r=1
t=27

replicate W_1

[2] r=2
t=0

regime 1 < 2

W_1'

A (primary for regime 1)

B

W_1

pause!

Elected primary with regime 2

r=2
t=0 [2]

W_2

W_2'

Wake up process W_1

[1] r=1
t=27

replicate W_1

[2] r=2
t=0

regime 1 < 2

reconcile

W_1'

r=1
t=27 [1]

[1] r=1
t=27

Both clock skew &
not using fsync were
known failure modes

Stay in clock-skew bounds & use durable flag...

looks linearizable!

But XDR scans, UDFs don't go through the SC mechanism yet

future work!

Recap

Read the docs!

CAREFULLY

Then test it

— for —

YOURSELF

"Strict"

"ACID"

"Strong"

Be Formal

&

Be Specific!

Figure out the Invariants your system needs

Consider your failure modes

Process Crash

#kill -9 1234

Node failure

- AWS terminate
- Physical power switch

Clock Skew

```
# date 10280000
# faketime ...
```

SC/IO Pause

```
# killall -s STOP foo
# killall -s CONT foo
```

Network Partition

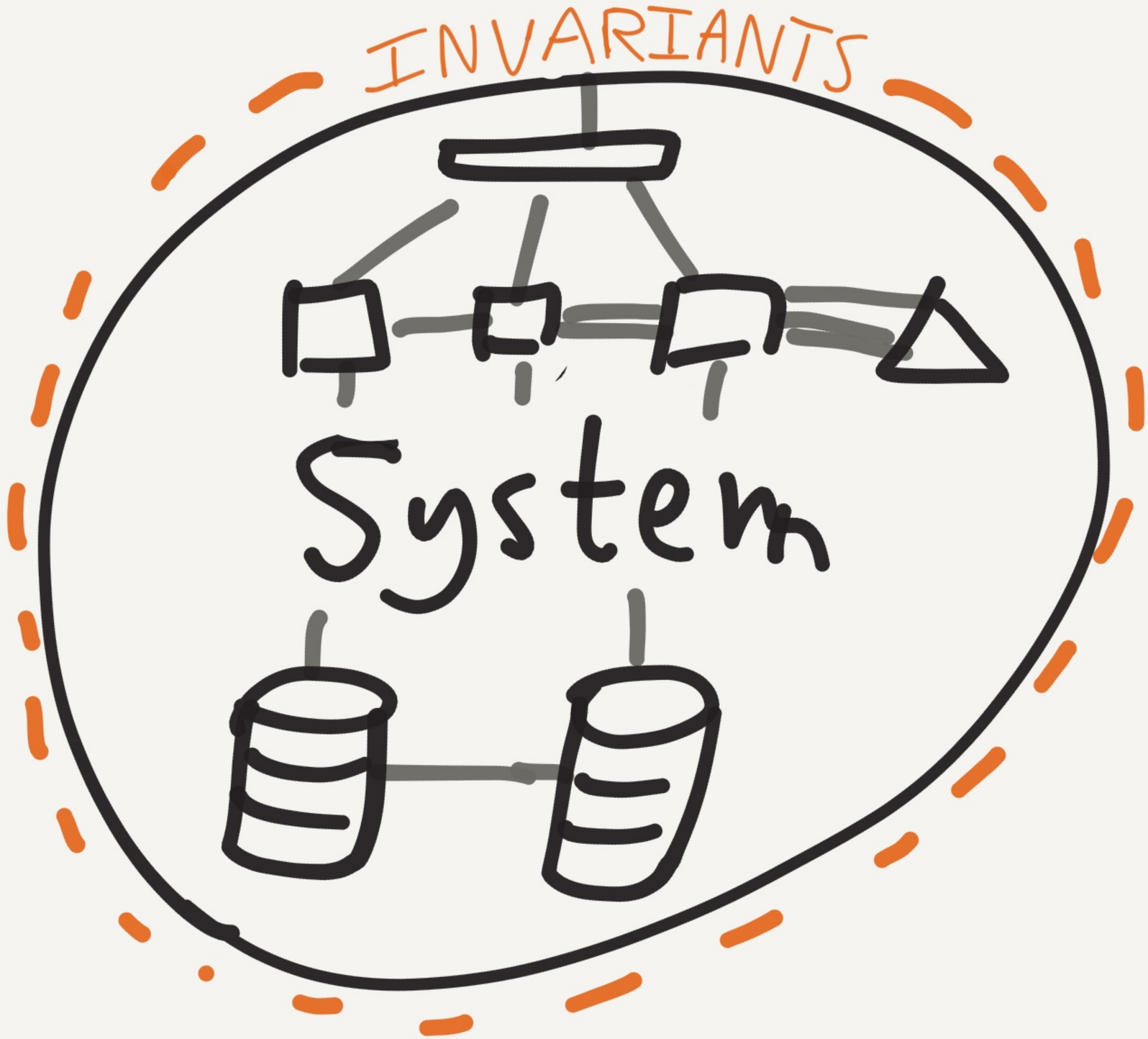
```
# iptables -j DROP
```

```
# tc qdisc ... delay ...
                drop ...
```

Test your Systems

end ——— to ——→ end

vs.



If you look for

perfection

you will never be

content

- Property testing
- High-level invariants
- With distsys failure modes

Thanks

Tendermint

Hazelcast

Jordan Halterman

Luigi Dell'Aquila

Luca Gamlli

Thanks

Denis Sukhoreslov

Aerospike

Peter Alvaro

http://jepsen.io