CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2015

goto; conference

JEPSEN IV

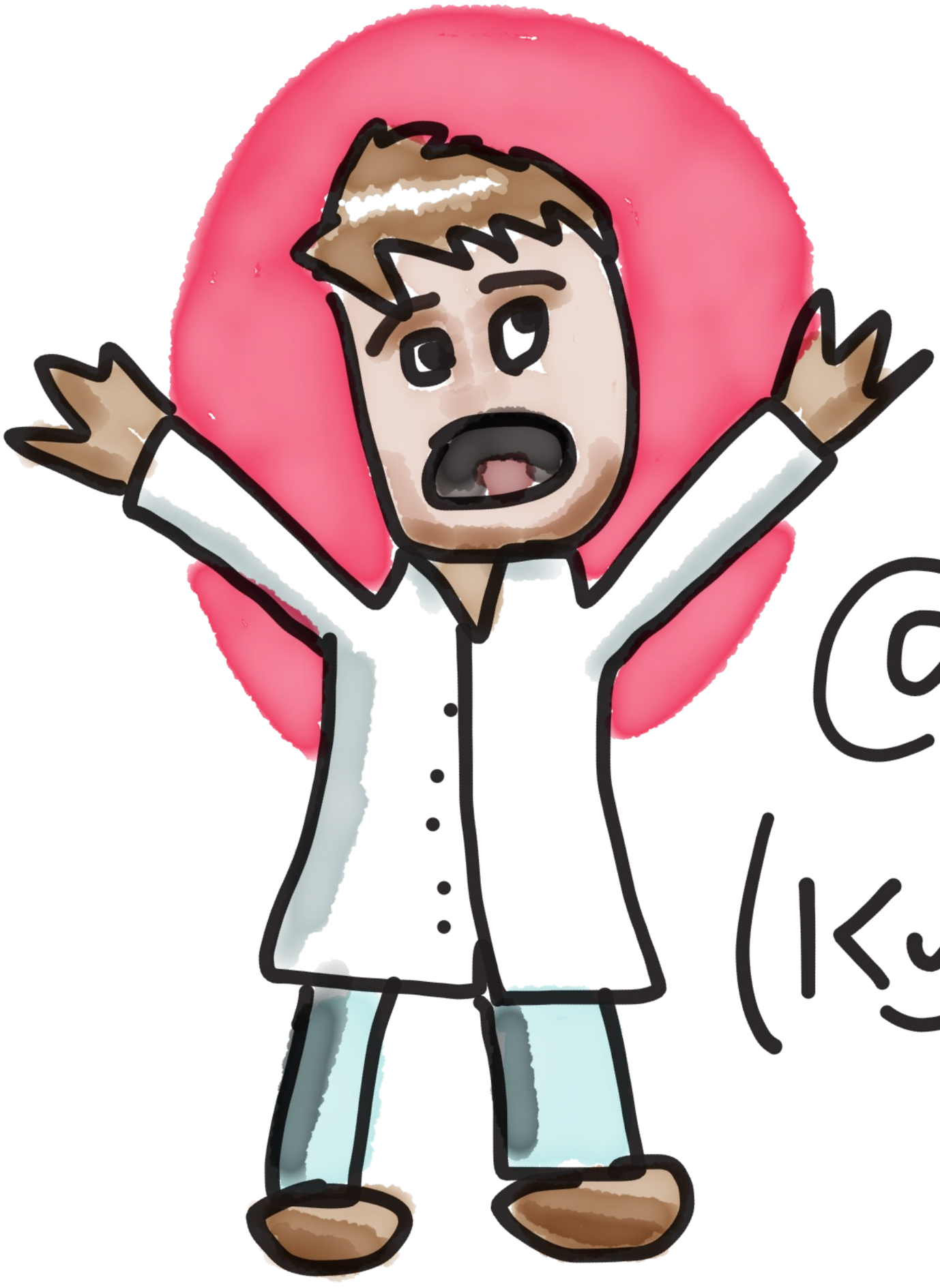
KYLE KINGSBURY

follow us @gotochgo Conference: May 11-12 / Workshops: 13-14

JEPSEN

IV

Hope Springs Eternal



@aphyr

(Kyle Kingsbury)

str/pe

Public API {

} API code

Ruby {

DBs

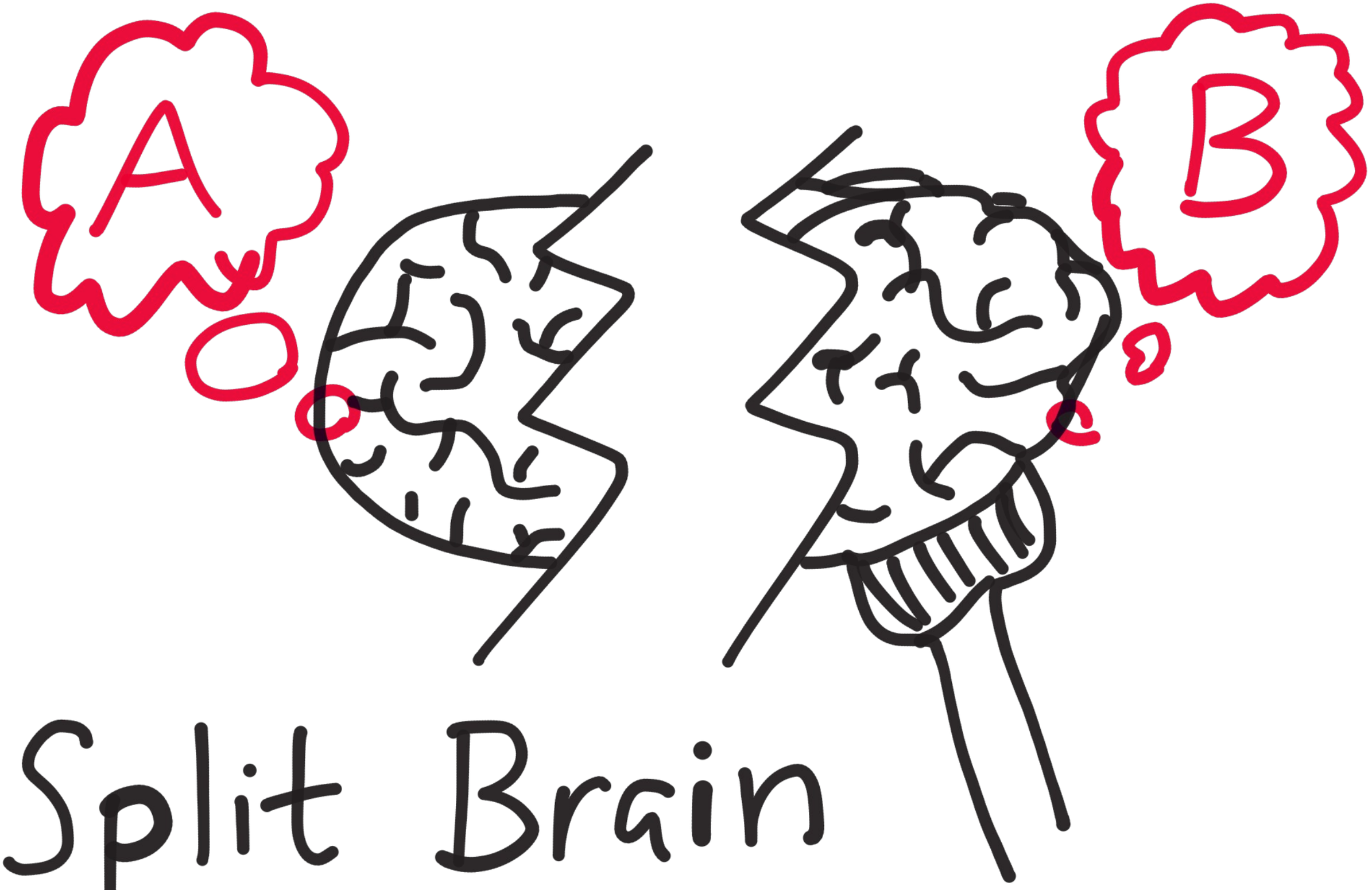
Every thing is fine!

Databases!

Queues!

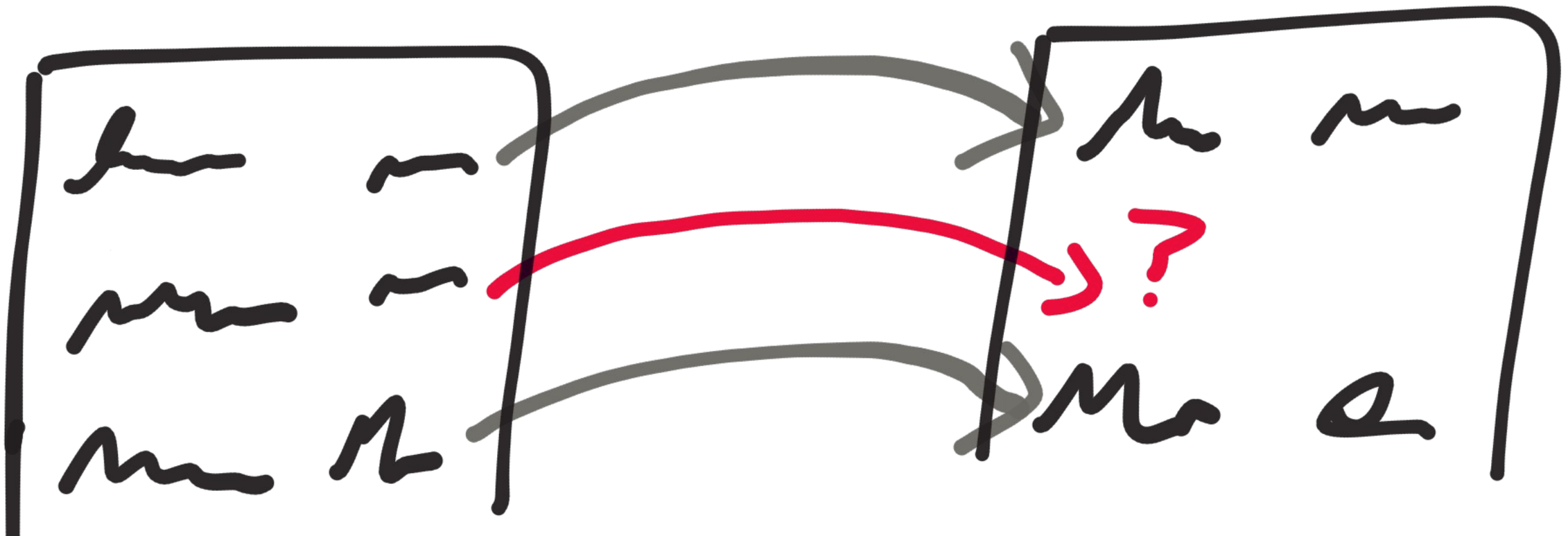
Discovery!

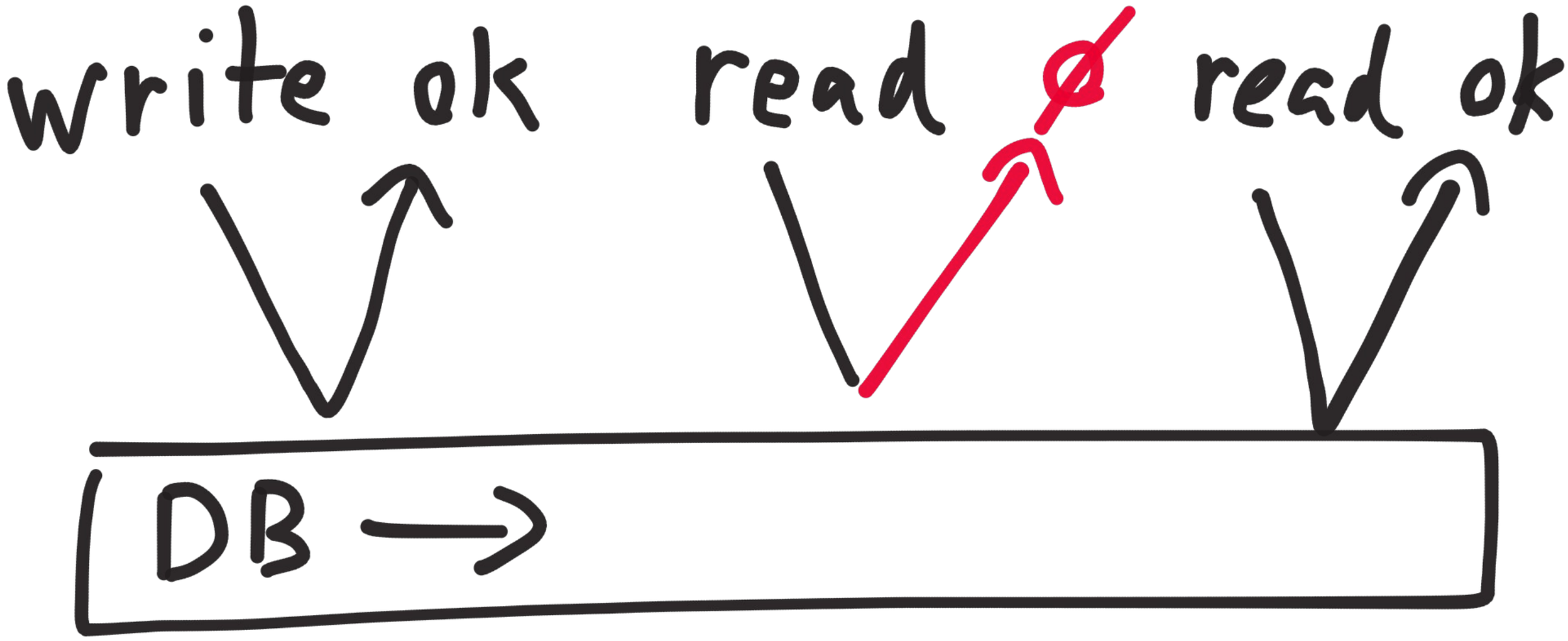
THE HORROR



Split Brain

Broken Foreign Keys

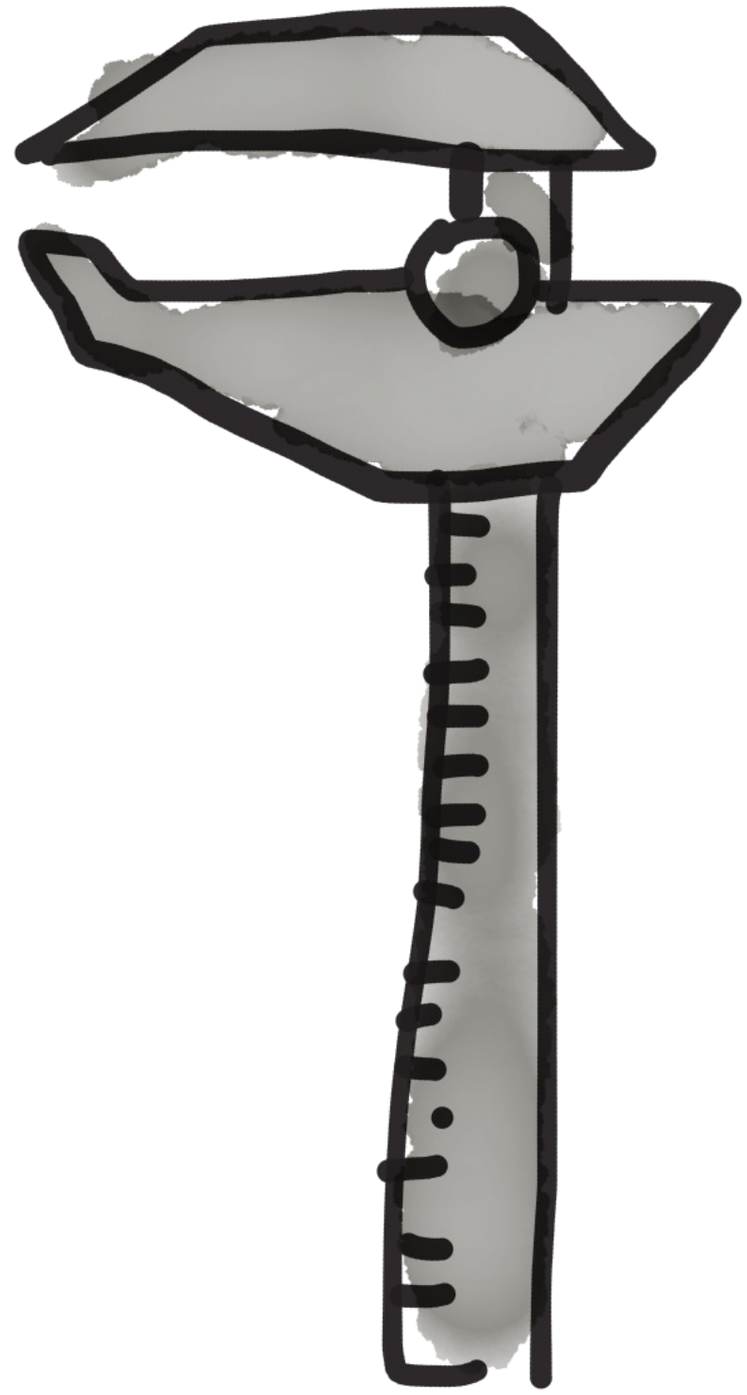




Anomalies

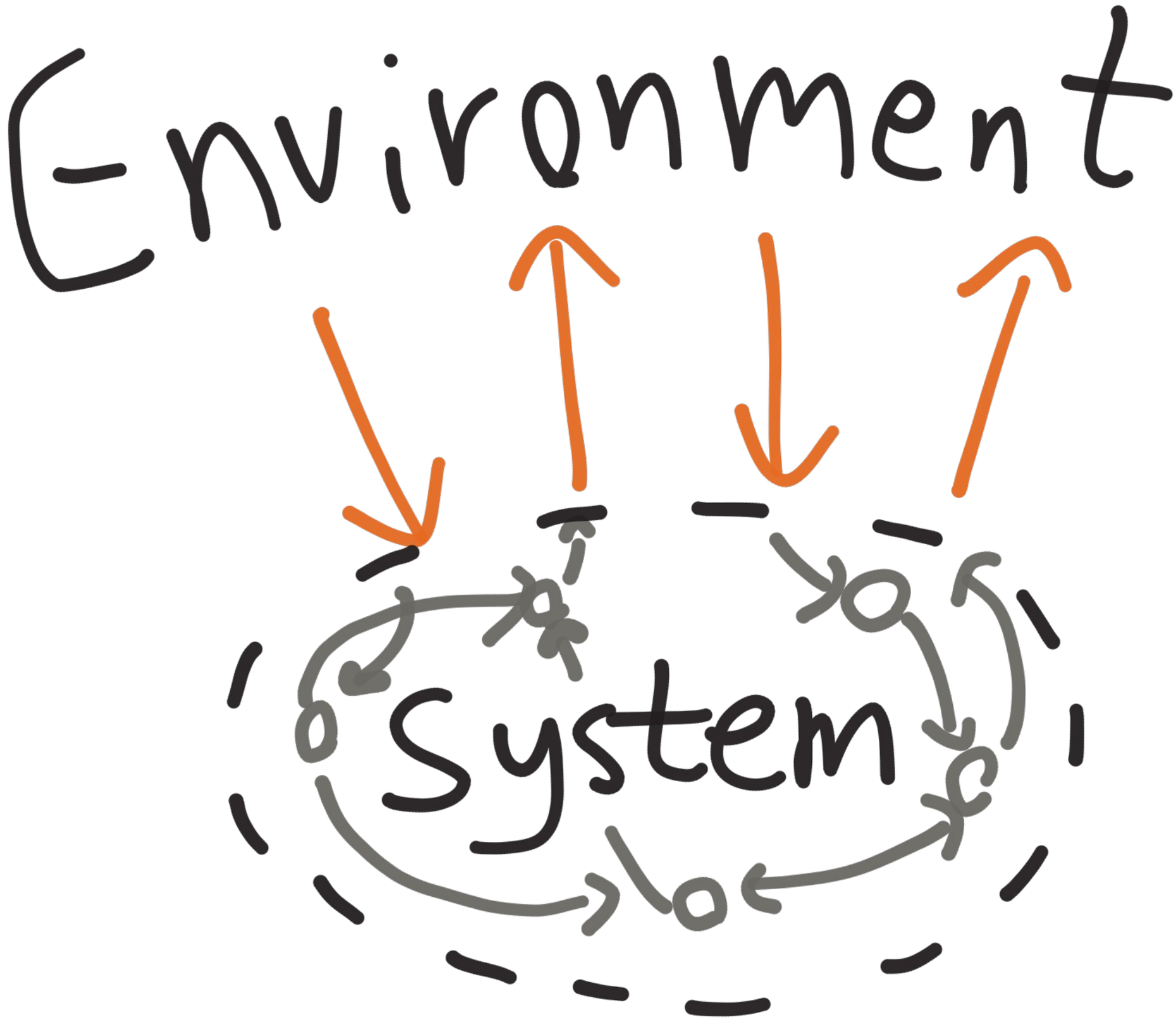
How do you
know if a
system is safe?

Measure your Systems



Jepsen

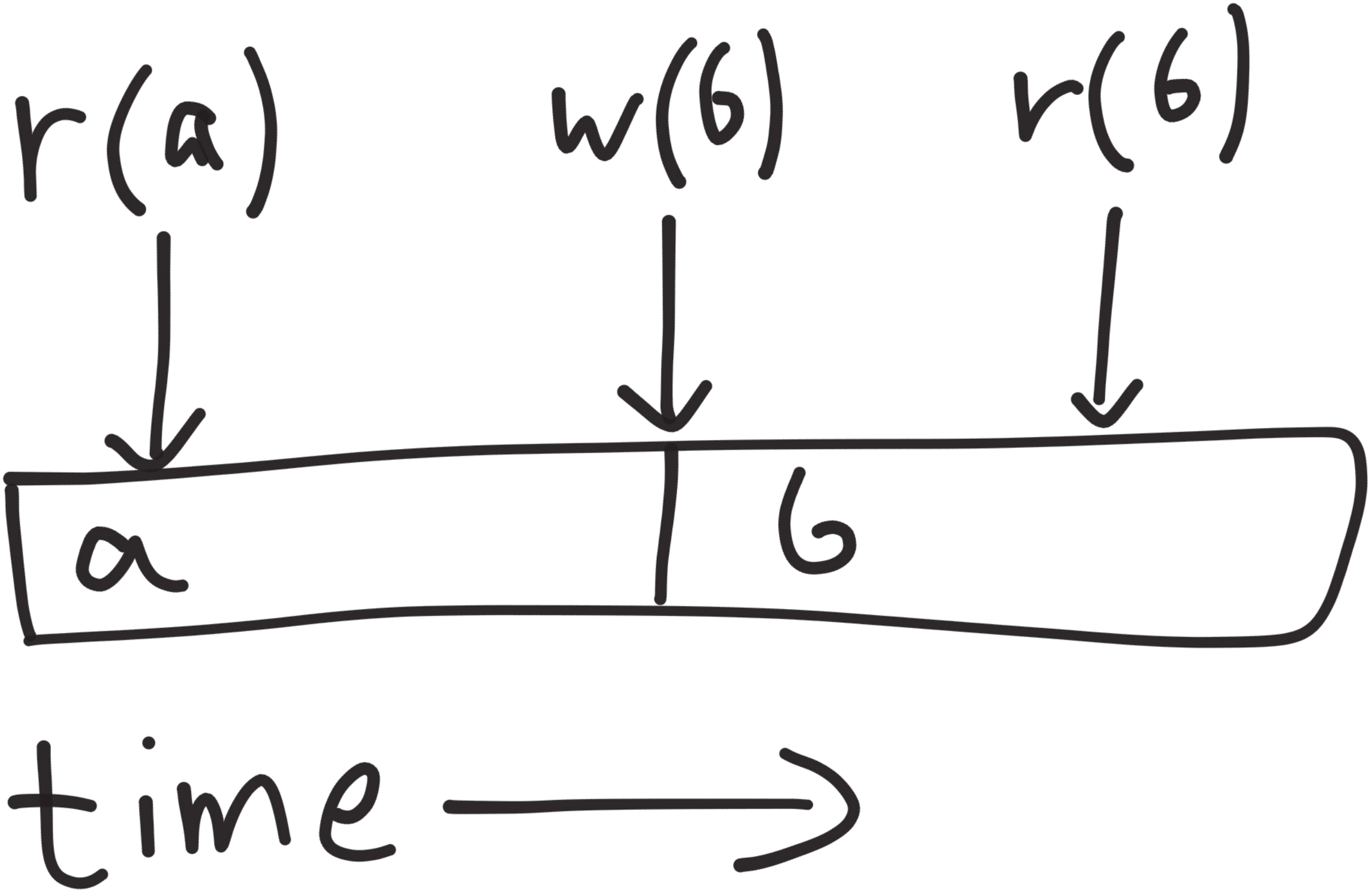
github.com/aphyr/jepsen

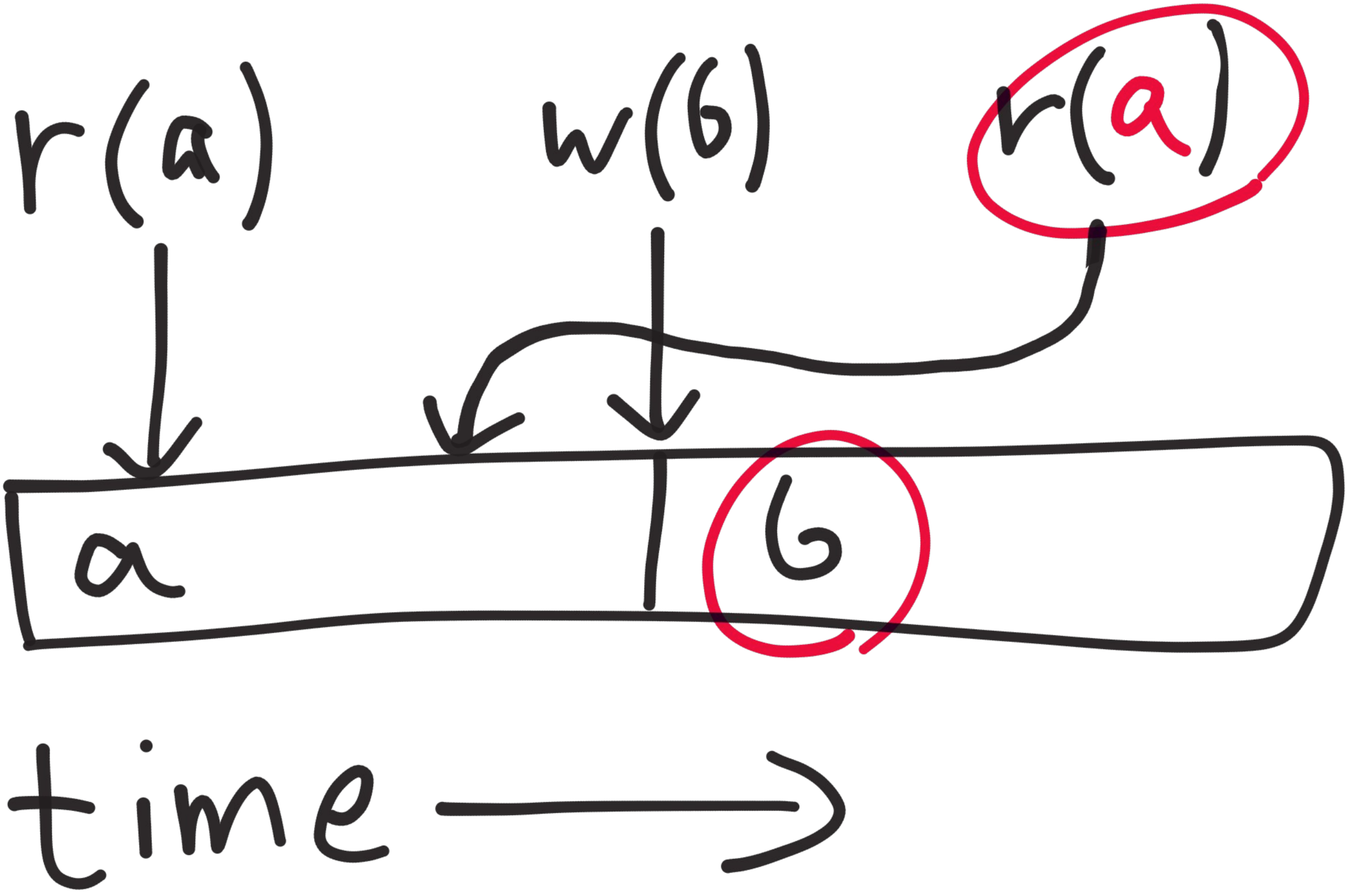

Environment
System

-INVARIANTS-

```
var x = a
print x
```

```
x = b
print x
```


r(a)
w(6)
r(6)
a
6
time


r(a)
w(6)
r(a)
a
6
time

Invariants

Constrain

Histories

What about

Concurrency?

r(a)

r(b)

a

b

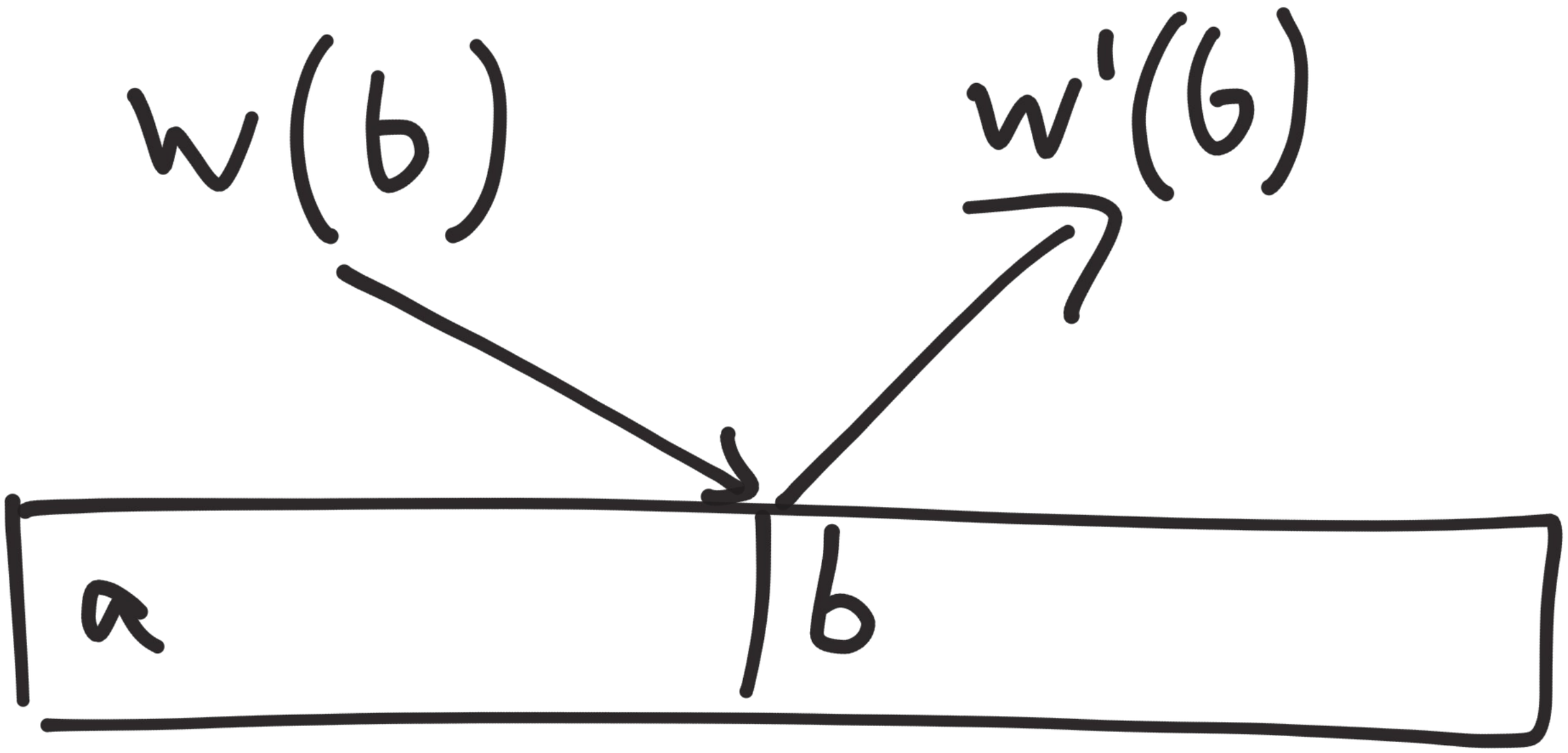
w(b)

r(b)

Distributed

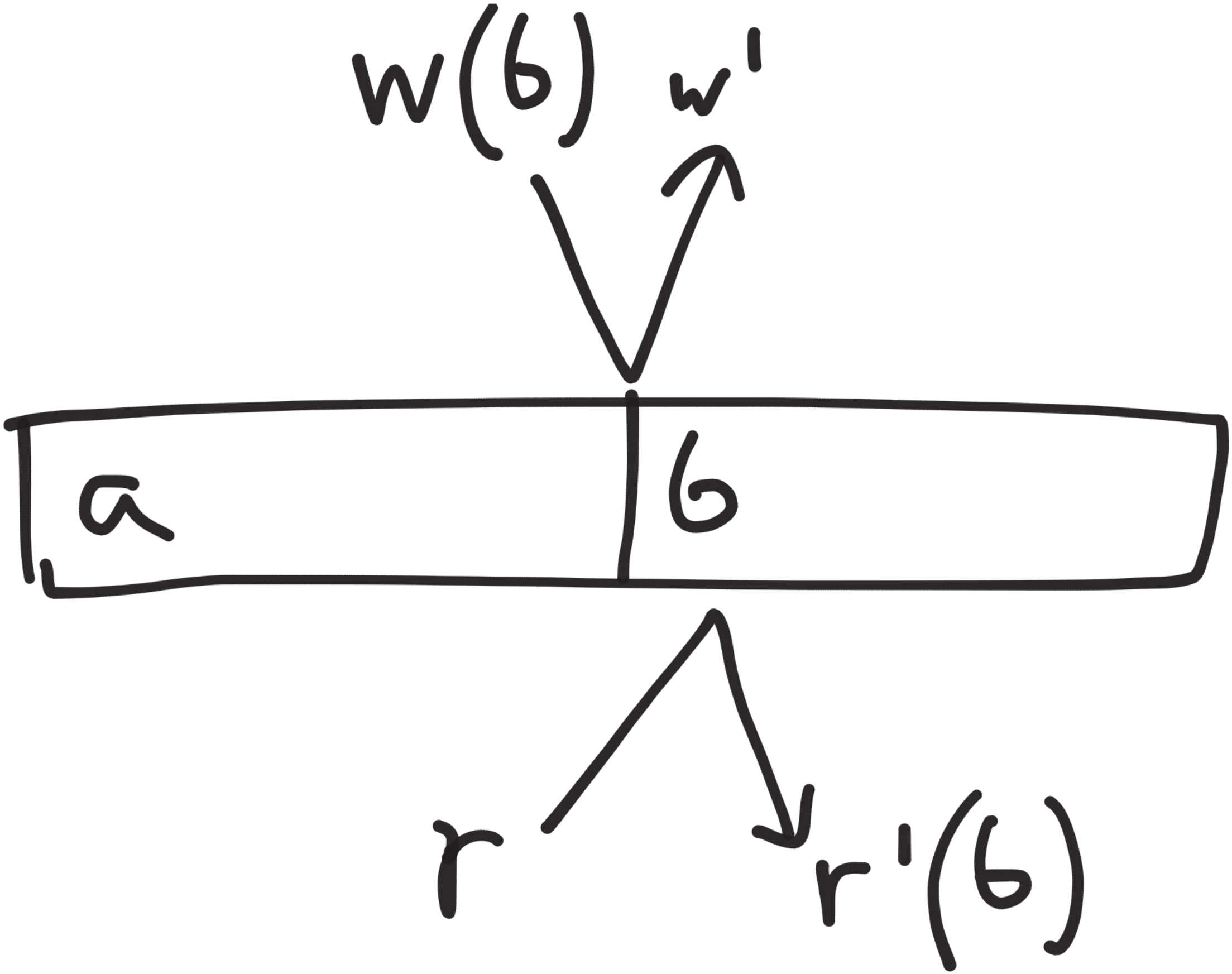
⇓

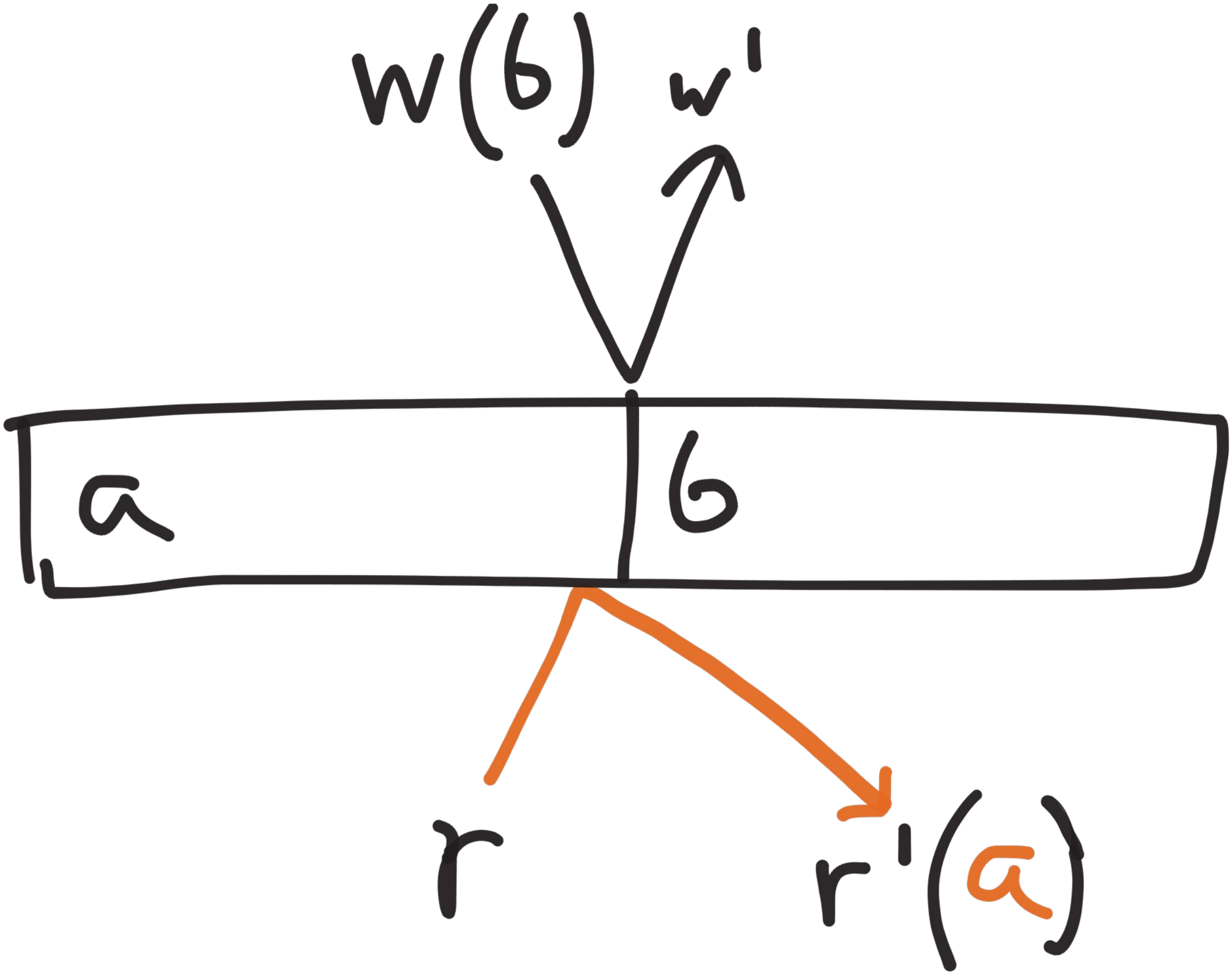
Distance



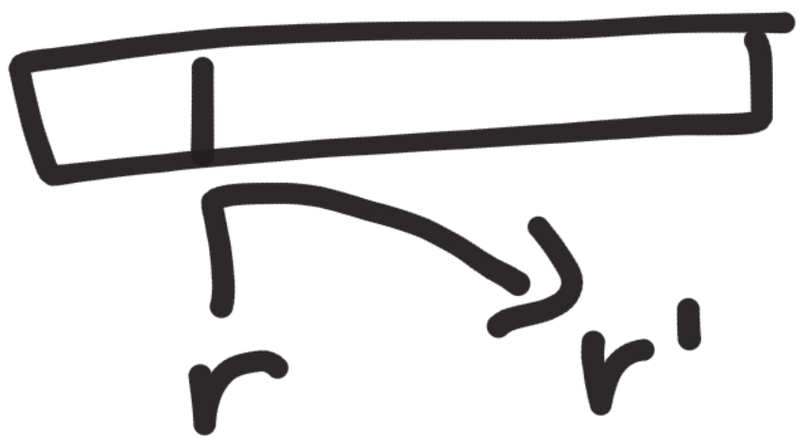
Messages take time

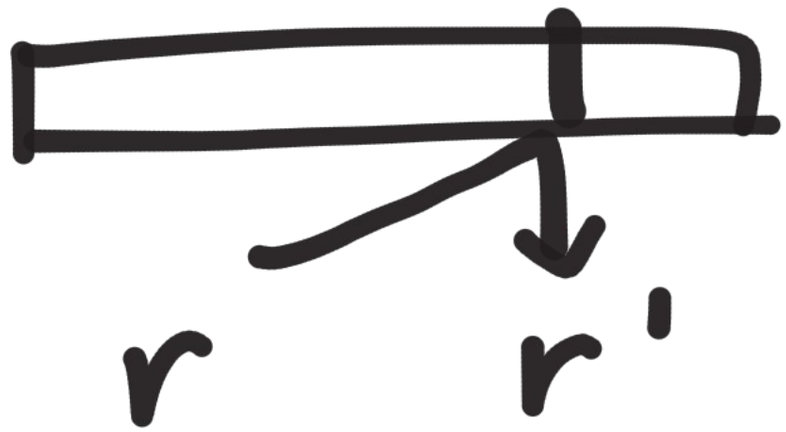
Delays imply ambignous orders

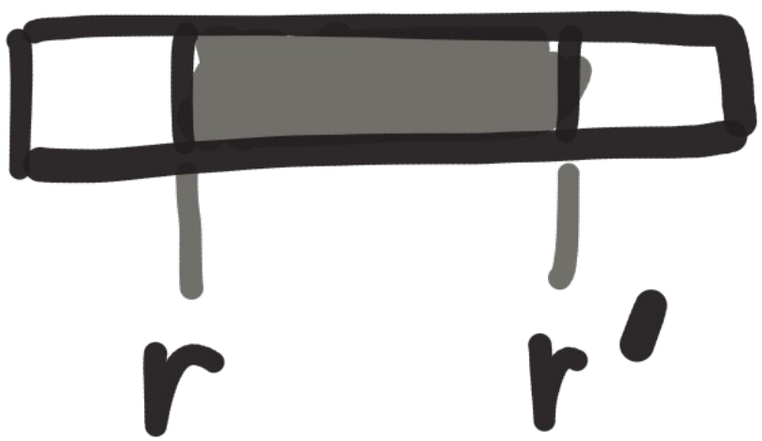

w(b)
w'
a
b
r
r'(b)


w(b)
w'
a
b
r
r'(a)

But there are finite bounds on ambiguity!

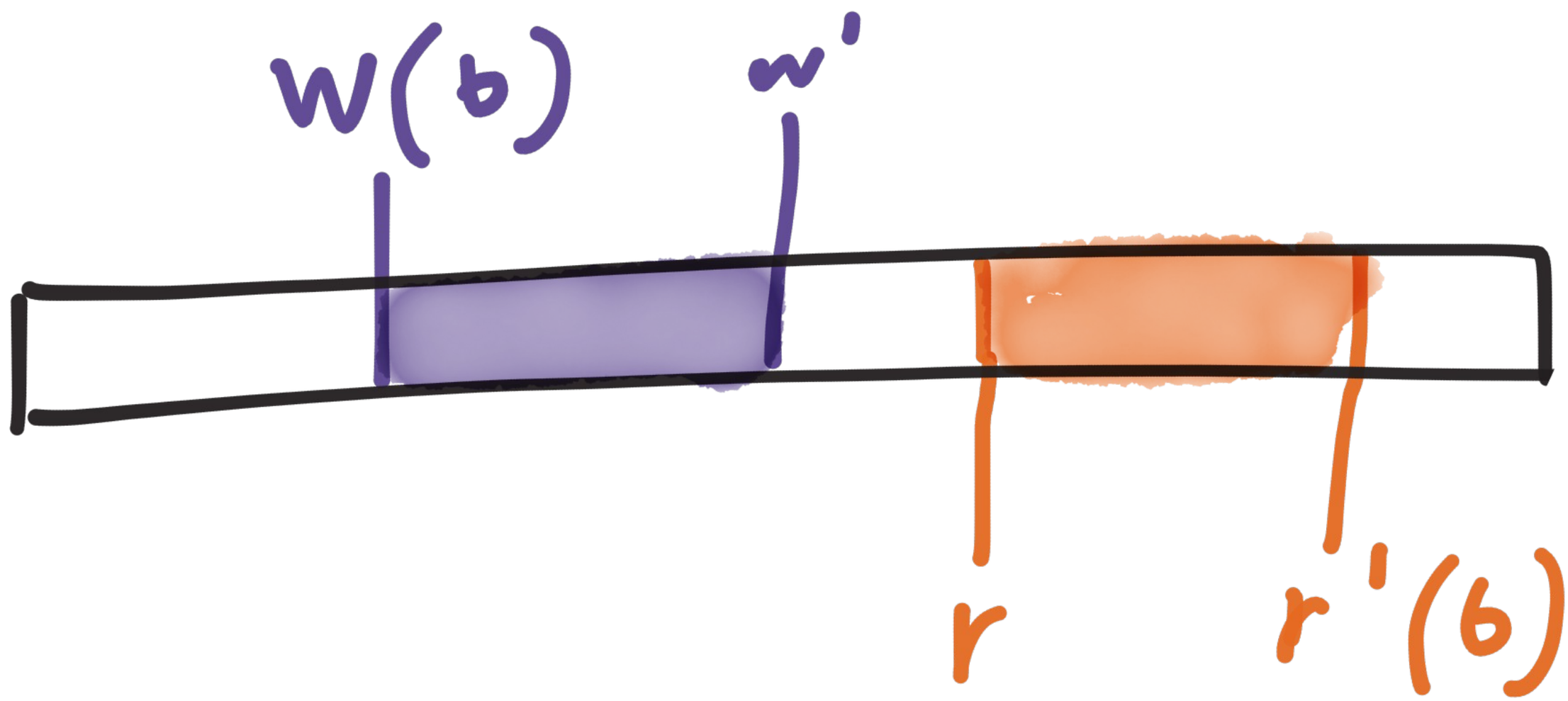






Linearizability

After op is complete, guaranteed visibility!



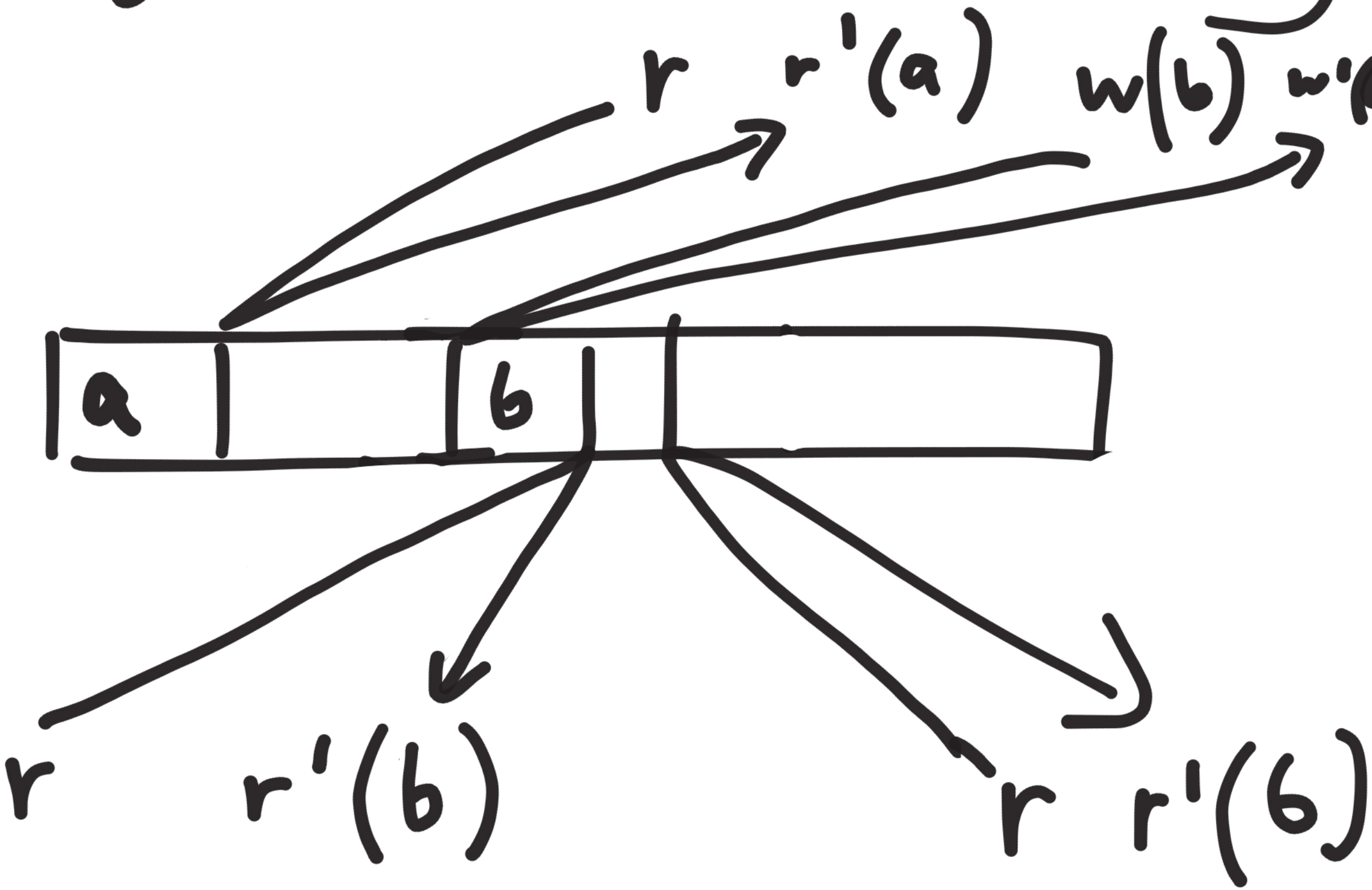
- Everyone agrees on order
- Ops are guaranteed to be visible to ALL clients once complete
- No stale reads

What if...

we relaxed the

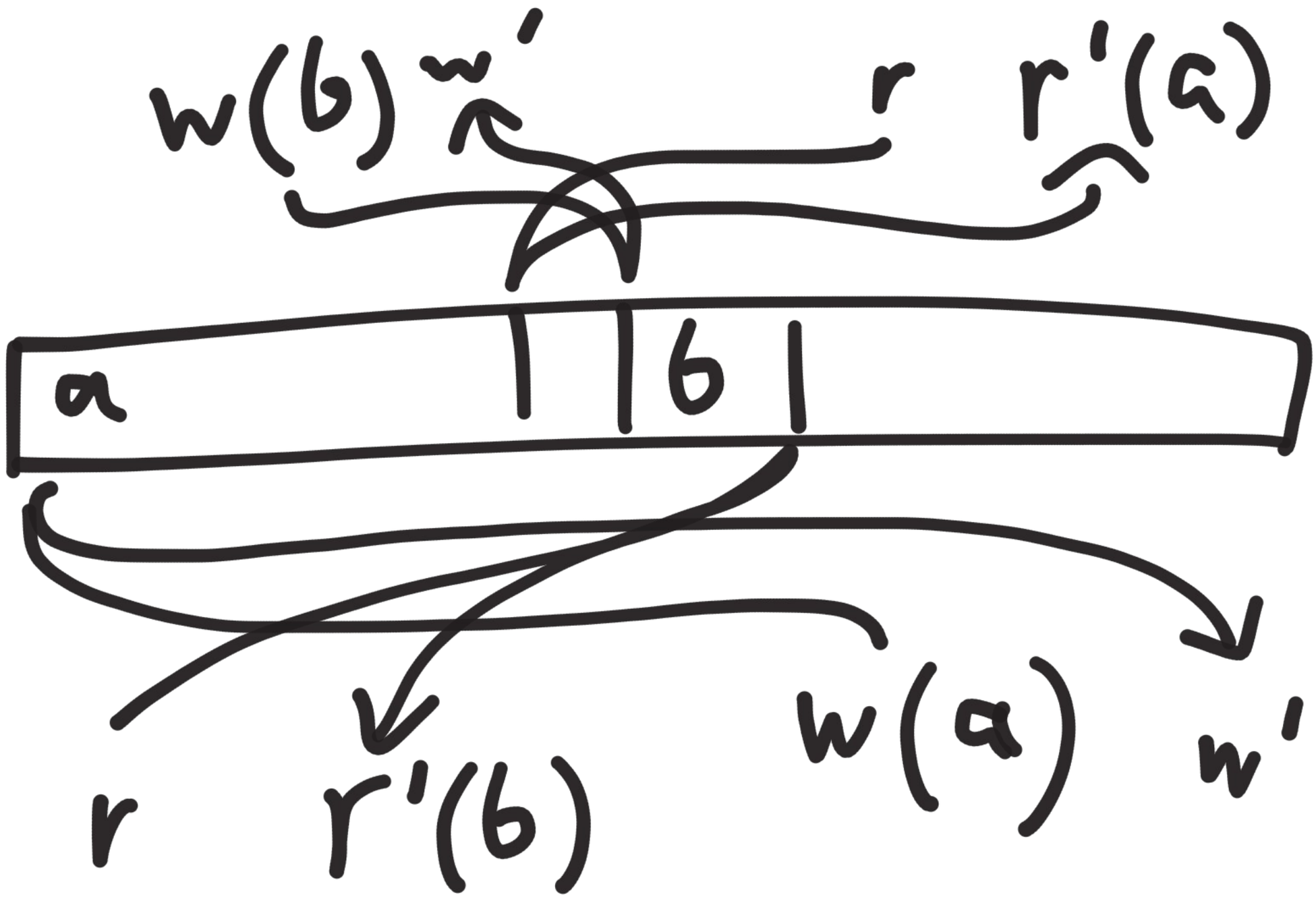
invariants?

Sequential Consistency



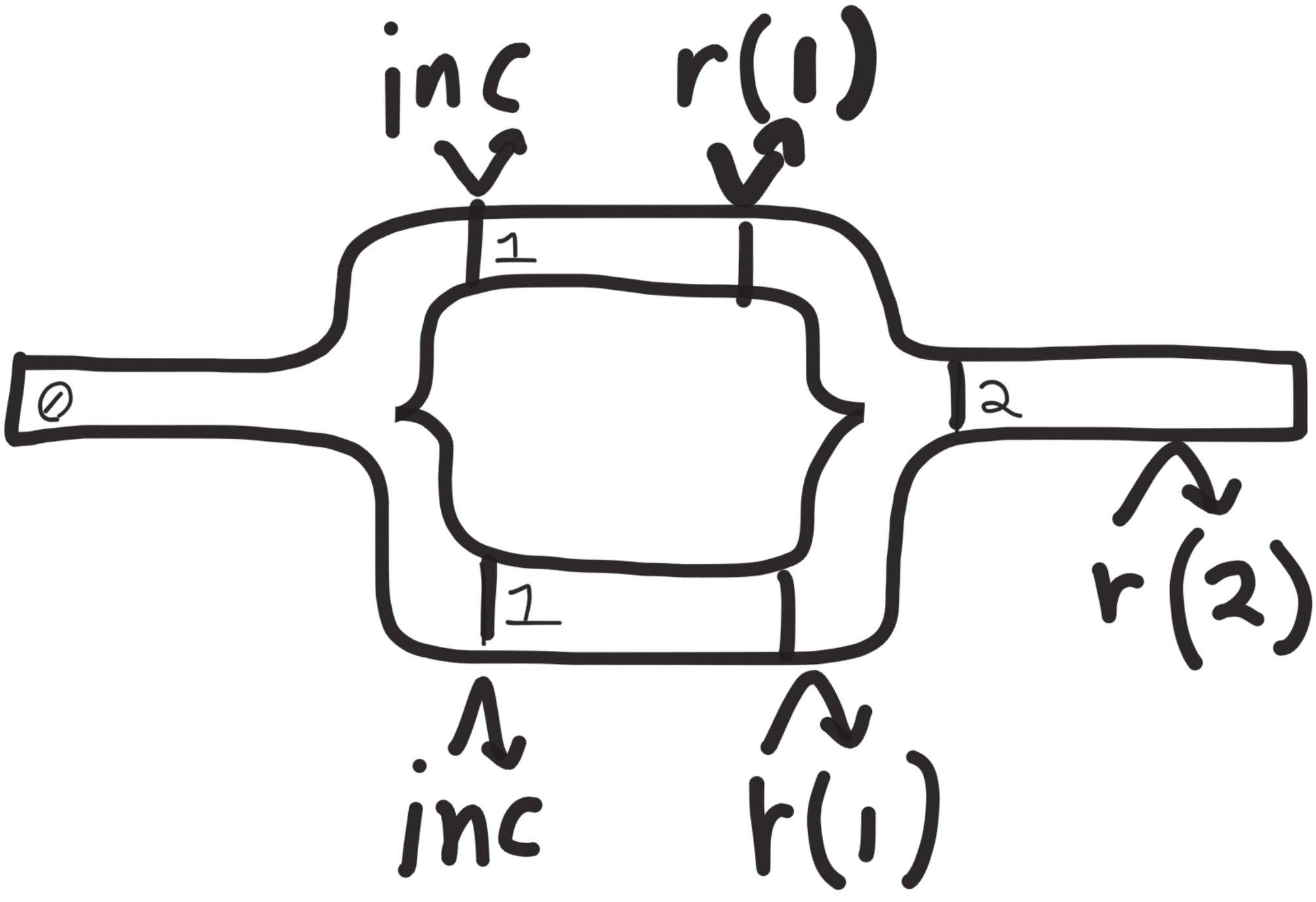
All processes agree on order of ops, but may be delayed in time.

Serializable Consistency



Ops happen in SOME order, but nobody has to agree on what that was.

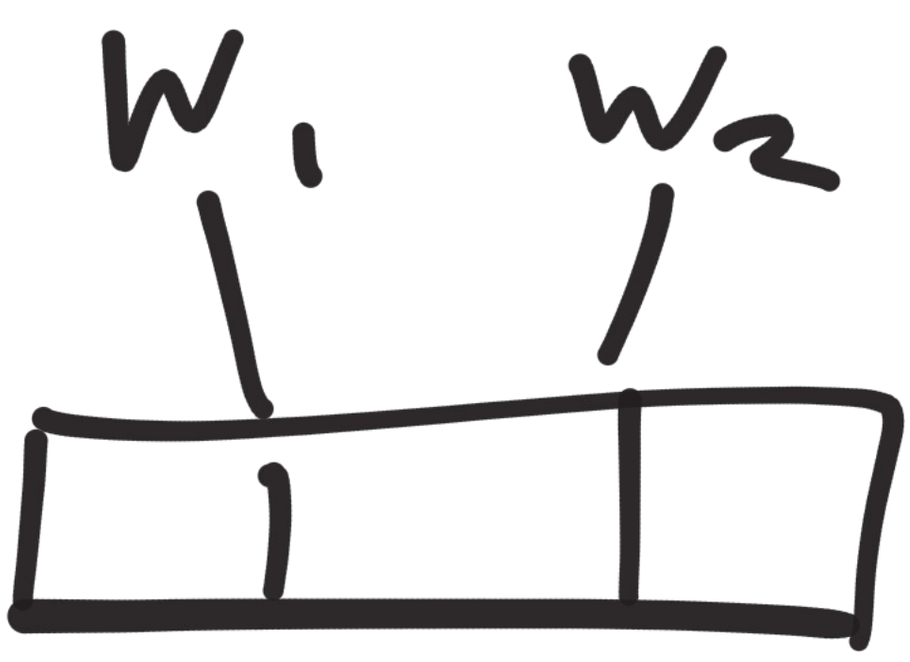
Eventual Consistency

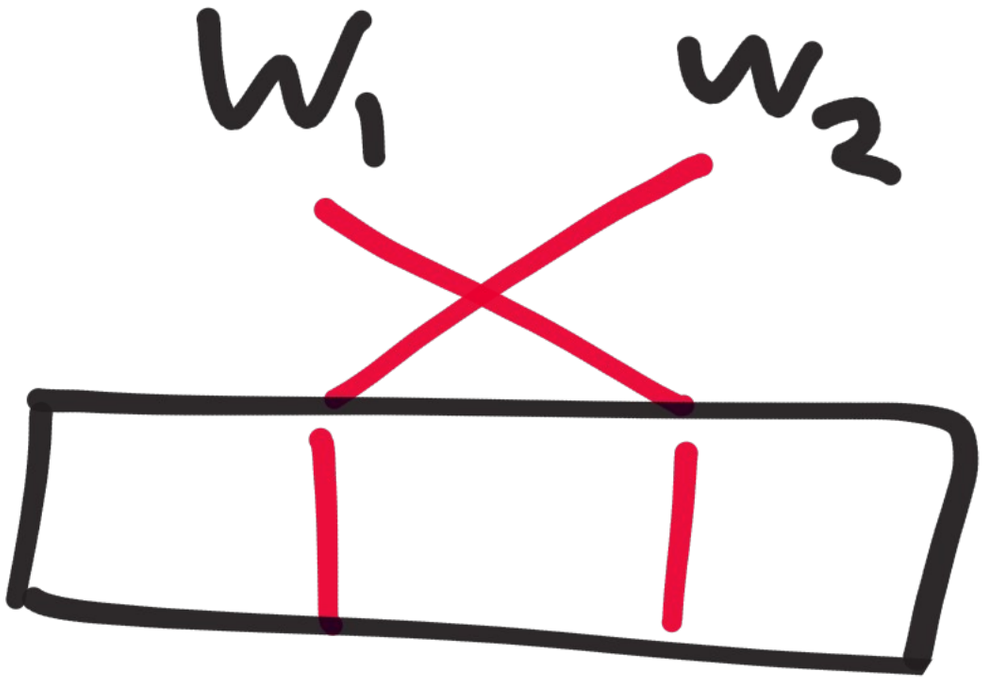


There doesnt have to be any single order so long as histories converge

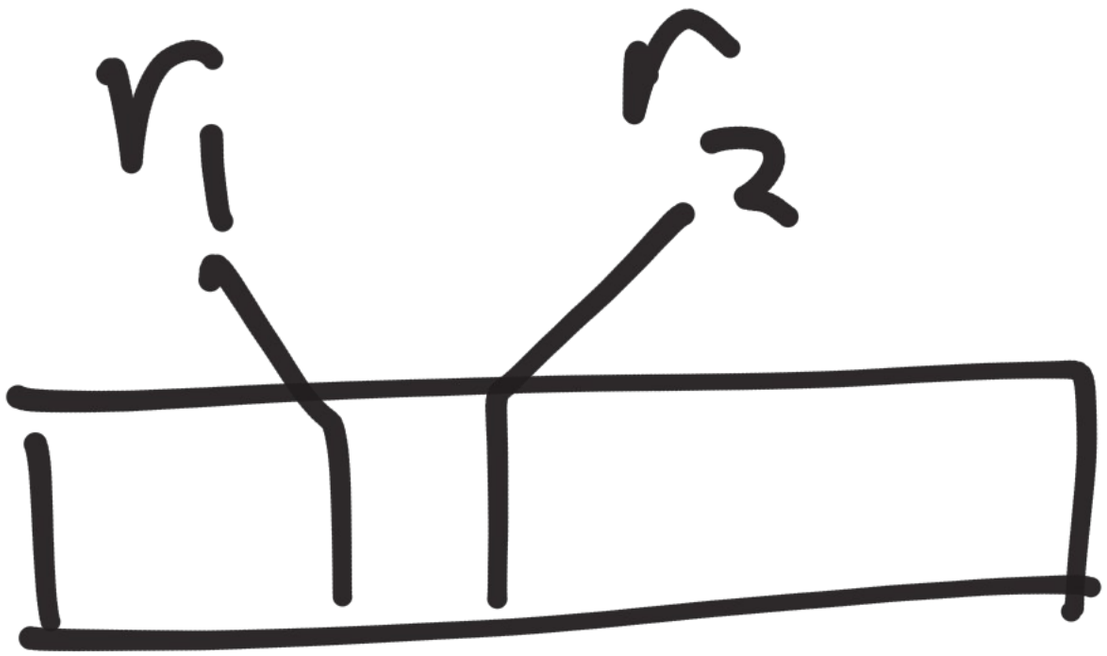
second op	first op: write	first op: read
write	MW	WFR
read	RYW	MR

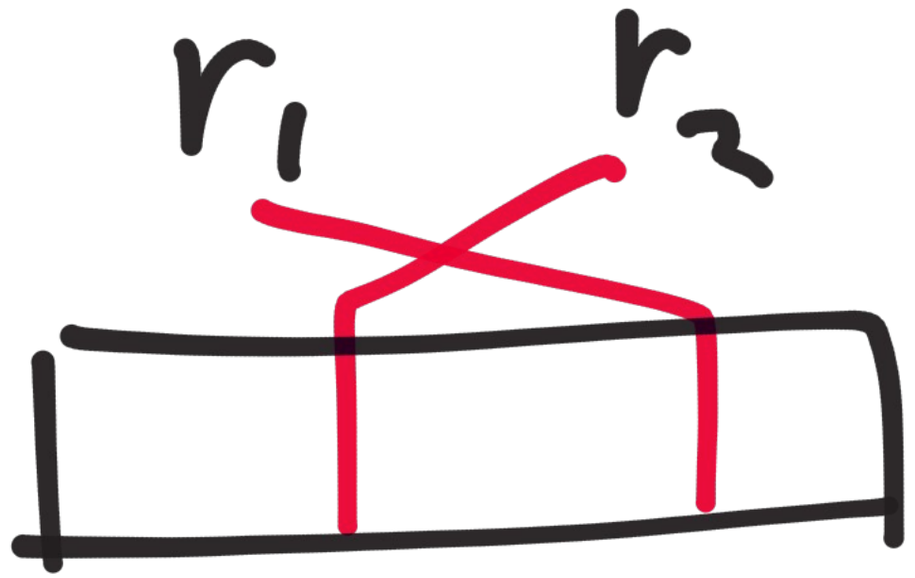
MW: Monotanic Write



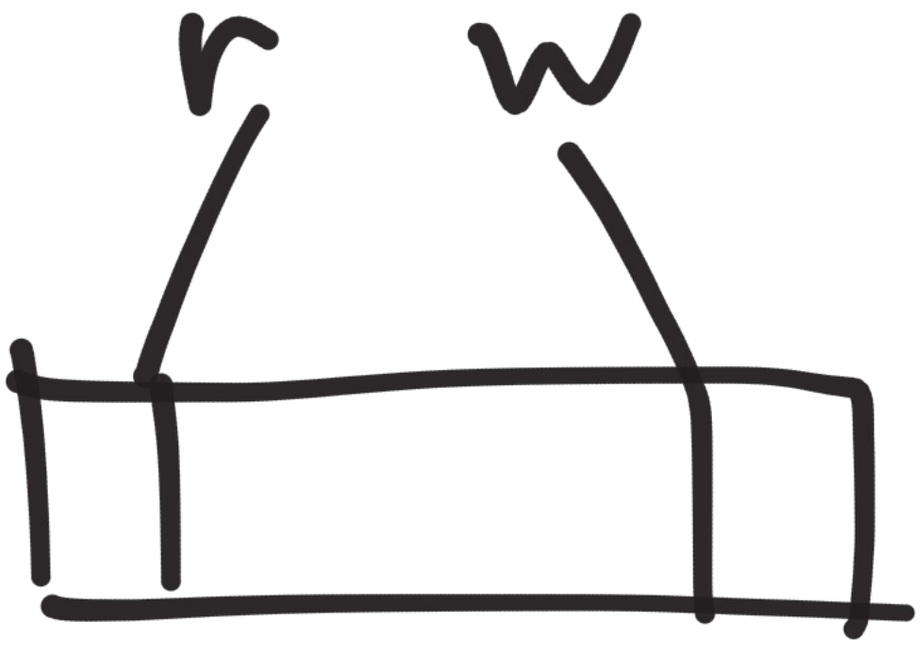


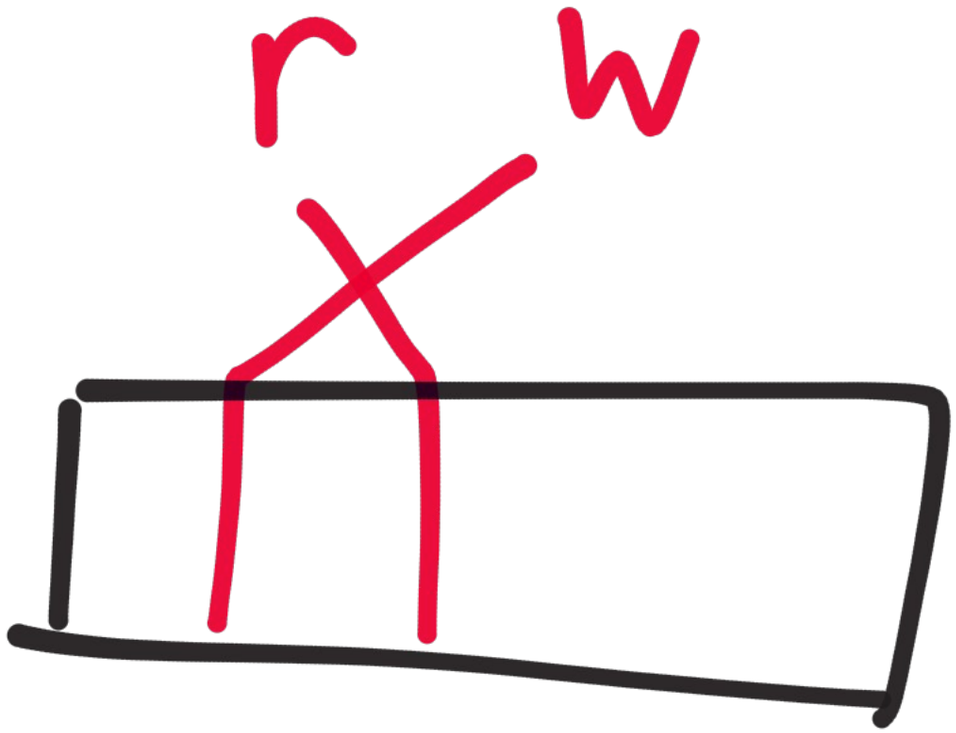
MR: Monotonic Read



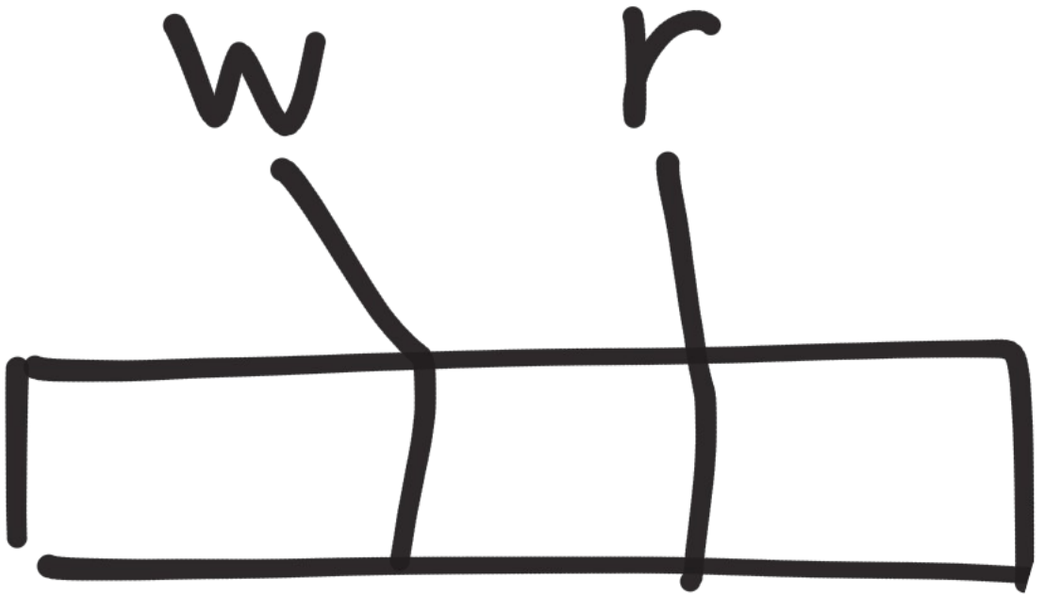


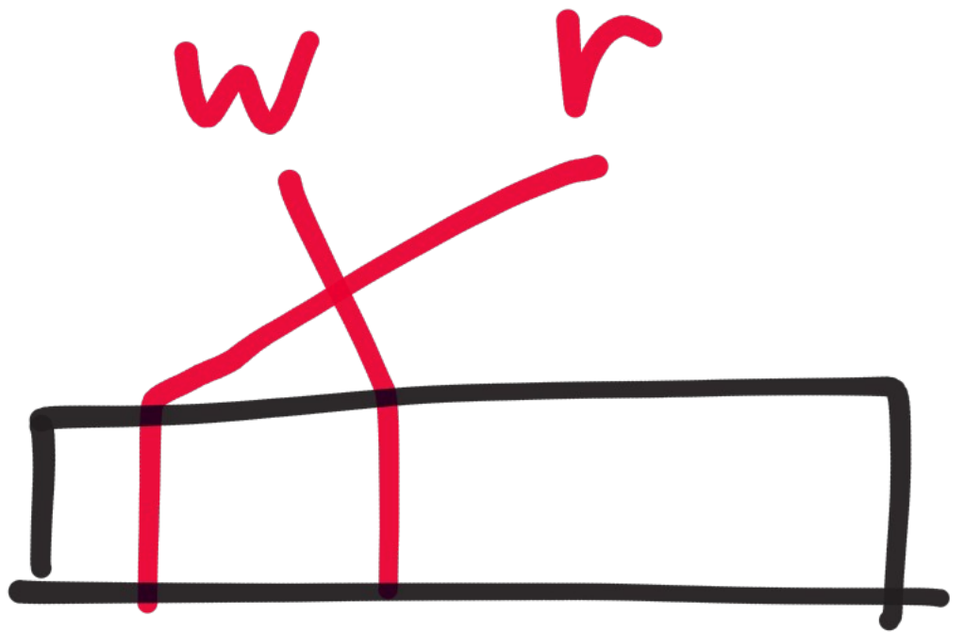
WFR: Writes Follow Reads





RYW: Read Your Writes

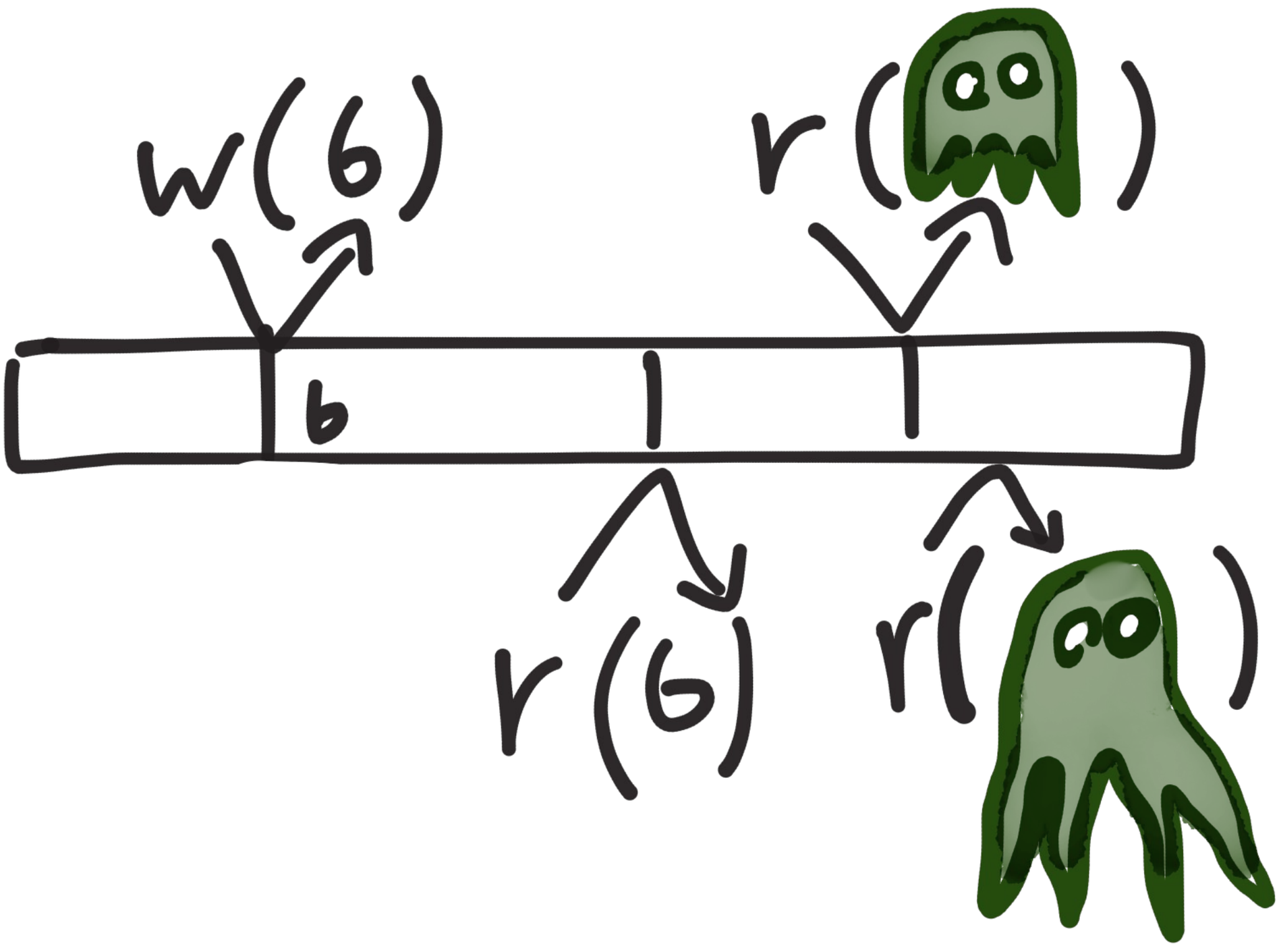




Infinitely many

Consistency classes

Cthuln Consistency



strong

serializable

linearizable

serializable

sequential

RR

SI

CS

causal

MAV

P-CI

PRAM

WFR

RC

RYW

MR

MW

"weaker" consistency models are more available in failure

"weaker" models are less intuitive

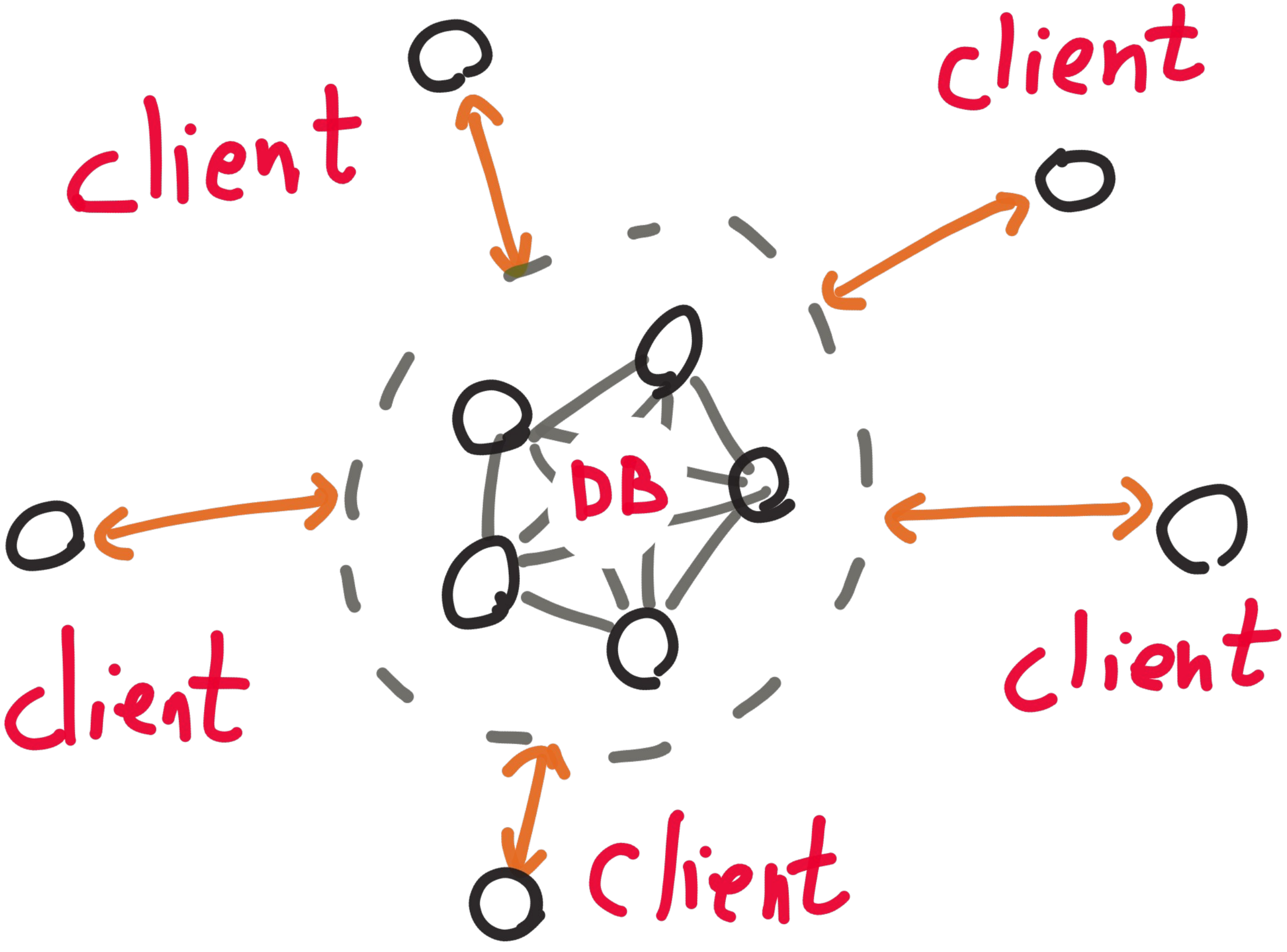
"weaker" models need less coordination, so they're faster!

→ CPU memory model

"weak" ≠ unsafe

- Diff algos require diff. levels of consistency

SO: Consistency models constrain the system-environment interactions.

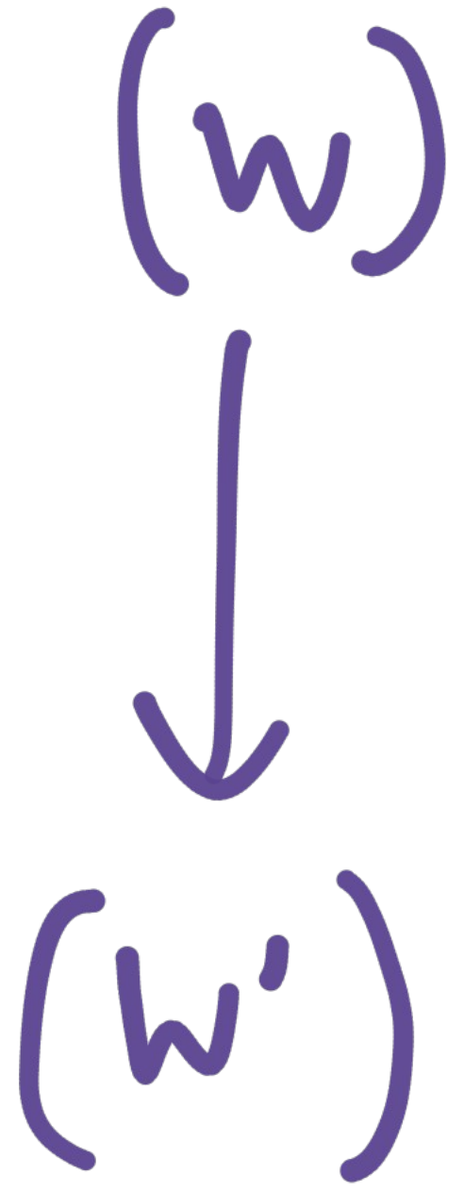

client
client
DB
client
client
client

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

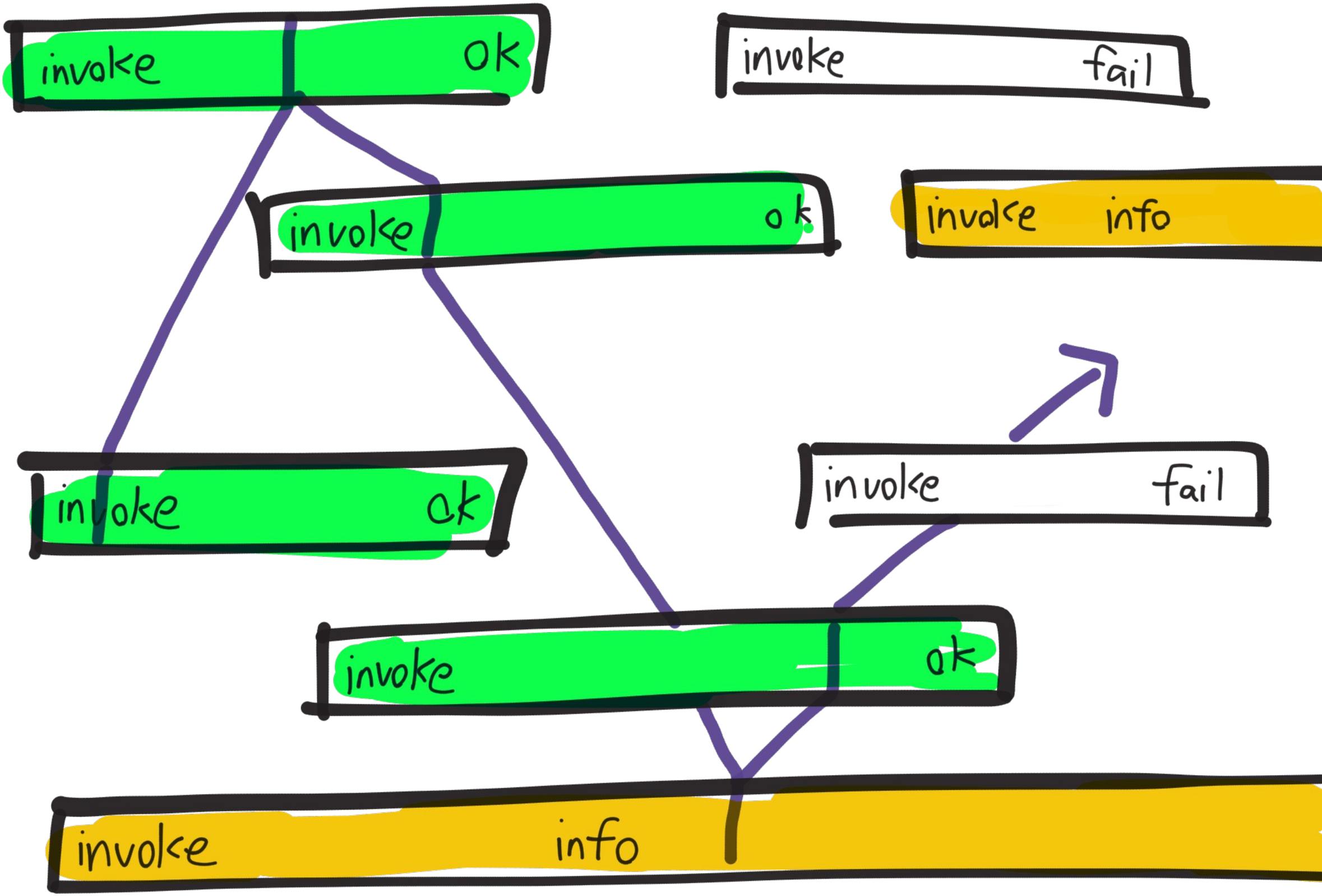
invoke info

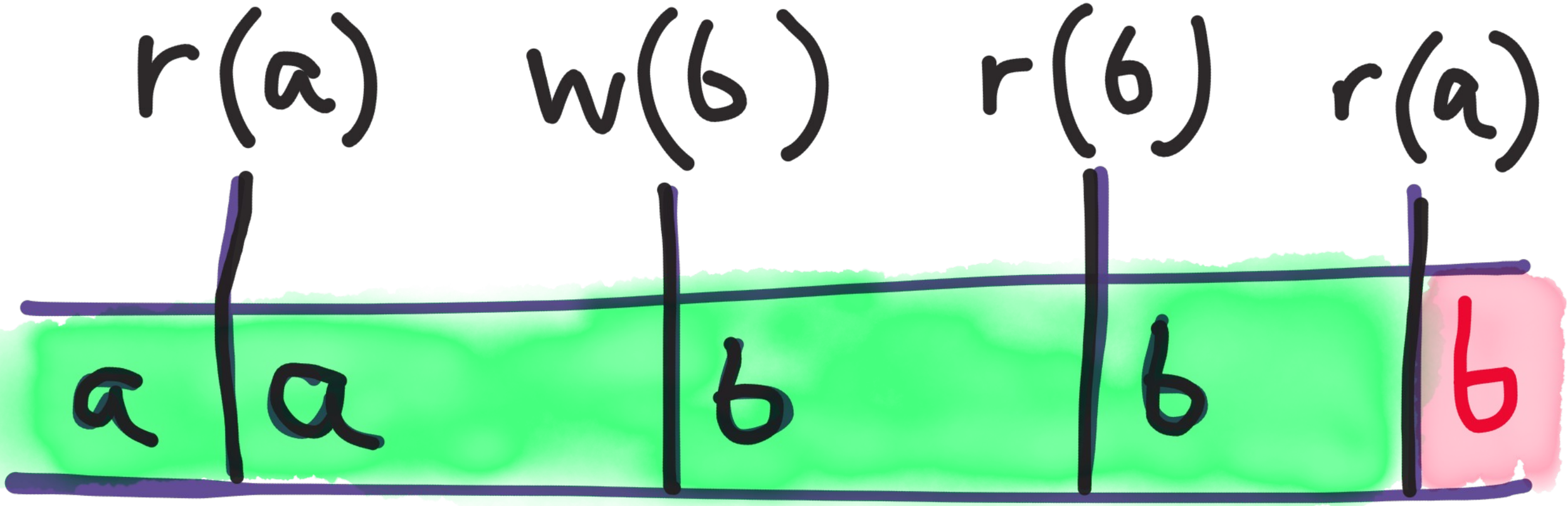
invoke ok

invoke fail

invoke ok

invoke info


invoke
ok
invoke
fail
invoke
ok
invoke
info
invoke
ok
invoke
fail
invoke
ok
invoke
info



Use a singlethreaded model to verify path

```
(defrecord Register [x]
  (step [r op]
    (case (:f op)
      :write (Register. (:value op))
      :read (if (= x (:value op))
              r
              (inconsistent "..."))))
```

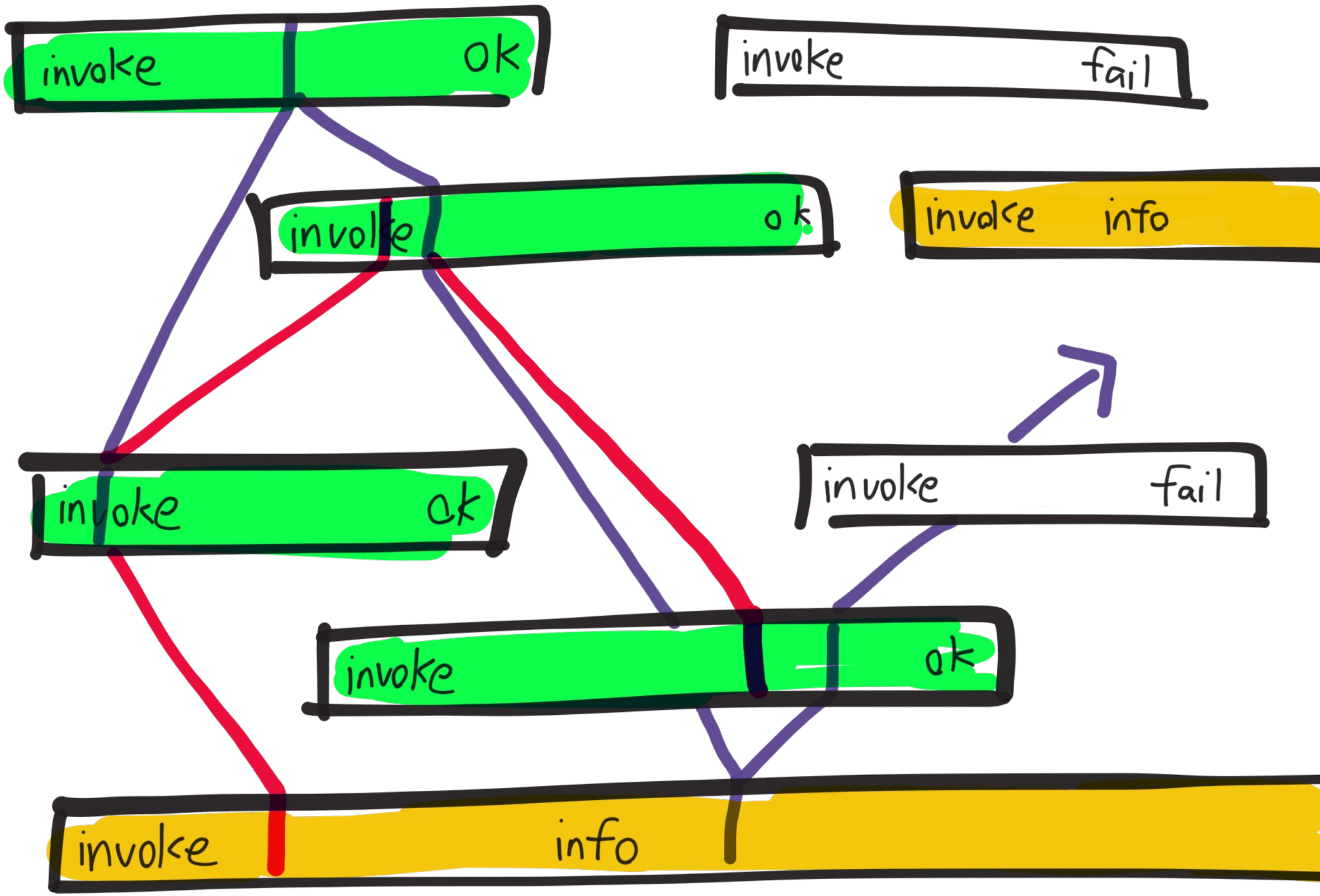
r

r'(2)

3

inconsistent

(step (Register. 3)
{:f :read, :value 2})


invoke
ok
invoke
fail
invoke
ok
invoke
info
invoke
ok
invoke
fail
invoke
ok
invoke
info

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

“So, what have you found?”

2013

- Redis Sentinel
- MongoDB
- Riak

2013

- Kafka
- NuoDB
- Cassandra

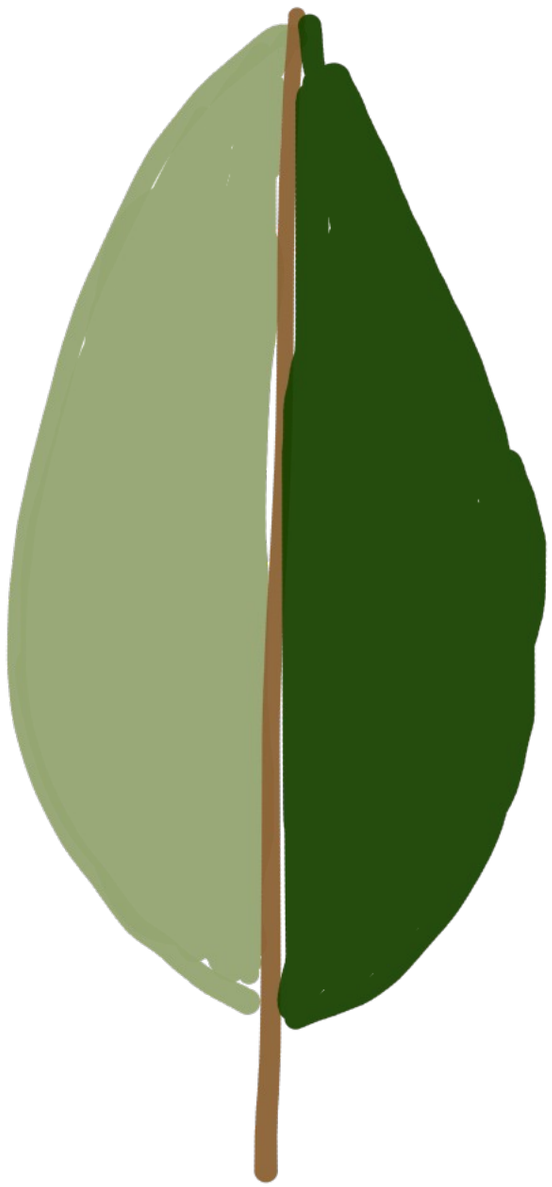
2014

- RabbitMQ
- etcd $ Consul
- Elasticsearch

And now, for something

COMPLETELY DIFFERENT

THE SAME



mongoDB

First Jepsen test in May 2013

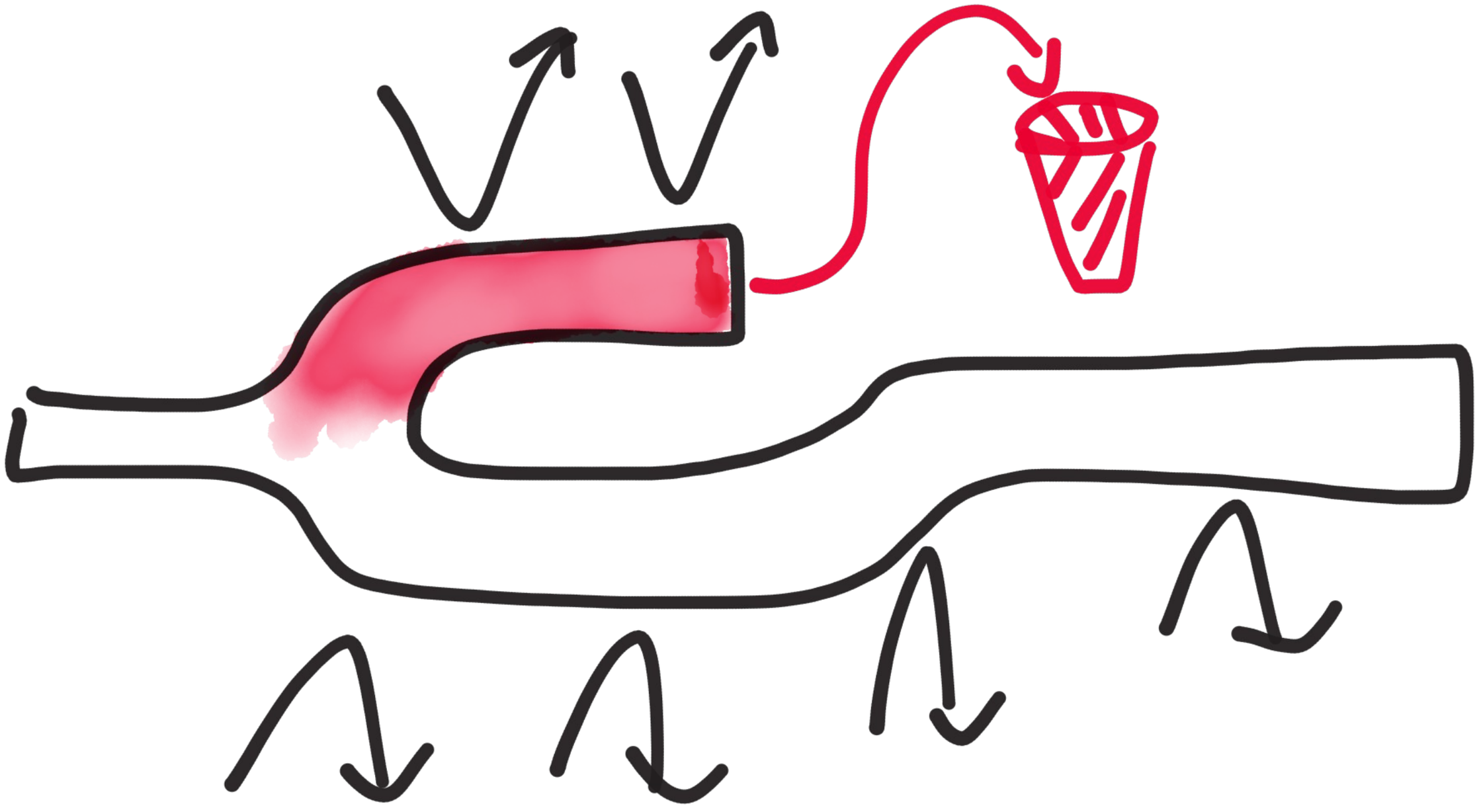
Data loss at all WriteConcern levels

Unsafe defaults

- Some clients didnt check for errors at all

Unsafe defaults

- Better clients only wait for 1 or nodes, instead of a majority!



Rollbacks

Even Majority unsafe!

Network failures were considered successful responses

But those bugs are fixed now!

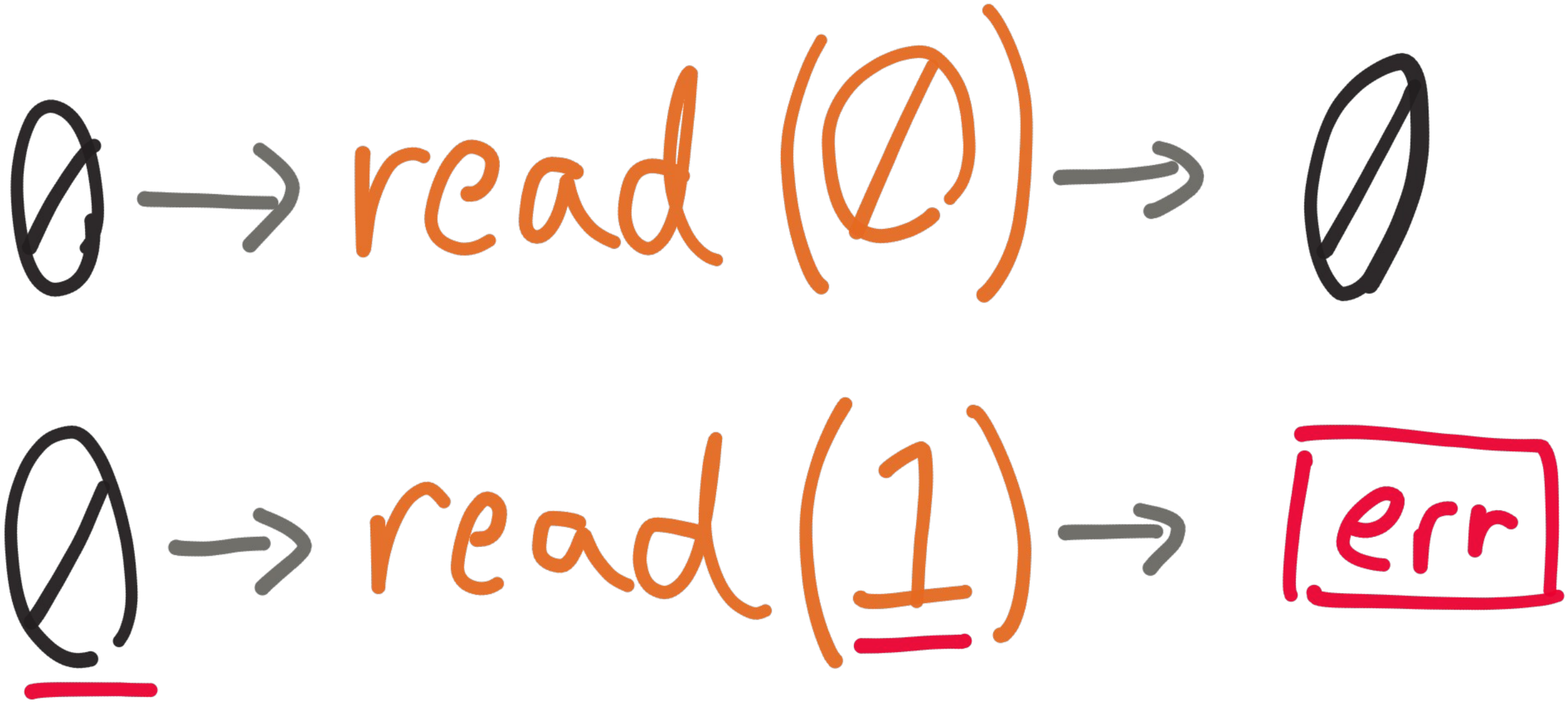
Ok! Let's

try 2.6.7

Linearizable

CaS Registers


∅ → write(1) → 1


∅ → read(∅) → ∅
∅ → read(1) → err

$$\emptyset \to cas(\emptyset, 1) \to 1$$

$$\underline{\emptyset} \to cas(\underline{1}, 2) \to \boxed{err}$$

An Anomaly Emerges!

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

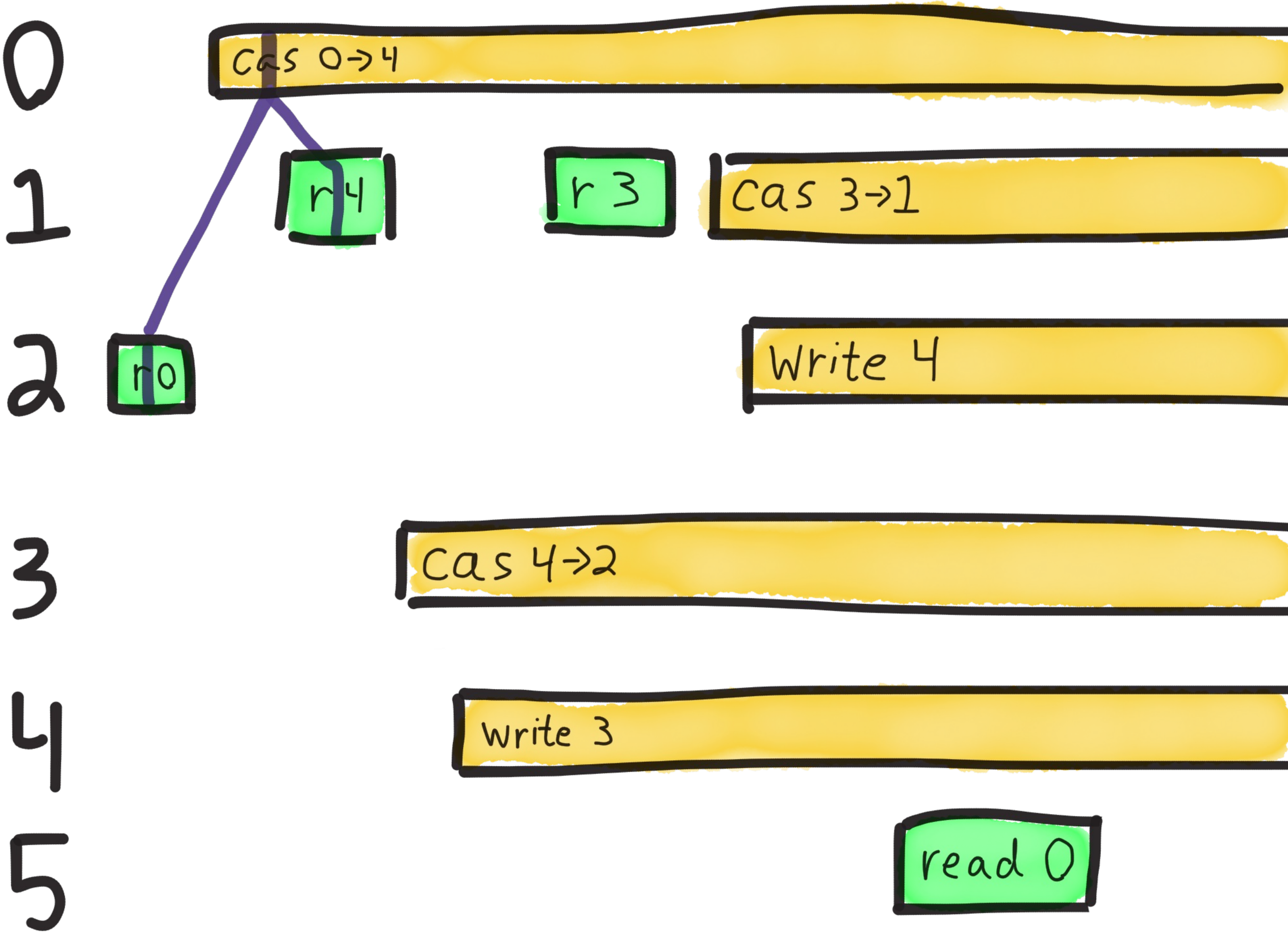
cas 4→2

4

write 3

5

read 0


0
cas 0→4
1
r4
r3
cas 3→1
2
r0
Write 4
3
cas 4→2
4
write 3
5
read 0

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

Cas 4→2

4

write 3

5

read 0

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

Cas 4→2

4

write 3

5

read 0

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

Cas 4→2

4

write 3

5

read 0

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

Cas 4→2

4

Write 3

5

read 0

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

Cas 4→2

4

write 3

5

read 0

0

cas 0→4

1

r4

r3

cas 3→1

2

r0

Write 4

3

Cas 4→2

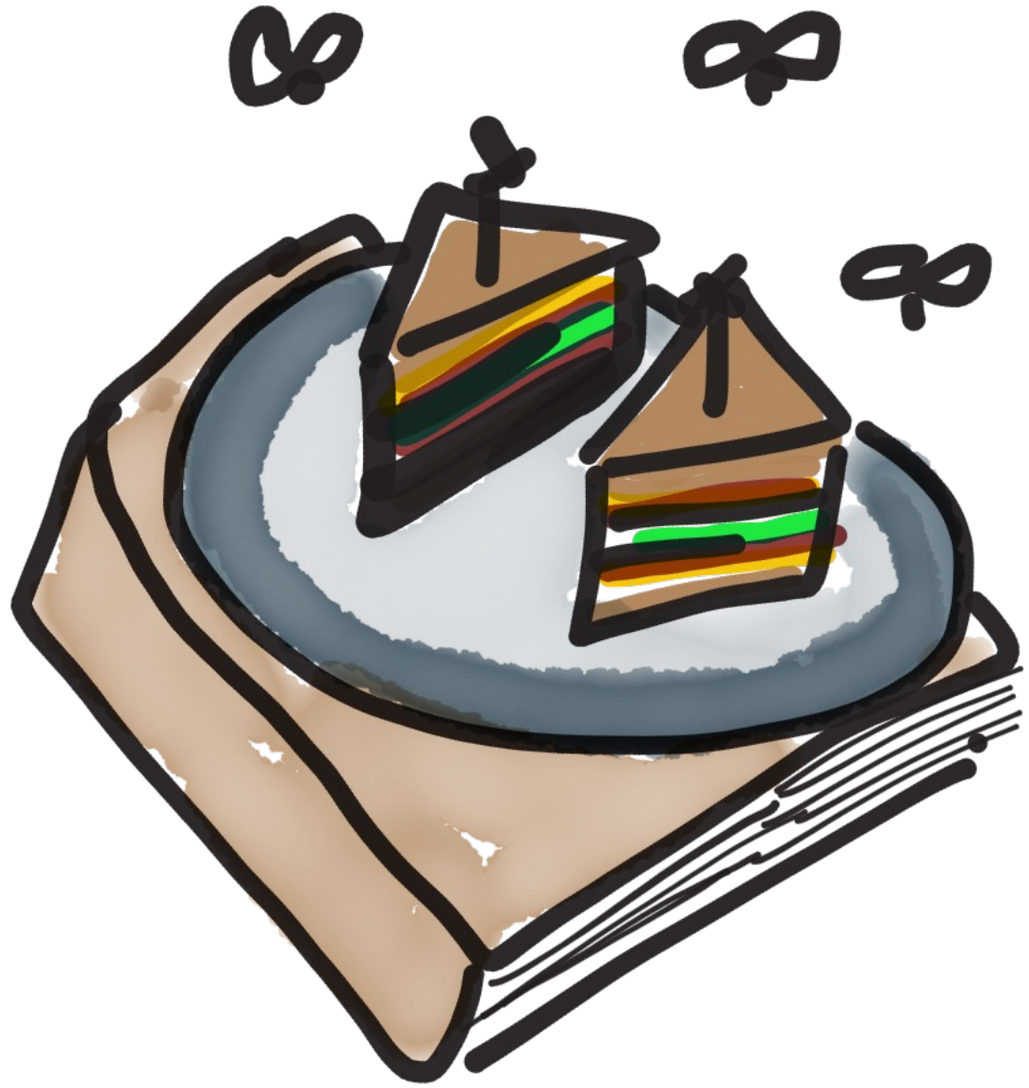
4

write 3

5

read 0

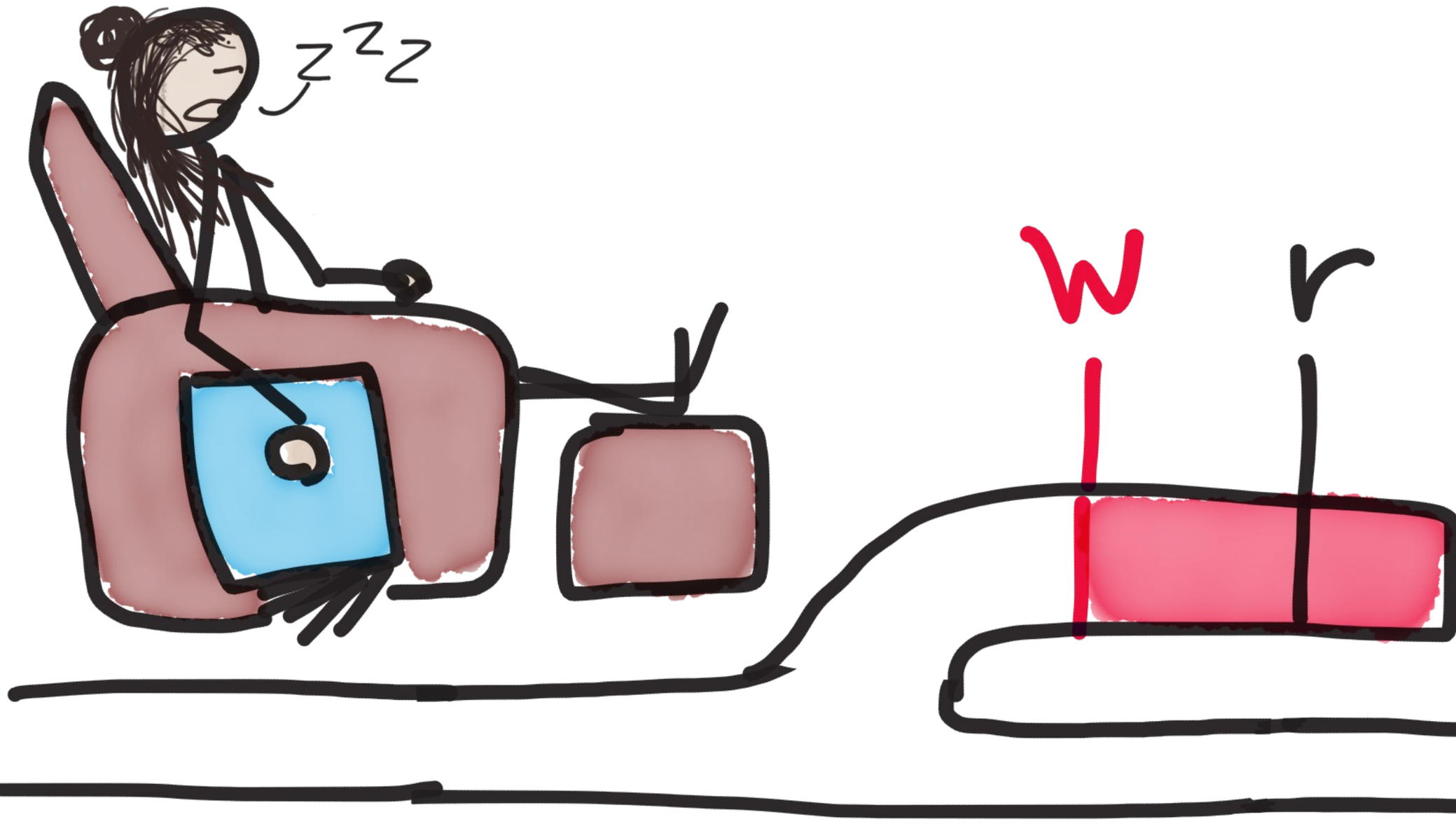
Stale Reads



w

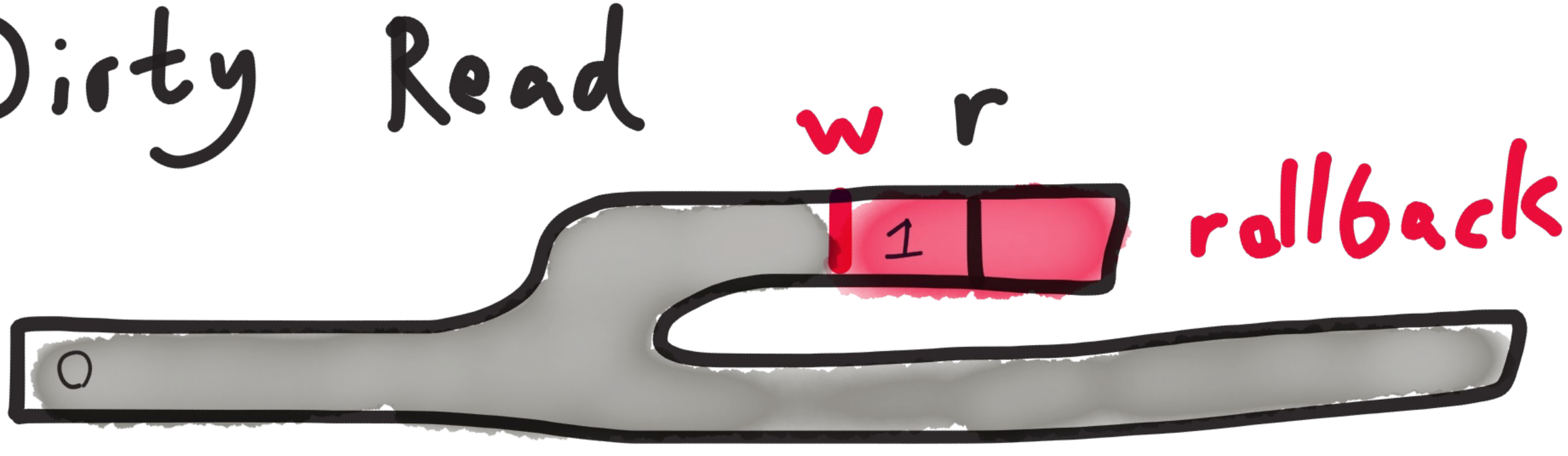
r

Read Uncommitted

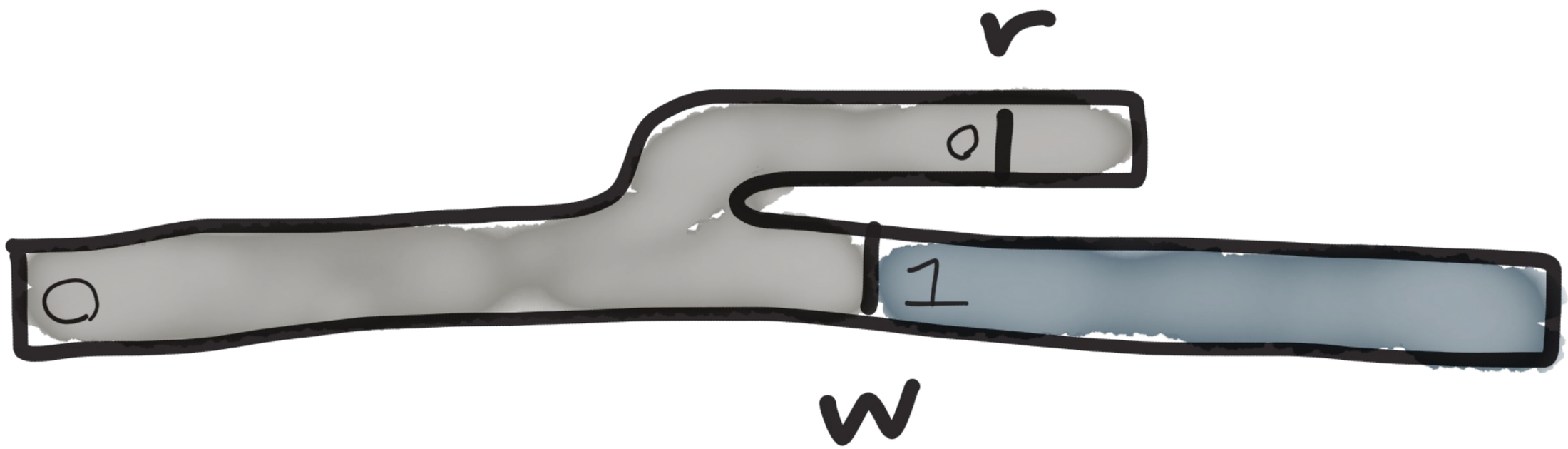


Server-17975

Dirty Read



Stale Read



strong

linearizable

sequential

causal

PRAM

RYW

MR

MW

WFR

serializable

serializable

RR

CS

SI

MAV

P-CI

RC

strong

serializable

linearizable

sequential

causal

PRAM

RYW

MR

MW

WFR

RR

CS

RC

SI

MAV

P-CI

strong

serializable

linearizable

serializable

sequential

RR

SI

CS

Causal

MAV

P-CI

PRAM

WFR

IRC

RYW

MR

MW

RU

strong

serializable

linearizable

serializable

sequential

RR

SI

CS

Causal

MAV

P-CI

PRAM

WFR

IRC

RYW

MR

MW

RU

Mongo

strong

serializable

linearizable

sequential

causal

PRAM

RYW

MR

MW

WFR

RC

RU

CS

RR

serializable

SI

MAV

P-CI

Mongo

vs. docs

Mongo Recommendations

- Plan for stale reads
- Use CaS to verify read values are legitimate

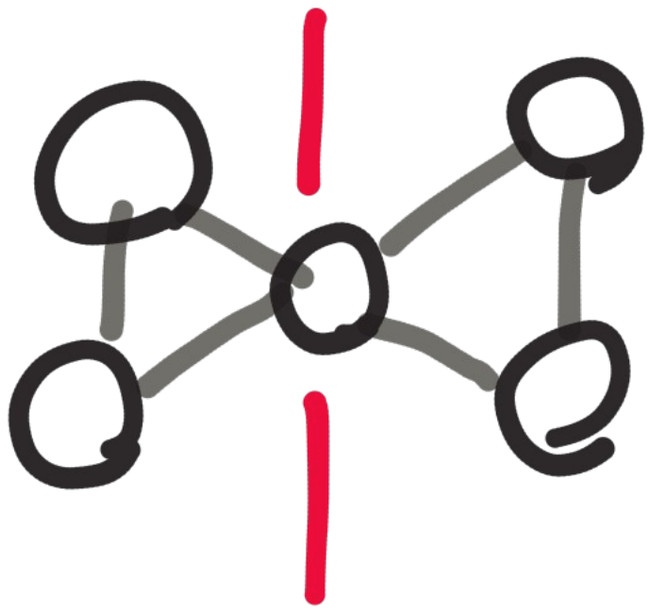
elastic

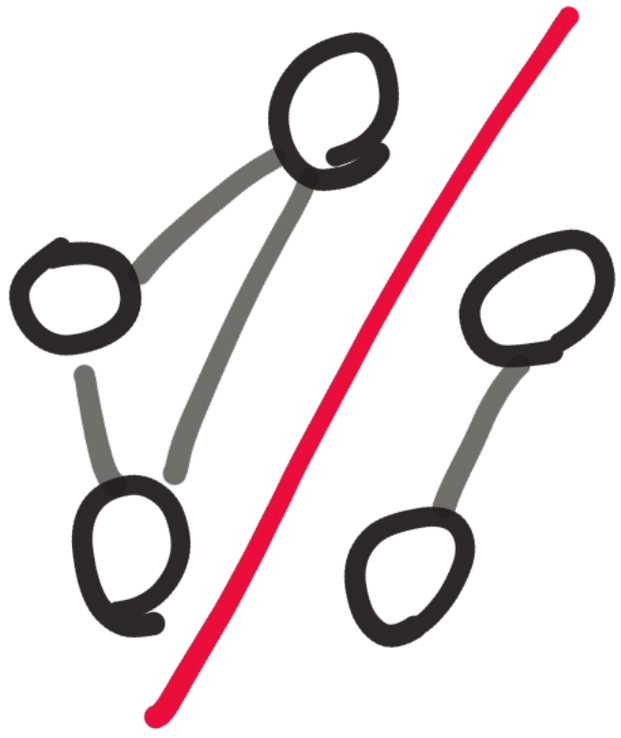
[search]

First Elasticsearch tests: June 2014

v1.1.0

Split Brain from





Massive data loss: 90%

- Cluster status lies
- 90 seconds for recovery
- Permanently wedged clusters

ES added a great page on safety issues

Loss of documents during network partition (STATUS: ONGOING)

edit

If a network partition separates a node from the master, there is some window of time before the node detects it. The length of the window is dependent on the type of the partition. This window is extremely small if a socket is broken. More adversarial partitions, for example, silently dropping requests without breaking the socket can take longer (up to 3x30s using current defaults).

If the node hosts a primary shard at the moment of partition, and ends up being isolated from the cluster (which could have resulted in split-brain before), some documents that are being indexed into the primary may be lost if they fail to reach one of the allocated replicas (due to the partition) and that replica is later promoted to primary by the master. #7572

A test to replicate this condition was added in #7493.

bleskes commented on Sep 1, 2014 Owner

I'm closing this issue, as it is solved, as specified, by the changes made in #7493. Of course, there is more work to be done and the effort continues.

Thx for all the input and discussion.

bleskes closed this on Sep 1, 2014

AeroNotix commented on Sep 1, 2014

@bleskes so it's 100% fixed?

bleskes commented on Sep 1, 2014 Owner

this issue (partial network splits causing split brain) is fixed now, yes.

Folks are still referring to the last talk

Mark Walkom
@warkolm

Follow

@falican durability in ES isn't a major problem these days, take a look at elasticsearch.org/guide/en/elast… for info on our view

9:36 PM - 20 Feb 2015

Reply to @warkolm @falican

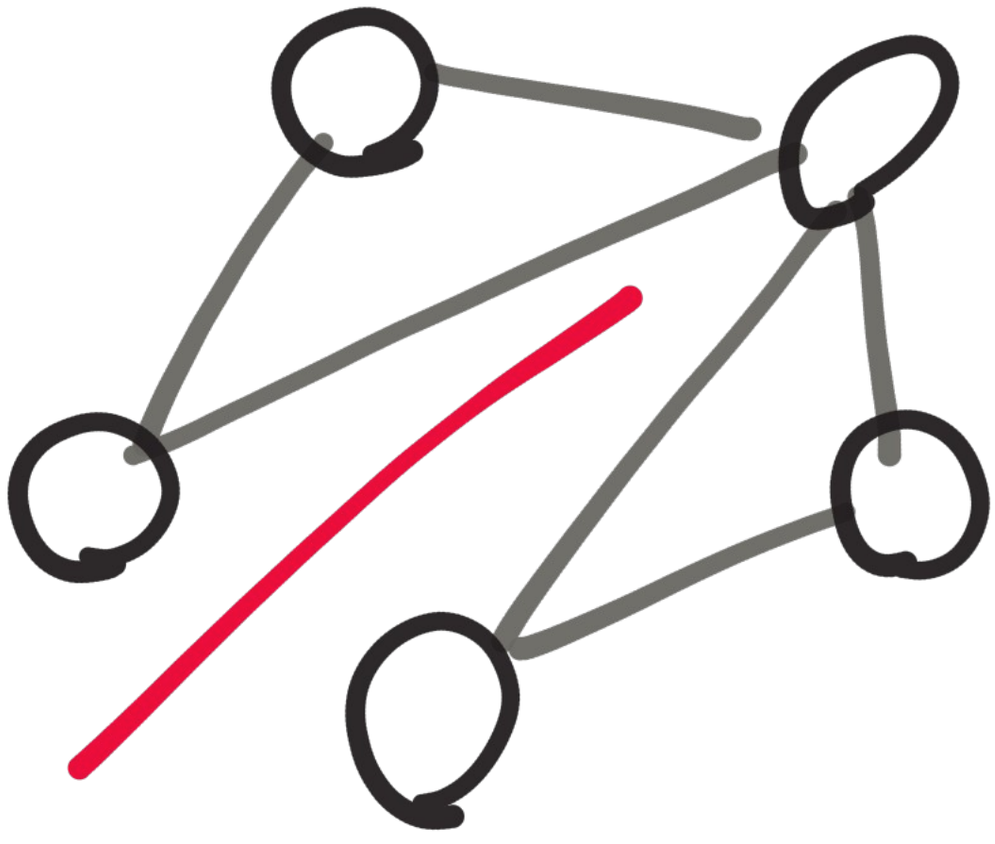
Rory Hart @falican · Feb 20
@warkolm yeah I've read through that, looks like you're all working very hard. I've got an abundance of caution today. Appreciate the time.

Mark Walkom @warkolm · Feb 20
@falican any time!

Mark Walkom @warkolm · Feb 20
@falican re-read that article, most of the complaints in it are pre 1.0 btw, given we're probably not far off 1.5, grains of salt needed ;)

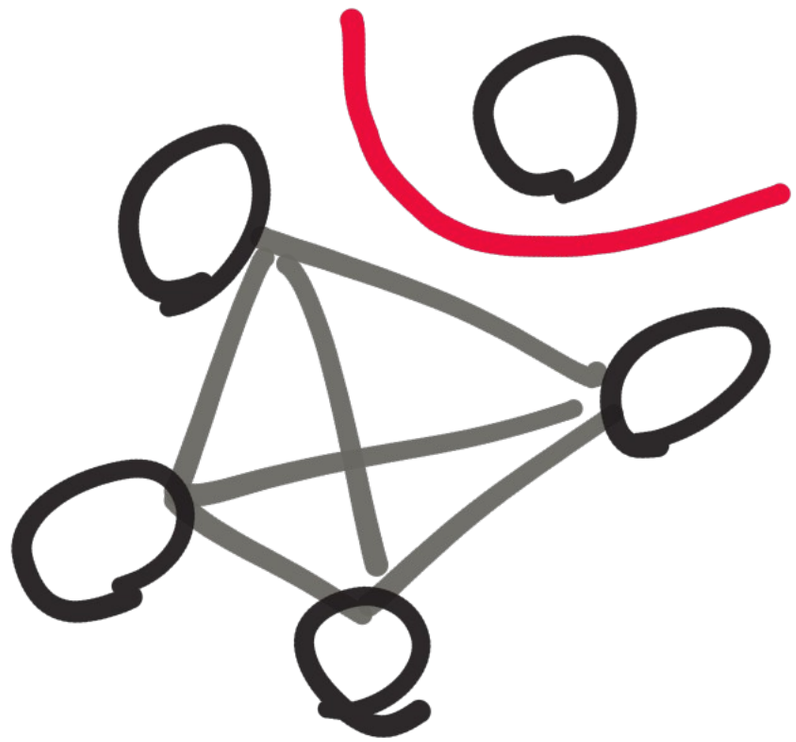
Retesting with ES 1.5.0

Intersecting partitions still cause data loss



Long-lived split-brain appears fixed, but docs are still last

Single node partition still cause data loss!



Electing a new primary still takes

~90 seconds

- Writes will stall

Elasticsearch is doing better, esp. with documentation!

Elastic Search Recommendations

- No longer catastrophic
- But, still loses data in all failure modes
- 


KEROSPIKE

High-performance 5-D KV store for online analytics (esp. in adtech)

RELIABILITY

100% Uptime
with strong consistency (ACID)

"No data loss. Row-level locking... immediate consistency (ACID), with synchronous replication"

We have to go

Deeper

"AS is by and large an AP system that provides high consistency"

"Aerospike provides read-committed isolation level using record locks to ensure isolation between [txns]"

strong

linearizable

sequential

causal

PRAM

RYW MR MW

WFR

serializable

serializable

RR

CS

SI

MAV

P-CI

RC

AP ✓

"for ops. on single keys with replication, Aerospike provides immediate consistency using synchronous writes to replicas"

"...read requests are guaranteed to find the newly written data..."

strong

linearizable

sequential

serializable

serializable

RR

SI

CS

causal

MAV

P-CI

PRAM

WFR

RC

RYW

MR

MW

"Virtually eliminate partition formation proven by years of deployment in DC and cloud environments."

"A key design point of As is to setup cluster nodes that are tightly coupled so that partitions are virtually impossible to create"

Deploy | Aerospike

www.aerospike.com/deploy/

Apps | Slack | Calendar | Music | livegrep | ADP Retirement | Other bookmarks

AEROSPIKE

Search

DISCOVER DOWNLOAD DEVELOP DEPLOY DOCUMENTATION

COMMUNITY

Get Started!

DEPLOY

TRAINING

RESOURCES

Aerospike is now available with Click-to-Deploy on Google Compute Engine

You can quickly spin up an Aerospike cluster for development or production.

Deploy Aerospike clusters easily in Amazon EC2

Learn how to install on EC2 instances, get tips on capacity planning, configuration, benchmarking and using bcache for faster persistent storage.

1M TPS on 1 Server

This can be achieved on a $5k commodity server. Don't believe it? Follow this easy, 4 step

So for a CaS

Register...

6

w2

7

w4

11

w0

4

r2

10

w4

12

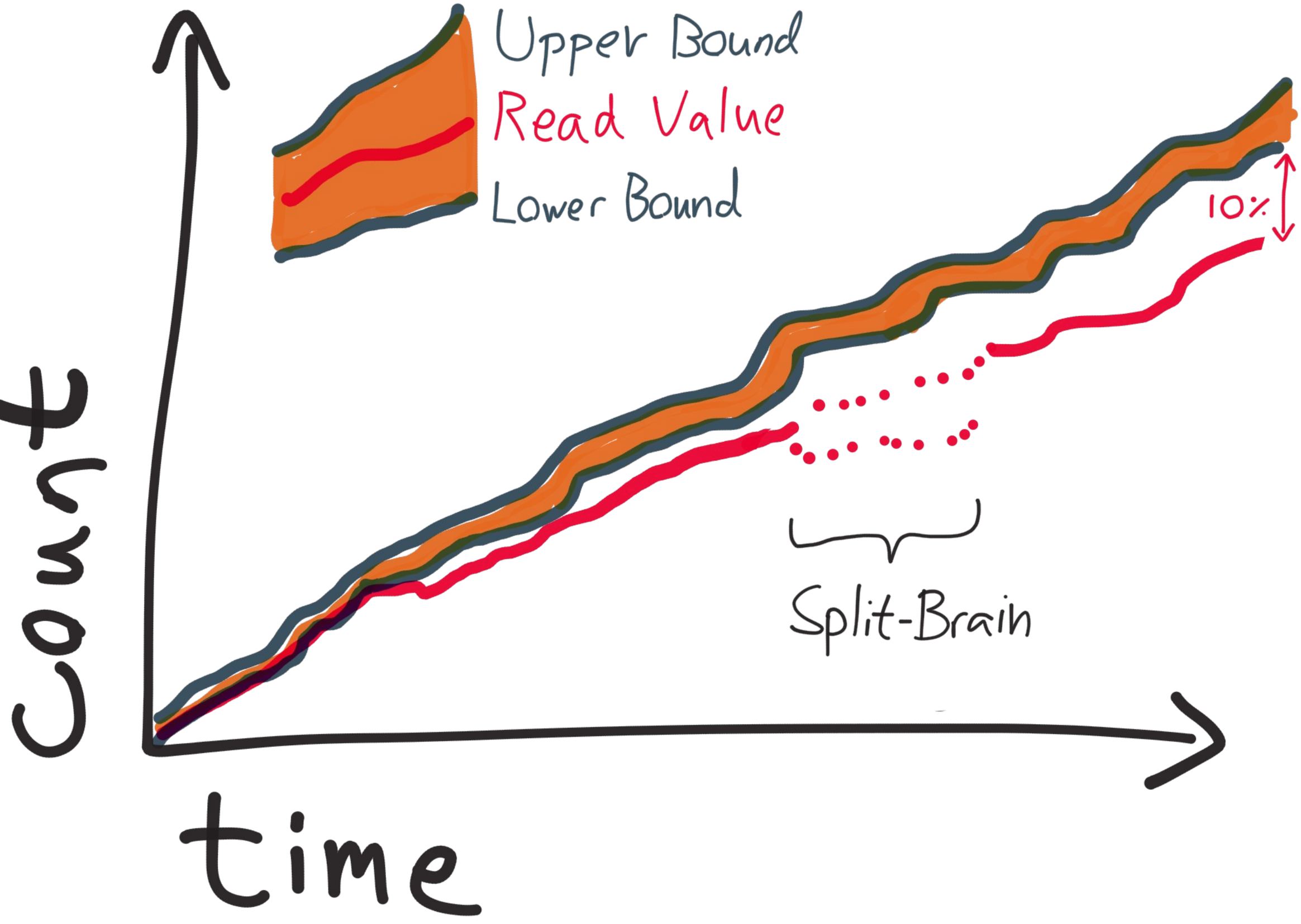
r0

CaS 0→3

What about commutative ops?

$$a + b = b + a$$

Counters will drop increments during a partition

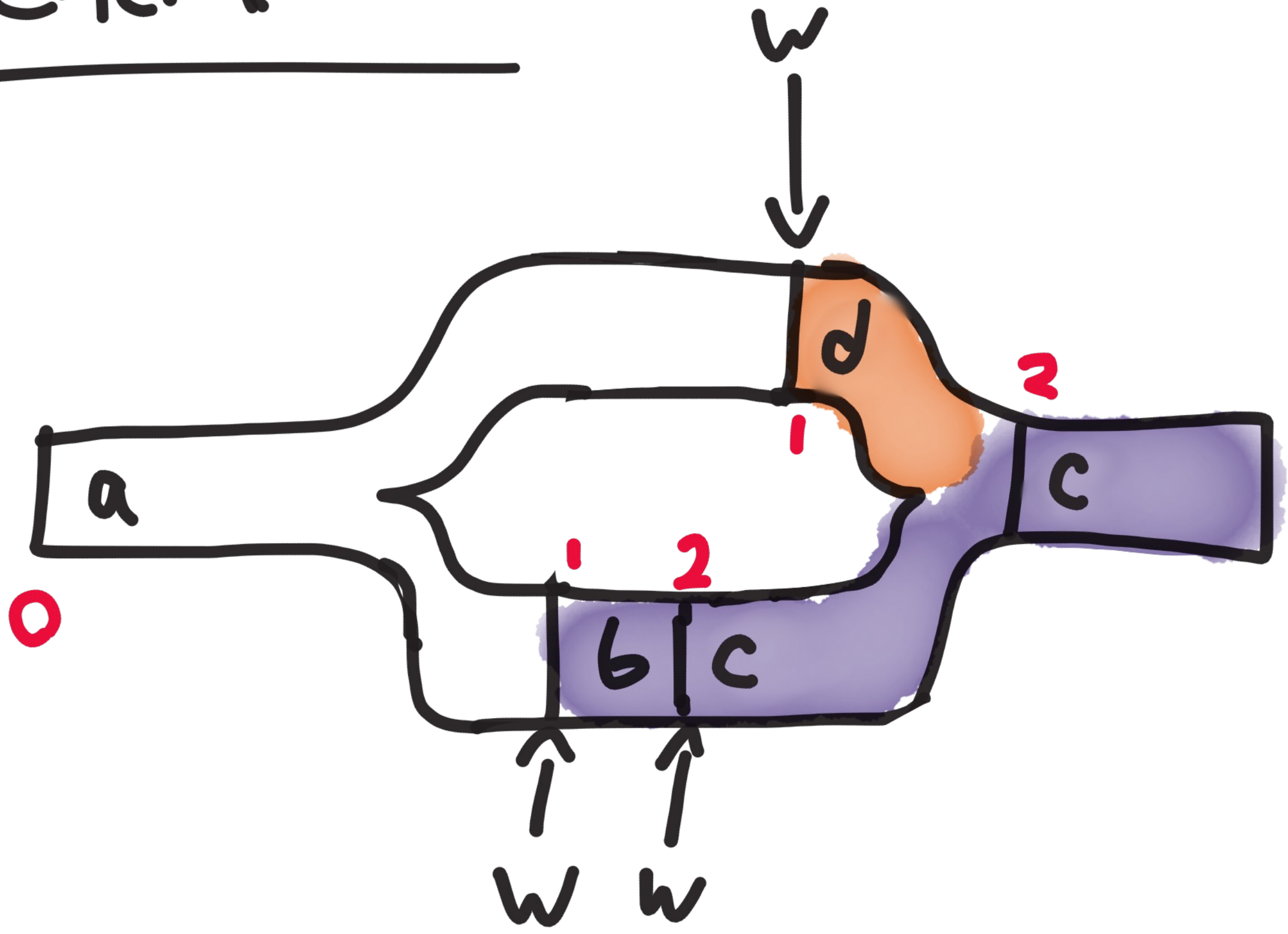

Upper Bound
Read Value
Lower Bound
10%
Split-Brain
count
time

"Deployment modes
AP vs. CP"

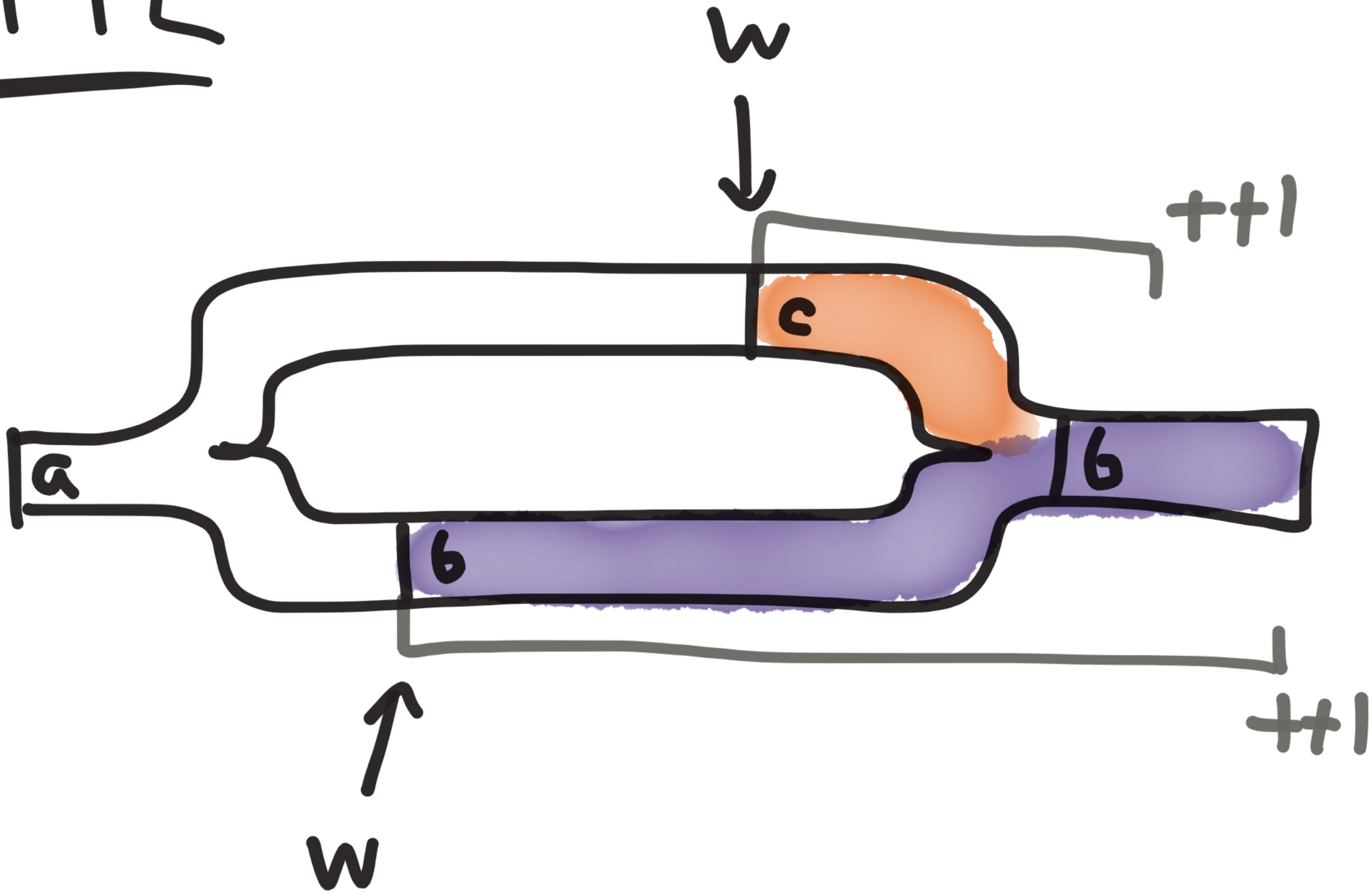
CP is vaporware

OK so how does AP mode work?

Generation



TTL



LOST

UPDATES

Application Merge!

"Read... will return both versions, allowing the app. to resolve"

Like "CP mode"
this feature does
not exist.

No CRDTs for you!

strong

linearizable

sequential

causal

PRAM

RYW

MR

MW

WFR

serializable

serializable

RR

CS

SI

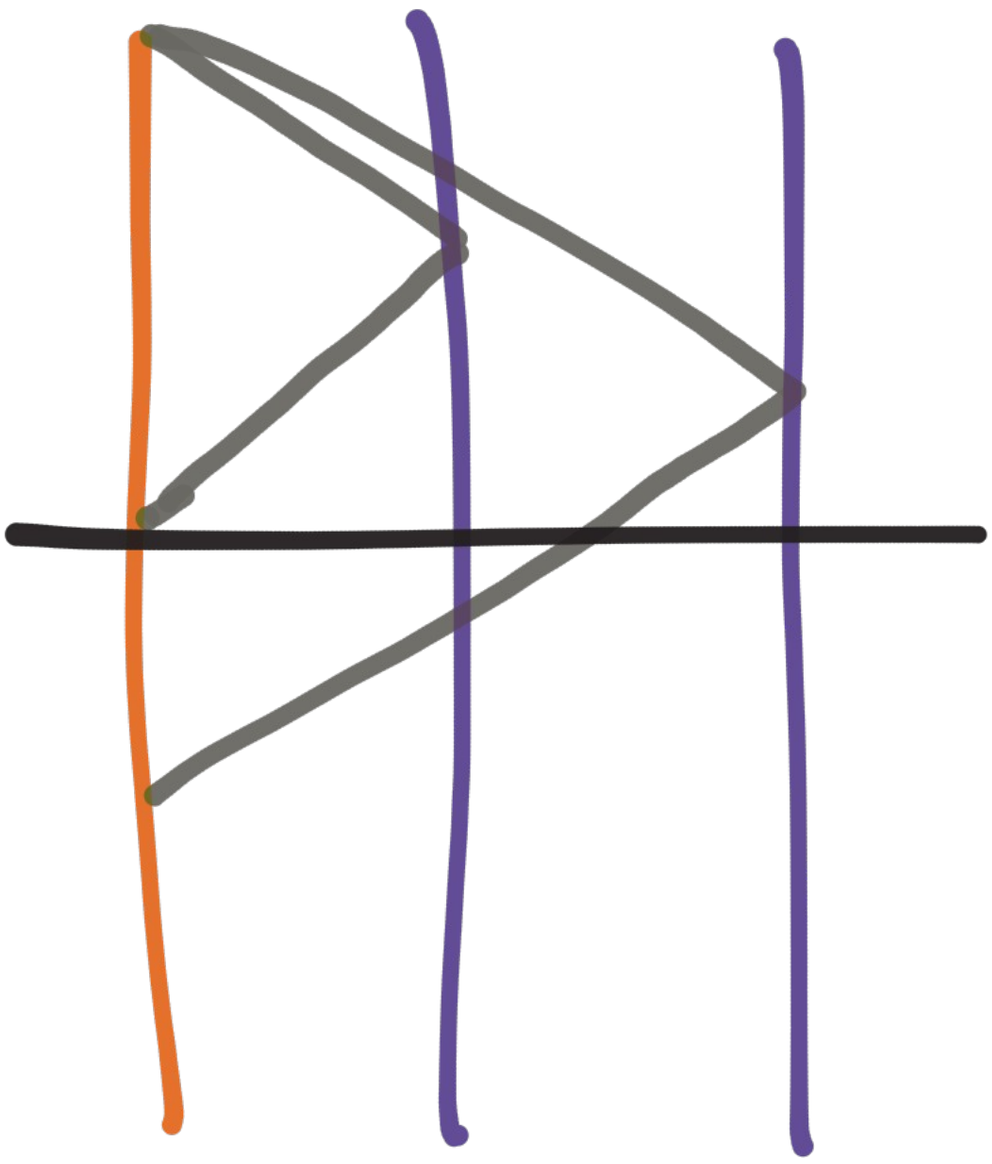
MAV

P-CI

RC

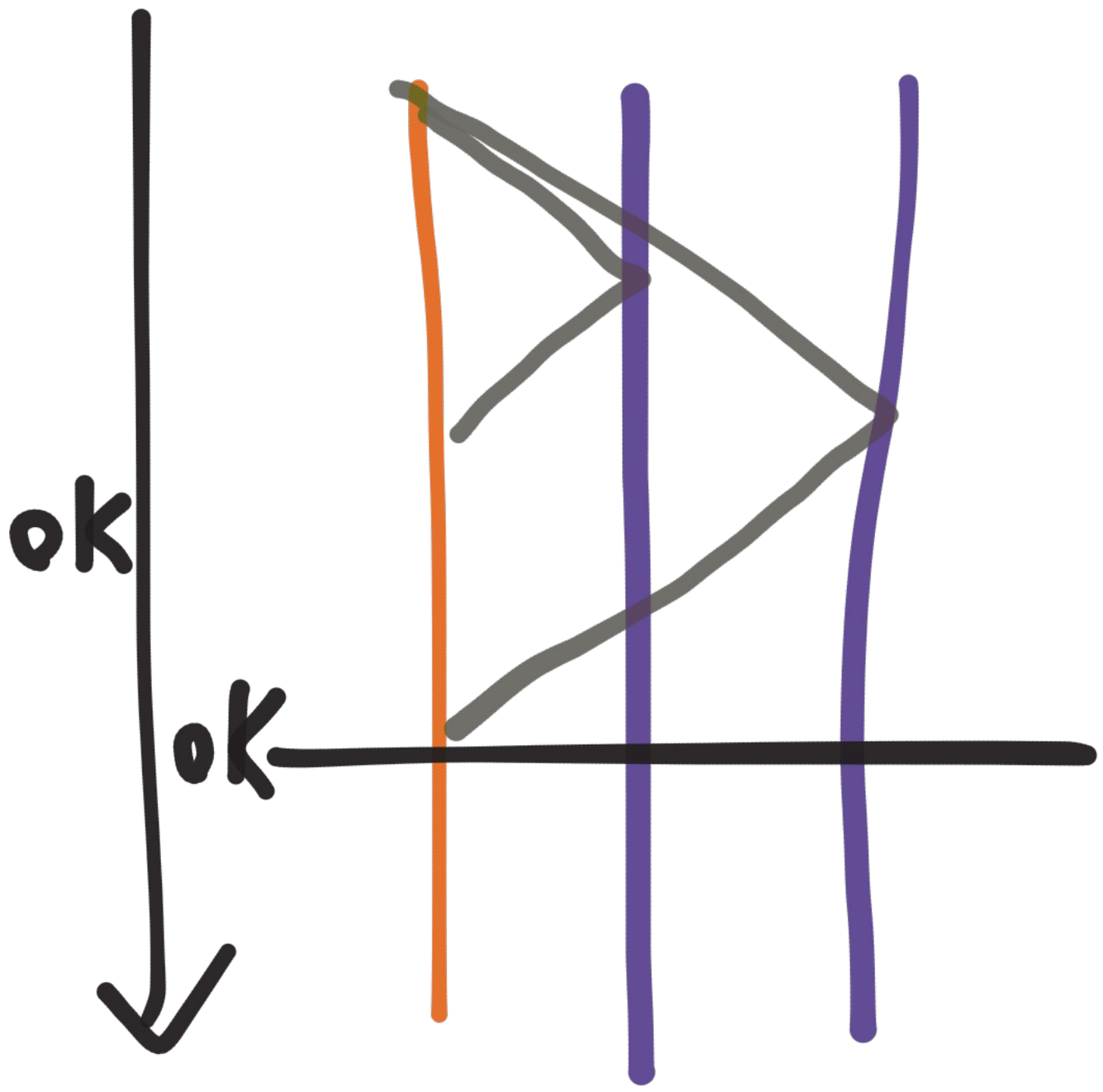
"Why not use your PAXOS implementation?"

F.6.P.



t

Aerospike



Aeraspike only stores

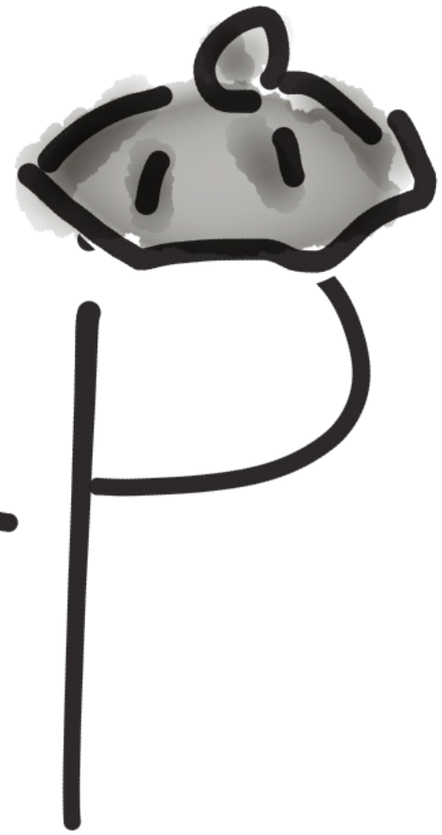
2 copies, not 3.

→ Can't tolerate loss

of a single node

Probably fine for stats data, but expect some loss

Recap



Absence of evidence

— is not —

Evidence of correctness

Mongo ฿ ES

have made

improvements!

Still Surprises

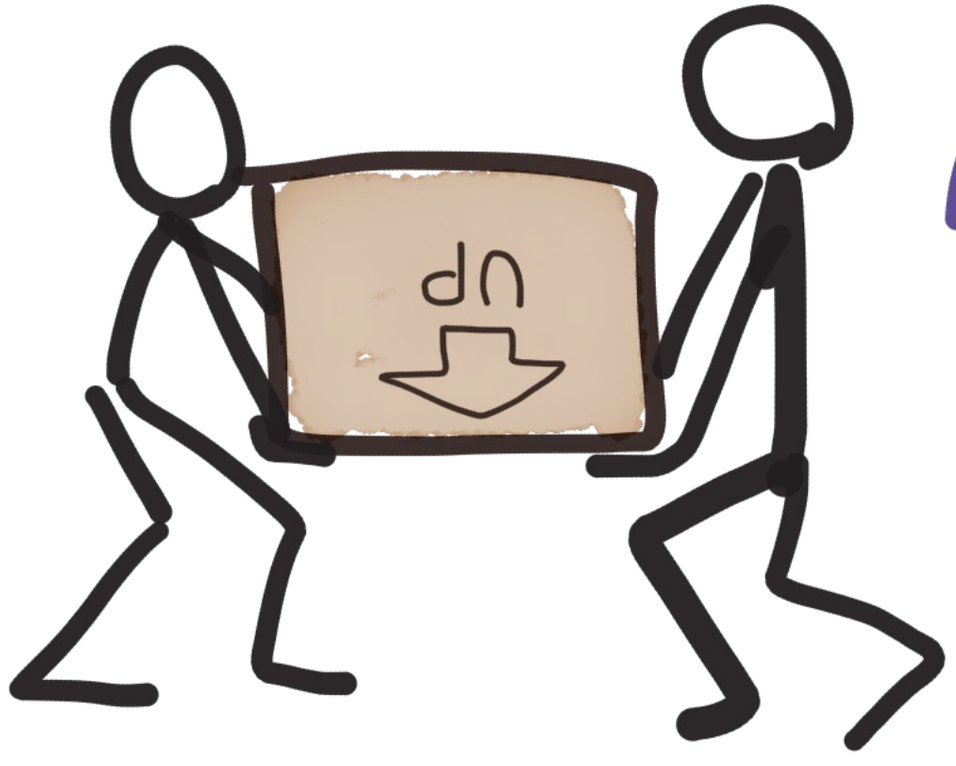
lurking in the

DEEP

Read the docs!

Then test it for YOURSELF

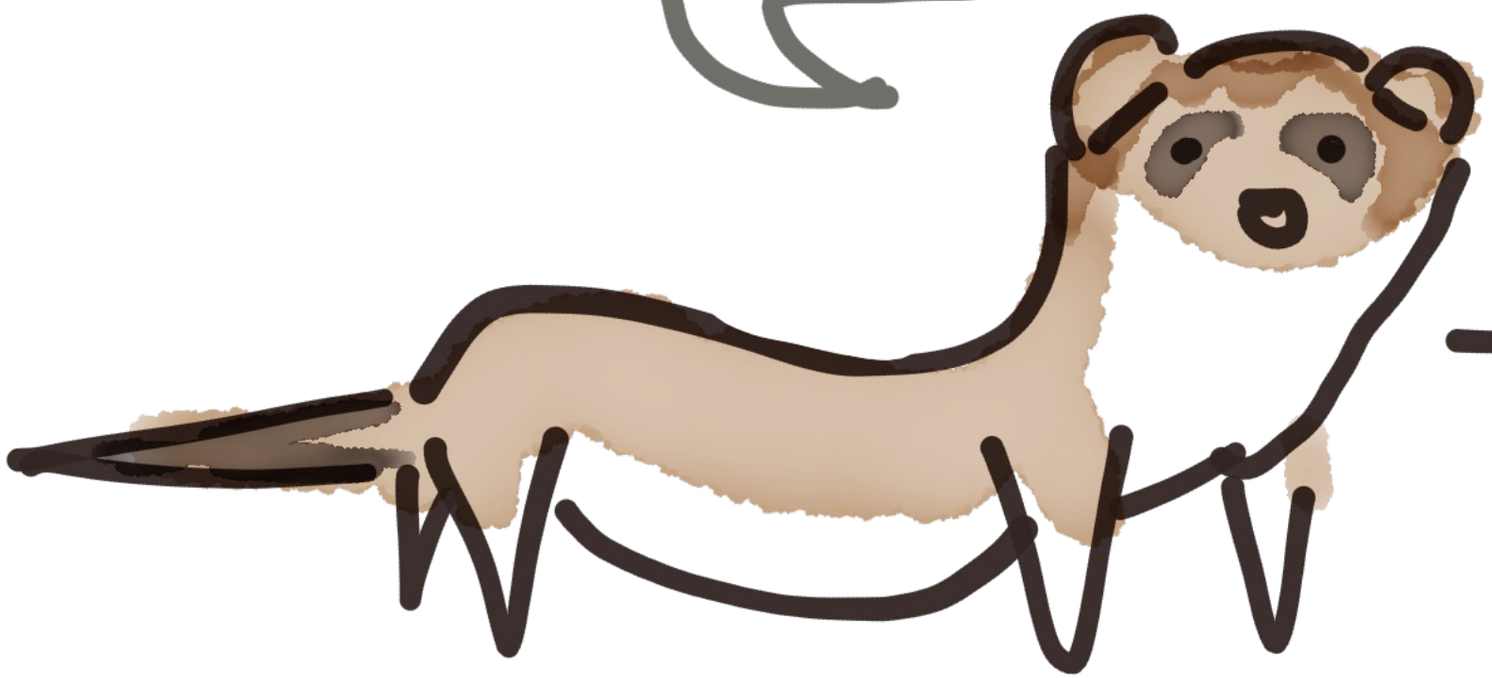
Marketing and Engineering must



"Strict"

"ACID"

"strong"



-WORDS

Be Formal

&

Be Specific!

Figure out the invariants your system needs

How much

can you tolerate?

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
```

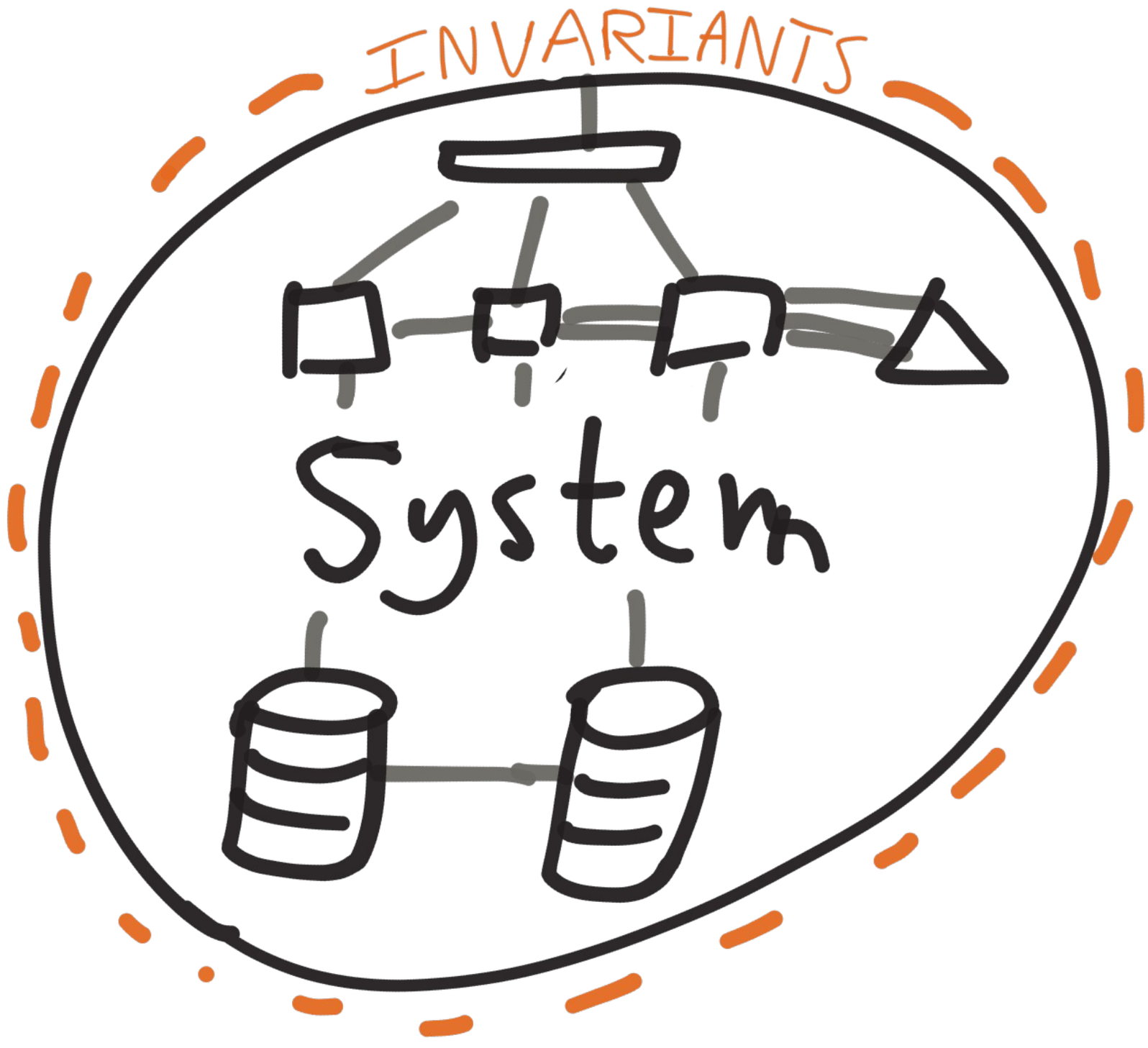
```
# tc qdisc ... delay ...
                drop ...
```

Test your

Systems

end ——— to ——→ end

vs.



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

Thanks!

Kevin Porter
Lucien Valmar
Stripe
Evan Broder
Boaz Leskes
Asya Kamsky
Lee Hinman
Kelly Stirman
Matt Kangas
Marc Hedlund

CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2015

goto; conference

QUESTIONS?

Please remember to evaluate via the GotoGuide app

follow us @gotochgo Conference: May 11-12 / Workshops: 13-14