

CONVERGENCE

versus

CONSENSUS

CRDTs

and the quest for

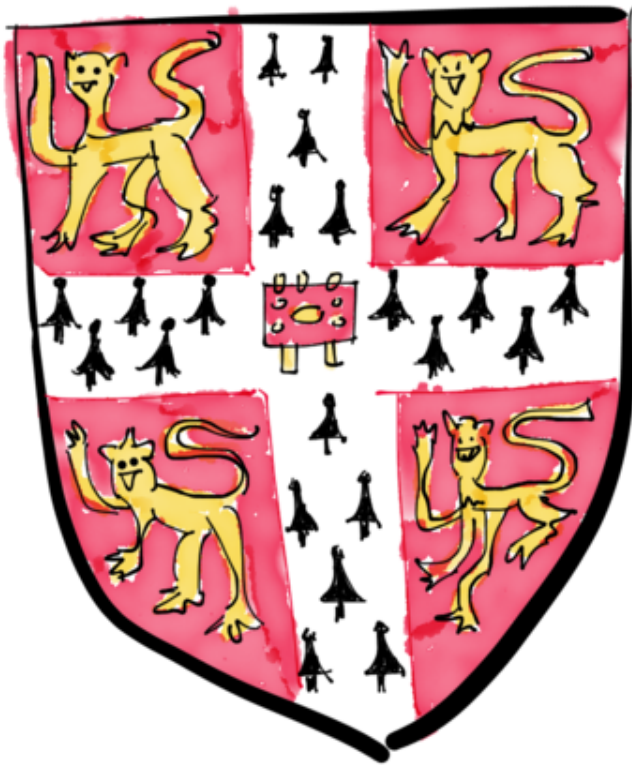
DISTRIBUTED CONSISTENCY

Martin Kleppmann • University of Cambridge • @martinkl

dataintensive.net



@martinkl

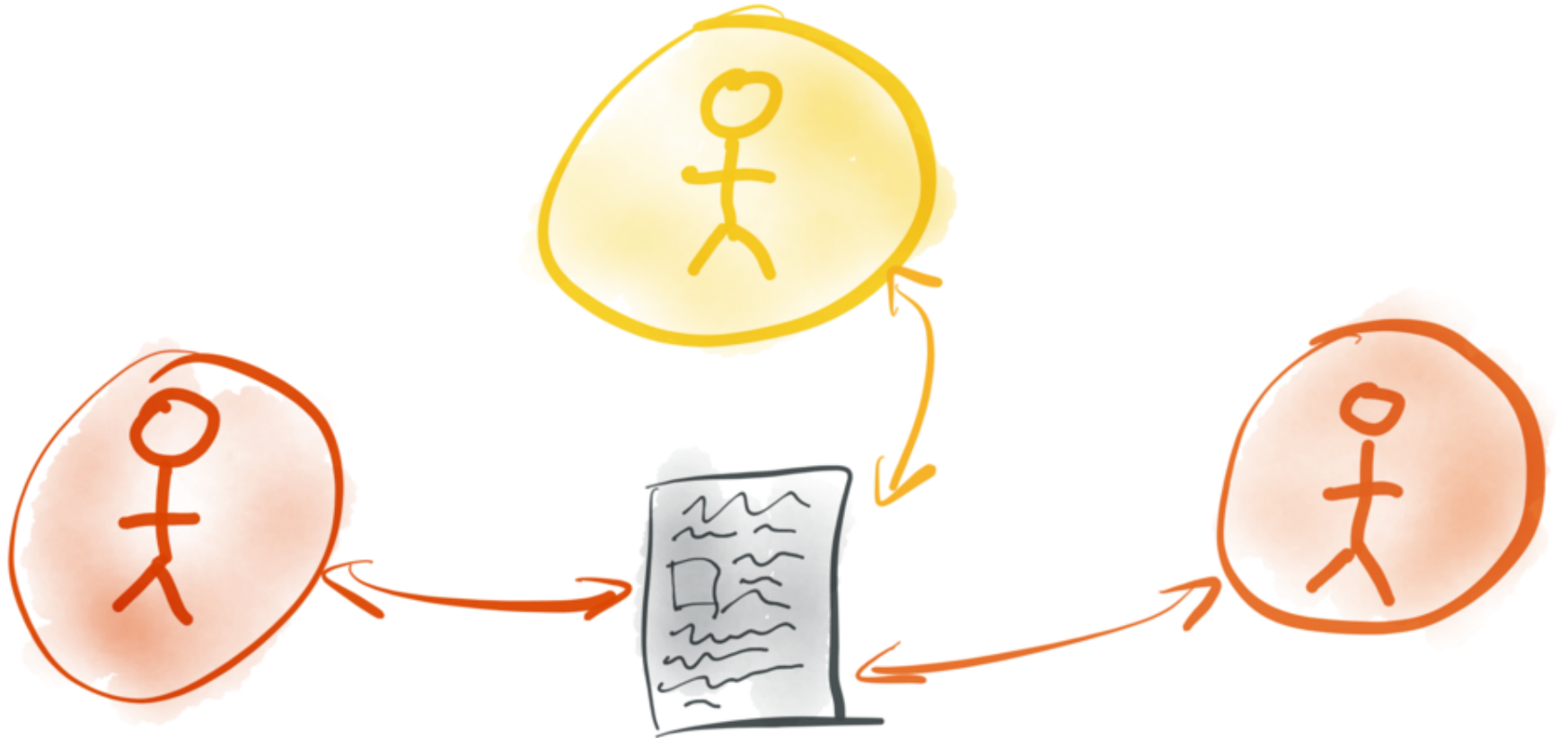


UNIVERSITY OF
CAMBRIDGE



~~UNIVERSITY OF CAMBRIDGE
ON STRIKE TODAY~~

COLLABORATIVE APPLICATIONS





hack on code



time

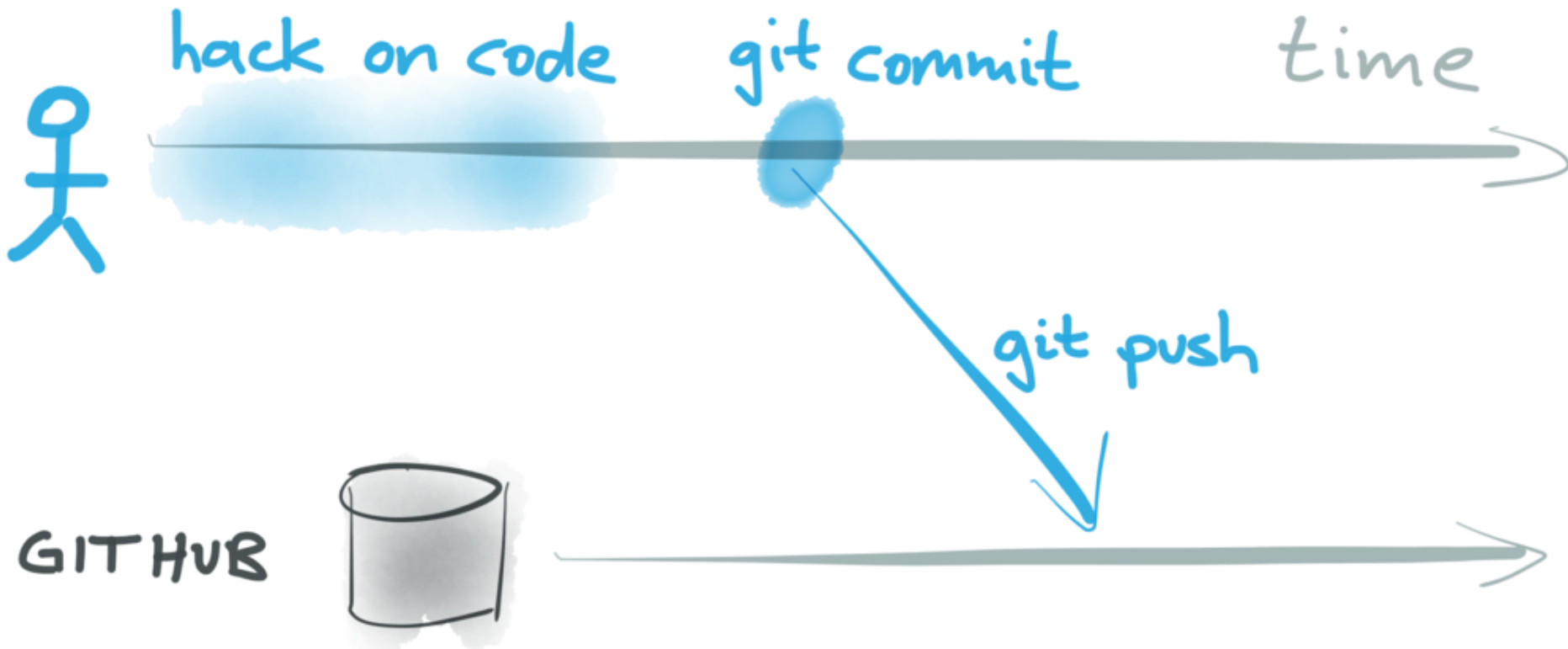


hack on code

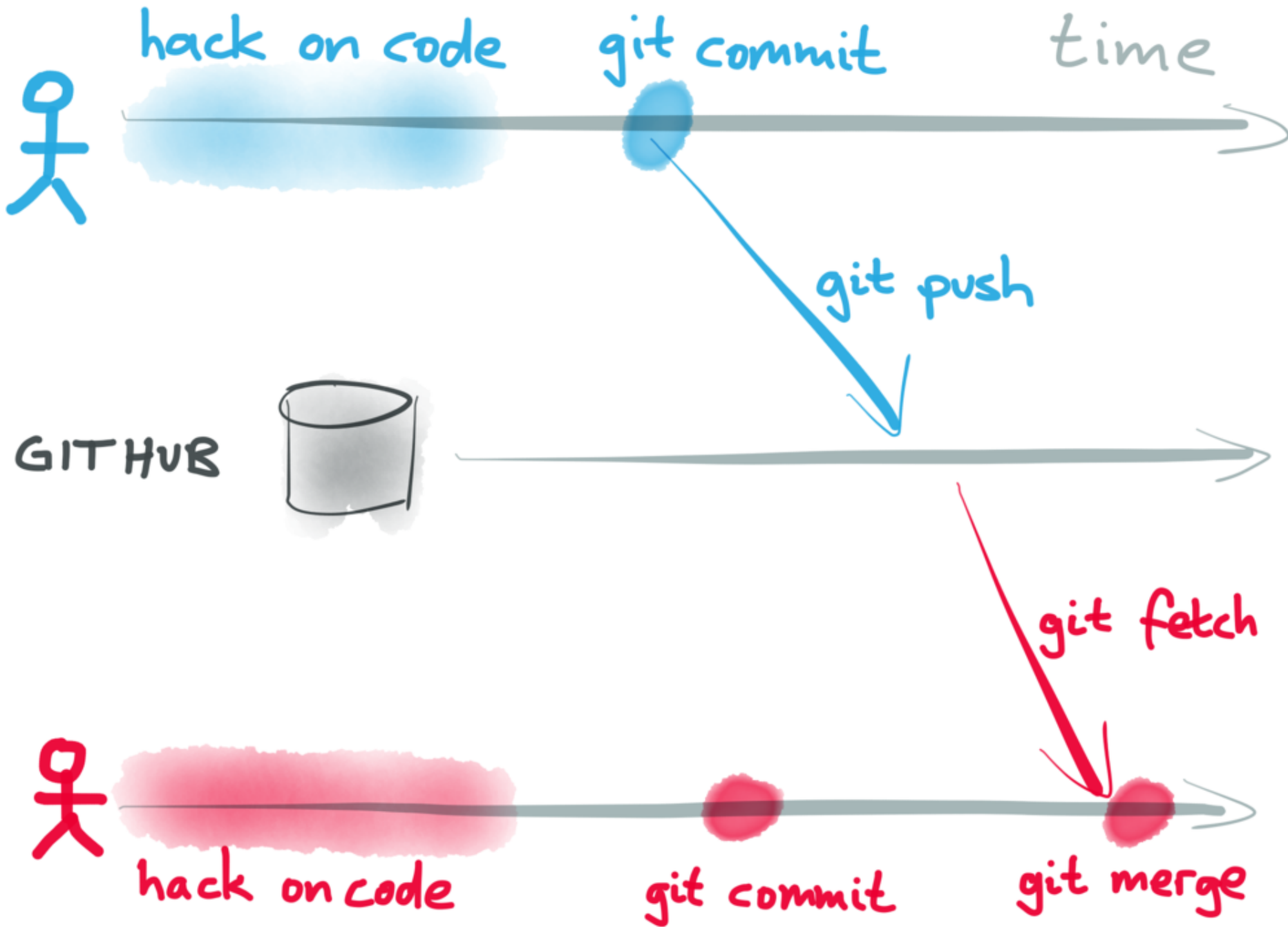
git commit

time







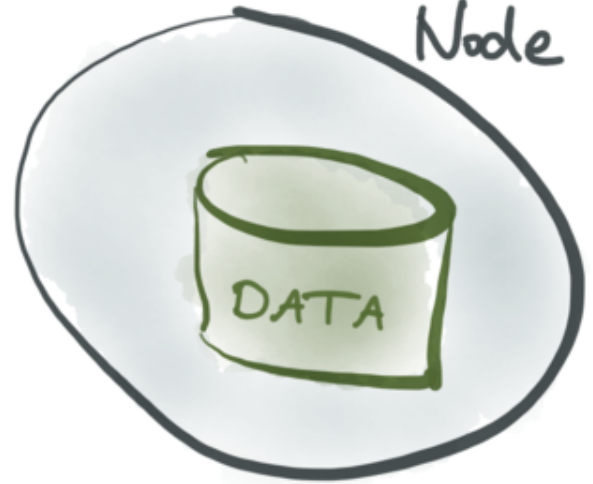


REPLICATION.

Node i



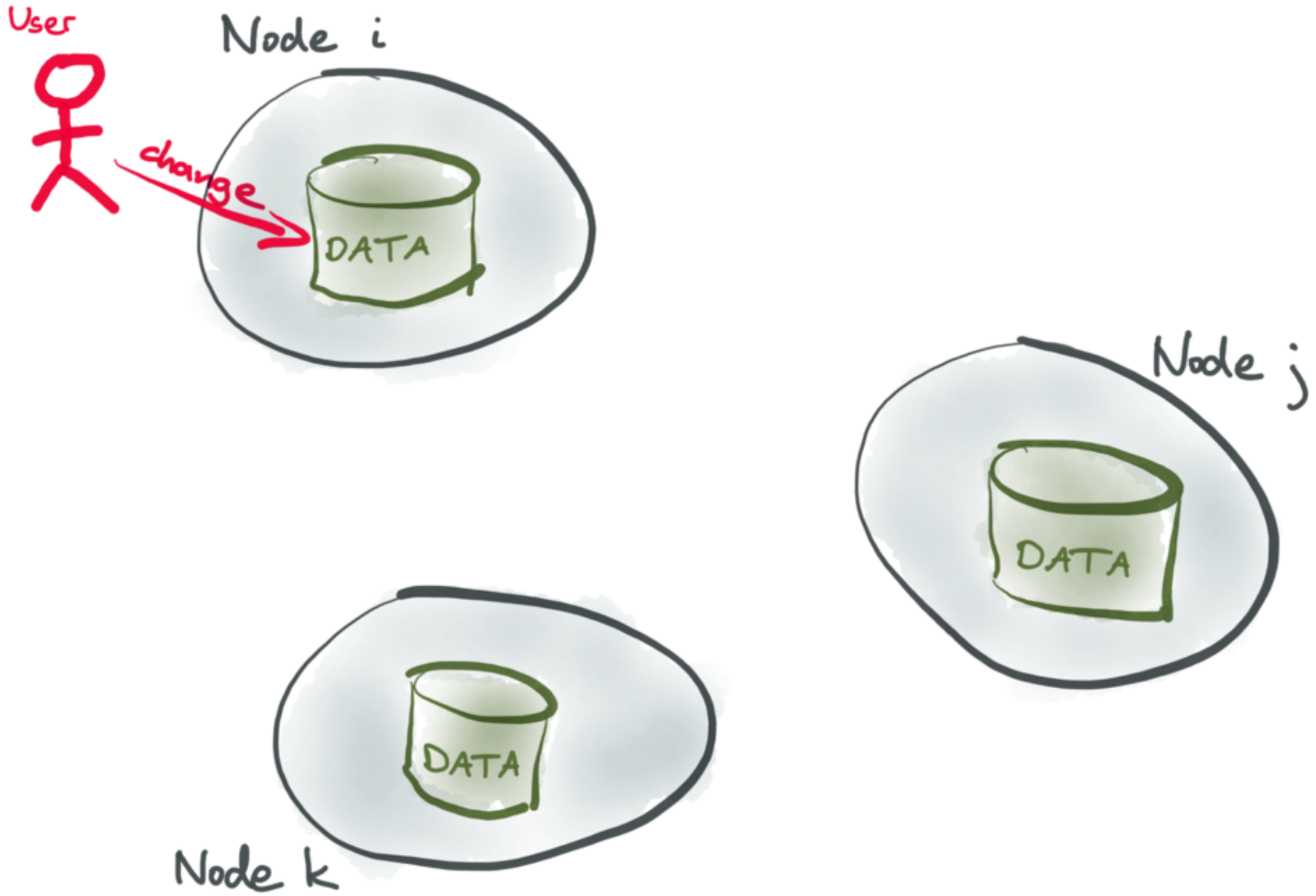
Node j



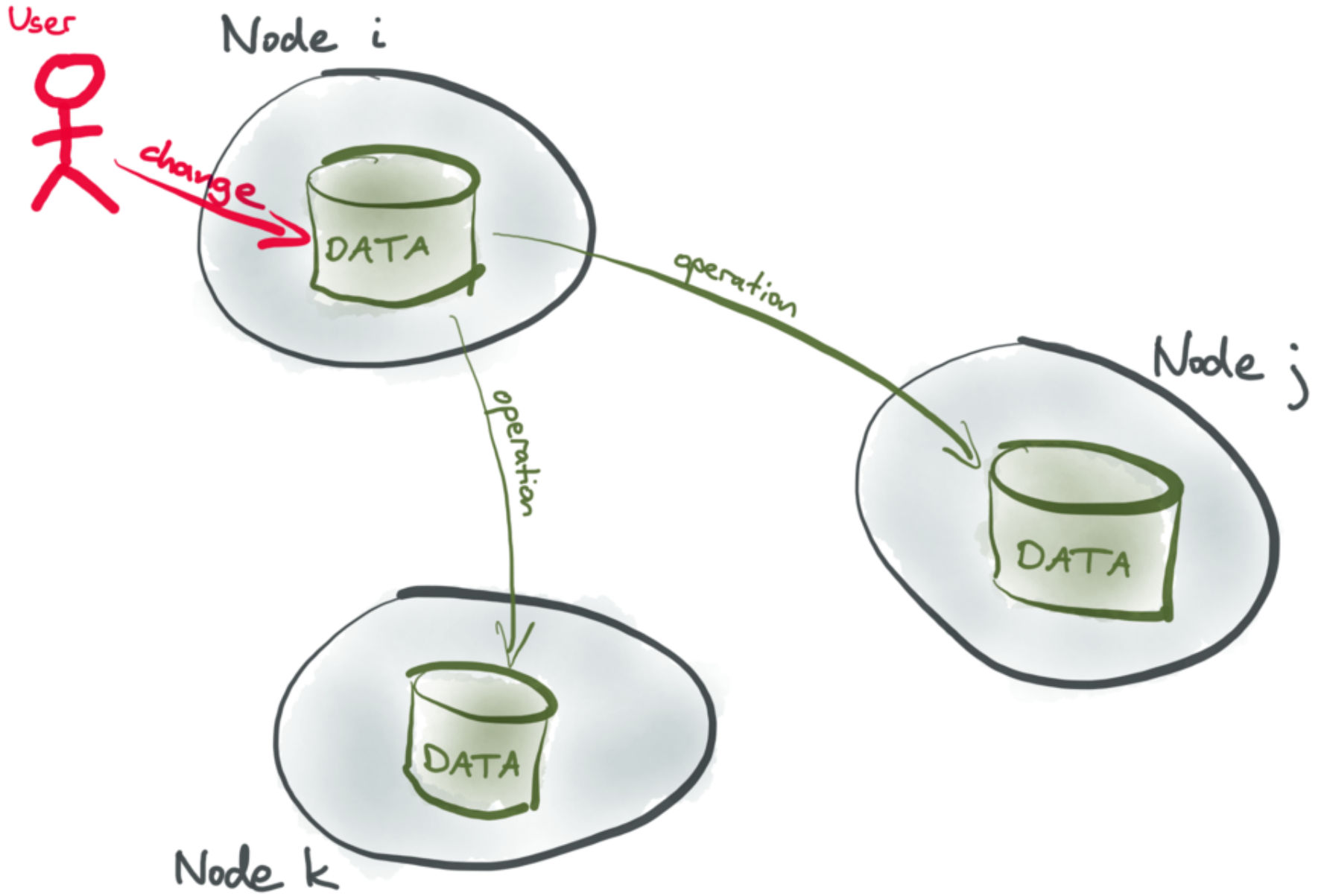
Node k



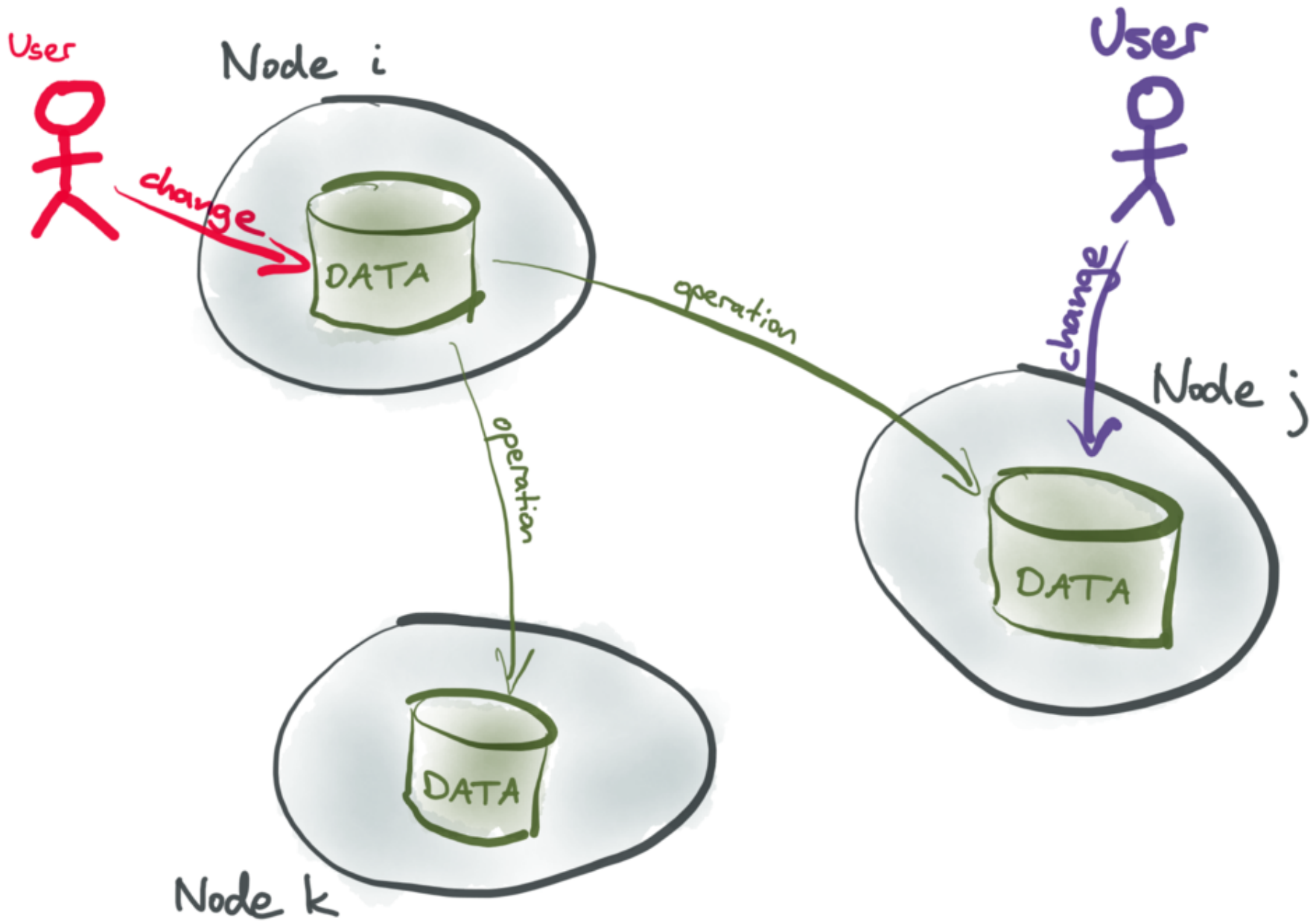
REPLICATION.



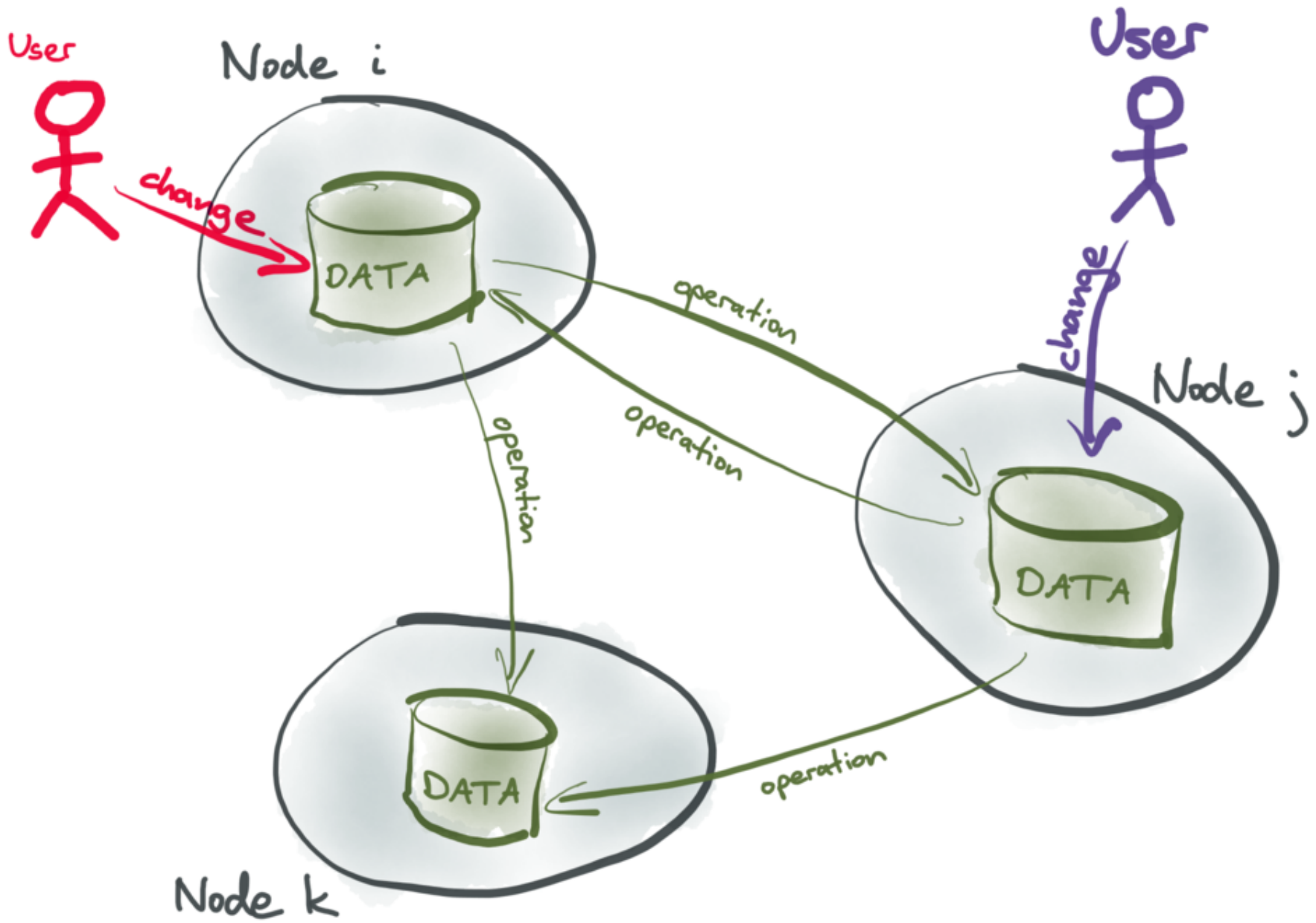
REPLICATION.



REPLICATION.

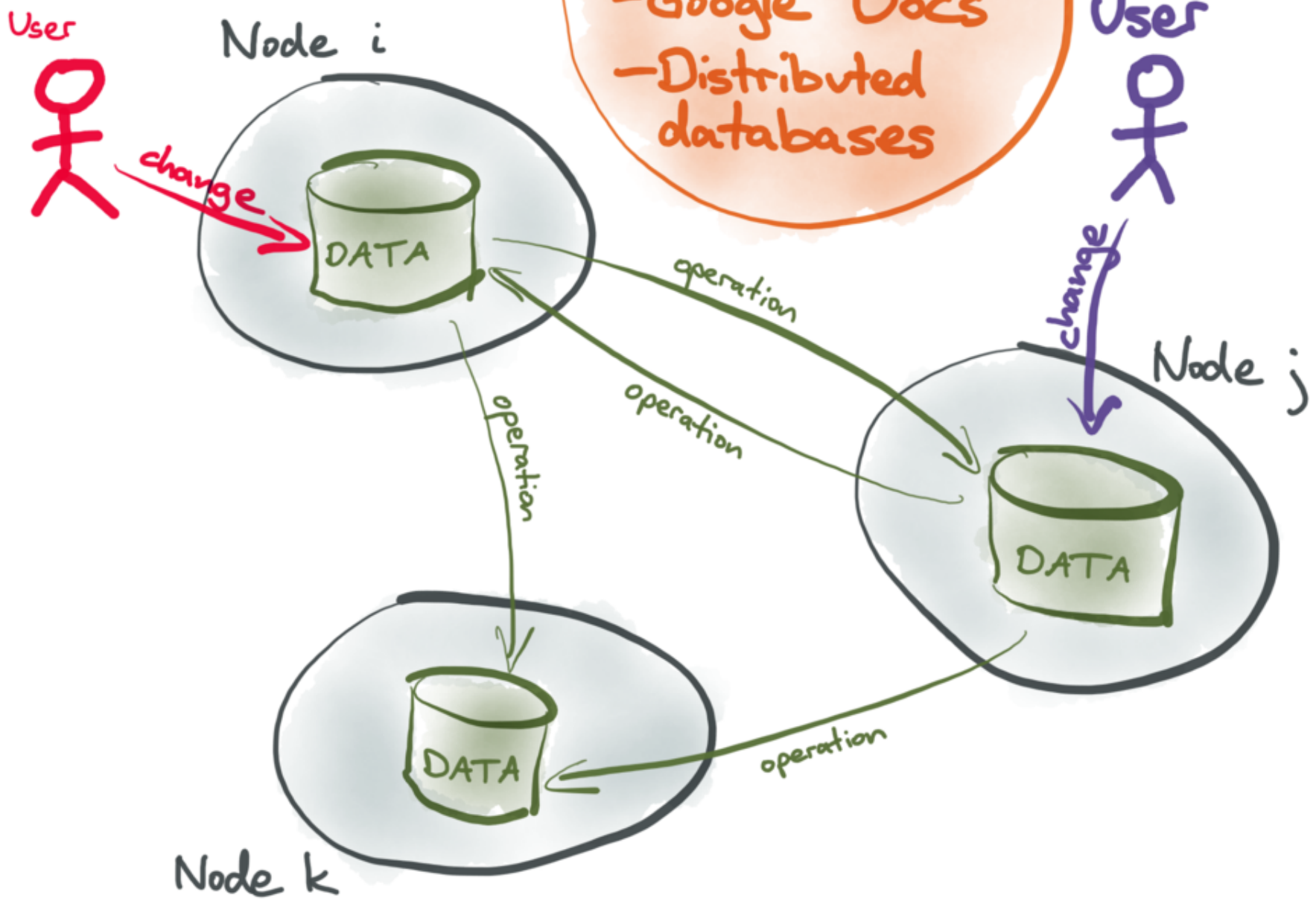


REPLICATION.

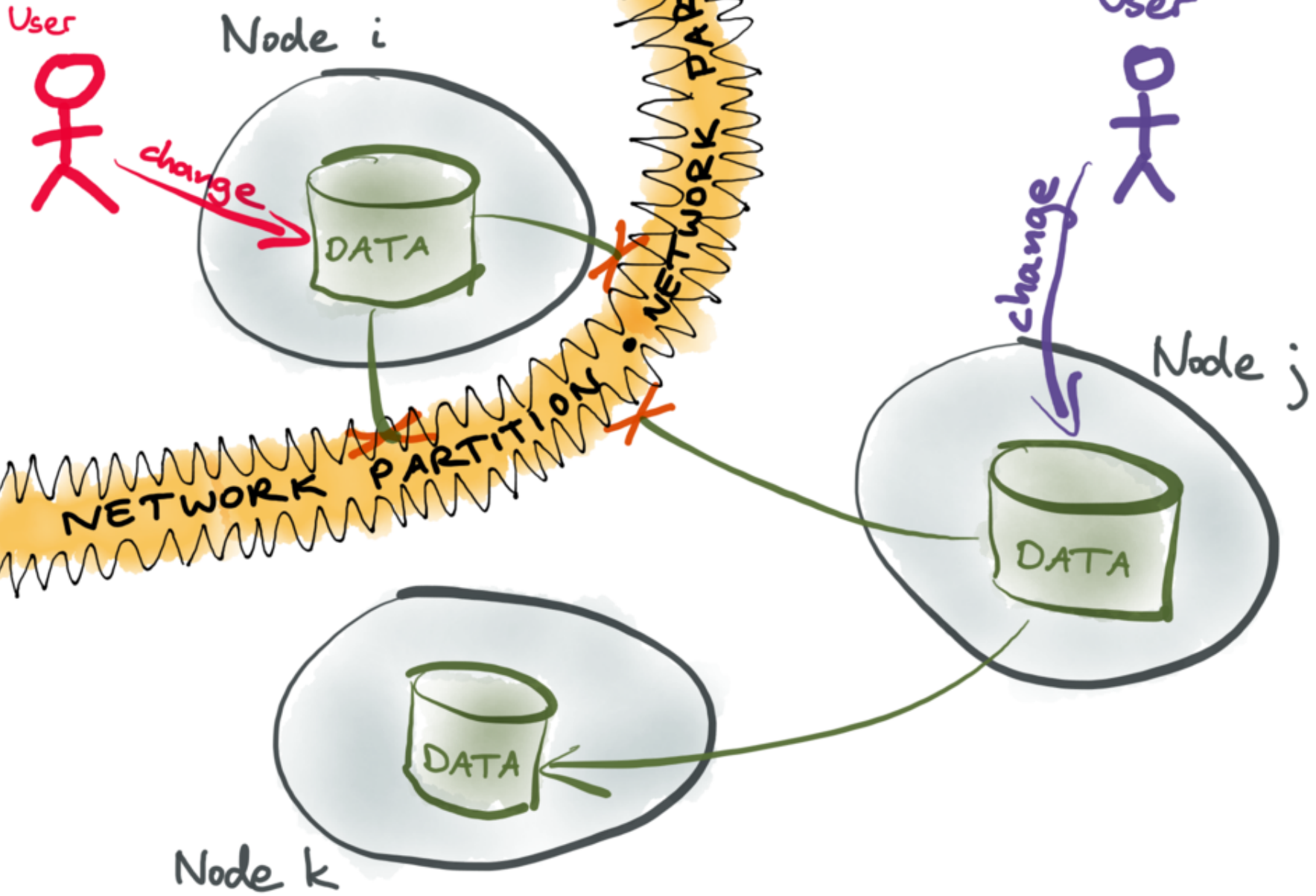


REPLICATION.

Examples:
- Google Docs
- Distributed databases



REPLICATION.



Example: Text editing



Example: Text editing

insert "World"
after "Hello"



"Hello!"

"Hello World!"

time

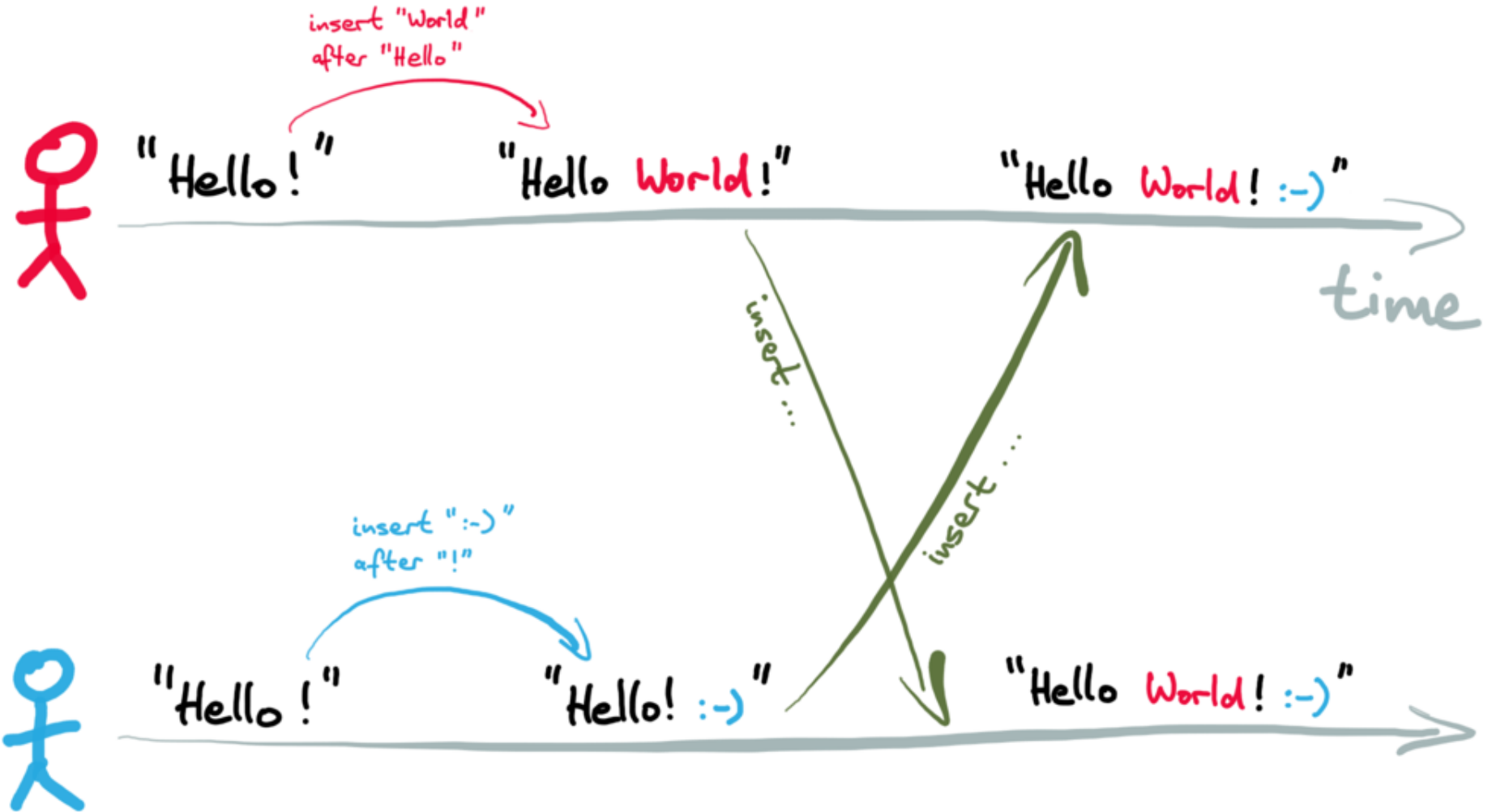
insert ":-)"
after "!"



"Hello!"

"Hello! :-)"

Example: Text editing



Example: set manipulation



$$s = \{a, b\}$$

time



$$s = \{a, b\}$$

Example: set manipulation



$$s = \{a, b\}$$

$$s' = s - \{b\}$$

$$s = \{a\}$$

time



$$s = \{a, b\}$$

$$s' = s \cup \{c\}$$

$$s = \{a, b, c\}$$

Example: set manipulation



$$s = \{a, b\}$$

$$s' = s - \{b\}$$

$$s = \{a\}$$

$$s = \{a, c\}$$

time



$$s = \{a, b\}$$

$$s' = s \cup \{c\}$$

$$s = \{a, b, c\}$$

$$s = \{a, c\}$$

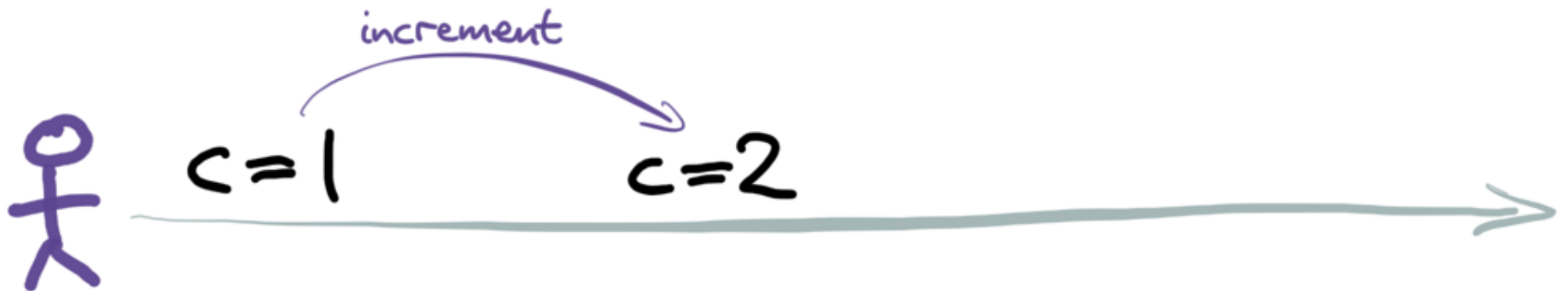
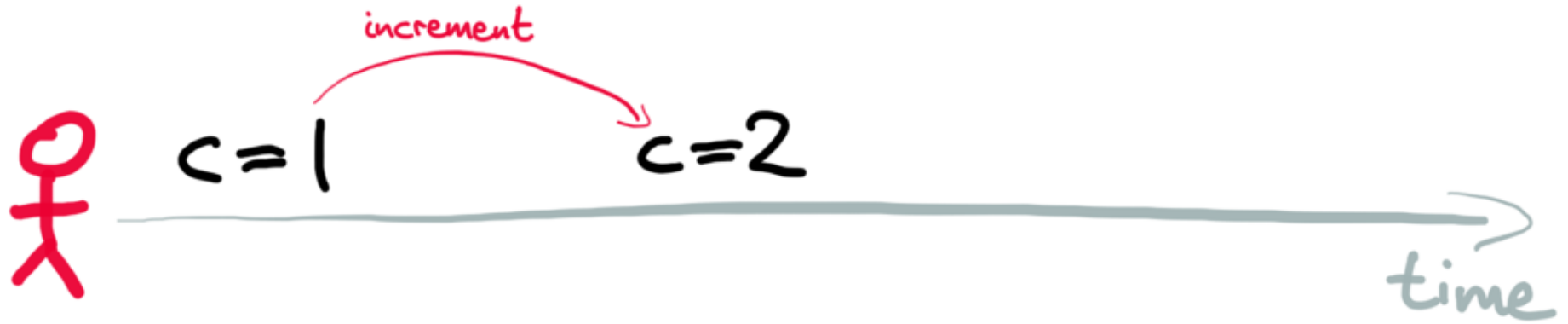
remove b

add c

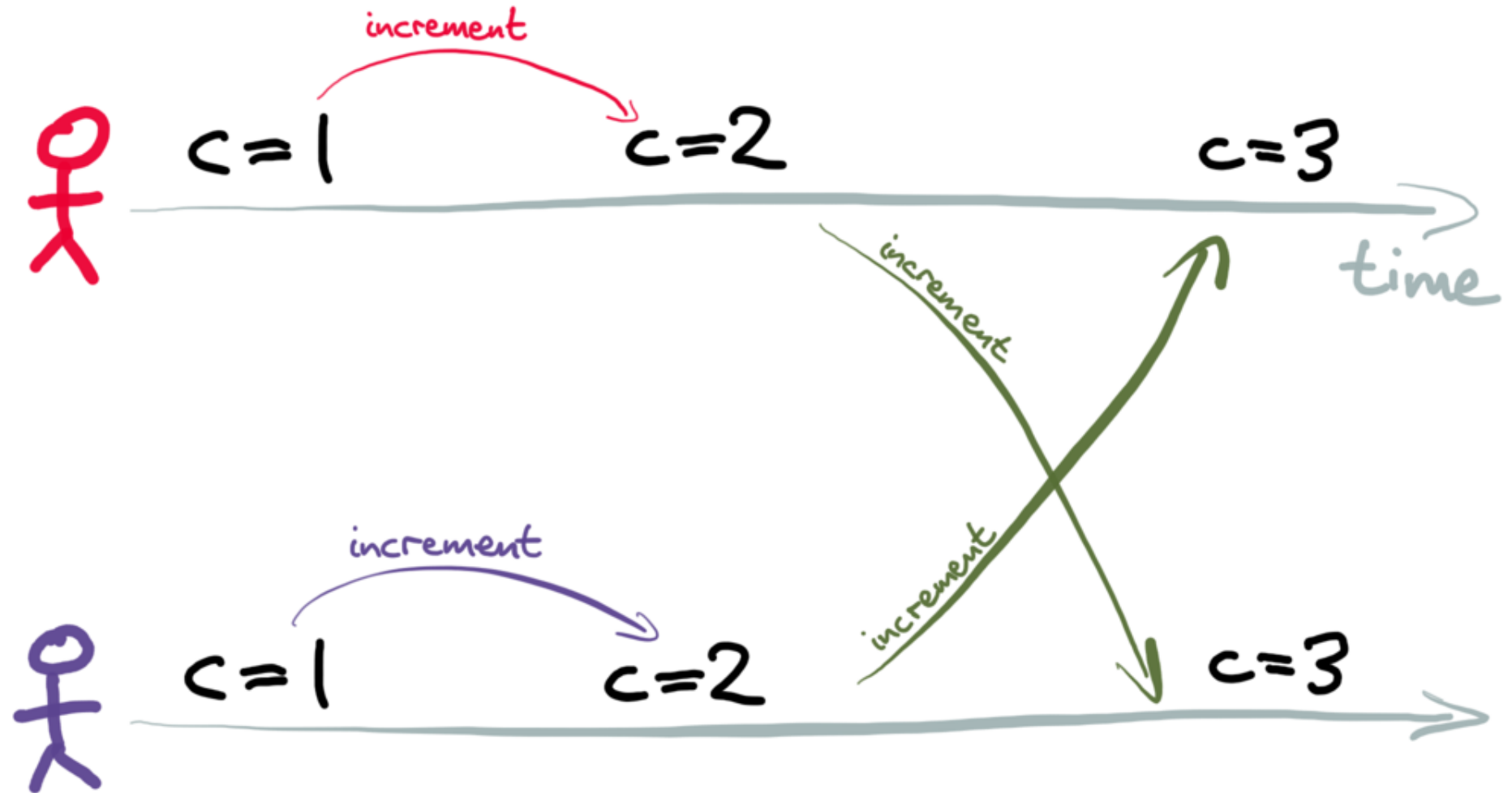
Example: counter



Example : counter



Example: counter



Algorithms for convergence

OPERATIONAL TRANSFORMATION (OT)

- e.g. Google Docs, MS Office Online

1989-2006

CONFLICT-FREE REPLICATED DATA TYPES (CRDTs)

- e.g. Riak, TomTom GPS, Teletype for Atom, ...

2006 - present

FAILURES OF OPERATIONAL TRANSFORM

dOFT

Ellis & Gibbs
1989

adOPTed

Ressel et al.
1996

IMOR

Imine et al.
2003

Jupiter

Nichols et al.
1995

SOCT2

Suleiman et al.
1997

SDT

Li & Li
2004

SOCT 3/4

Vidot et al.
2000

TTF

Oster et al.
2006

FAILURES OF OPERATIONAL TRANSFORM

~~DOPT~~

~~Ellis & Gibbs~~

~~1989~~

wrong

adOPTed

Ressel et al.

1996

IMOR

Imine et al.

2003

Jupiter

Nichols et al.

1995

SOCT2

Suleiman et al.

1997

SDT

Li & Li

2004

SOCT 3/4

Vidot et al.

2000

TTF

Oster et al.

2006

FAILURES OF OPERATIONAL TRANSFORM

~~DOPT~~

~~Ellis & Gibbs~~

~~1989~~

wrong

~~adOPTed~~

~~Ressel et al.~~

~~1996~~

wrong

IMOR

Imine et al.

2003

Jupiter

Nichols et al.

1995

SOCT2

Suleiman et al.

1997

SDT

Li & Li

2004

SOCT 3/4

Vidot et al.

2000

TTF

Oster et al.

2006

FAILURES OF OPERATIONAL TRANSFORM

~~DOPT~~

~~Ellis & Gibbs~~

~~1989~~

wrong

~~adOPTed~~

~~Ressel et al.~~

~~1996~~

wrong

IMOR

Imine et al.

2003

Jupiter

Nichols et al.

1995

~~SOCT2~~

~~Sulaiman et al.~~

~~1997~~

wrong

SDT

Li & Li

2004

SOCT 3/4

Vidot et al.

2000

TTF

Oster et al.

2006

FAILURES OF OPERATIONAL TRANSFORM

~~DOPT~~

~~Ellis & Gibbs~~

~~1989~~

wrong

~~adOPTed~~

~~Ressel et al.~~

~~1996~~

wrong

~~IMOR~~

~~Imine et al.~~

~~2003~~

wrong

Jupiter

Nichols et al.

1995

~~SOCT2~~

~~Sulaiman et al.~~

~~1997~~

wrong

SDT

Li & Li

2004

SOCT 3/4

Vidot et al.

2000

TTF

Oster et al.

2006

FAILURES OF OPERATIONAL TRANSFORM

~~DOPT~~

~~Ellis & Gibbs~~

~~1989~~

~~wrong~~

~~adOPTed~~

~~Ressel et al.~~

~~1996~~

~~wrong~~

~~IMOR~~

~~Imine et al.~~

~~2003~~

~~wrong~~

Jupiter

Nichols et al.

1995

~~SOCT2~~

~~Sulaiman et al.~~

~~1997~~

~~wrong~~

~~SDT~~

~~Li & Li~~

~~2004~~

~~wrong~~

SOCT 3/4

Vidot et al.

2000

TTF

Oster et al.

2006

FAILURES OF OPERATIONAL TRANSFORM

~~DOPT~~

~~Ellis & Gibbs~~

~~1989~~

wrong

~~adOPTed~~

~~Ressel et al.~~

~~1996~~

wrong

~~IMOR~~

~~Imine et al.~~

~~2003~~

wrong

Jupiter

Nichols et al.

1995

require
central
server

~~SOCT2~~

~~Sulaiman et al.~~

~~1997~~

wrong

~~SDT~~

~~Li & Li~~

~~2004~~

wrong

SOCT 3/4

Vidot et al.

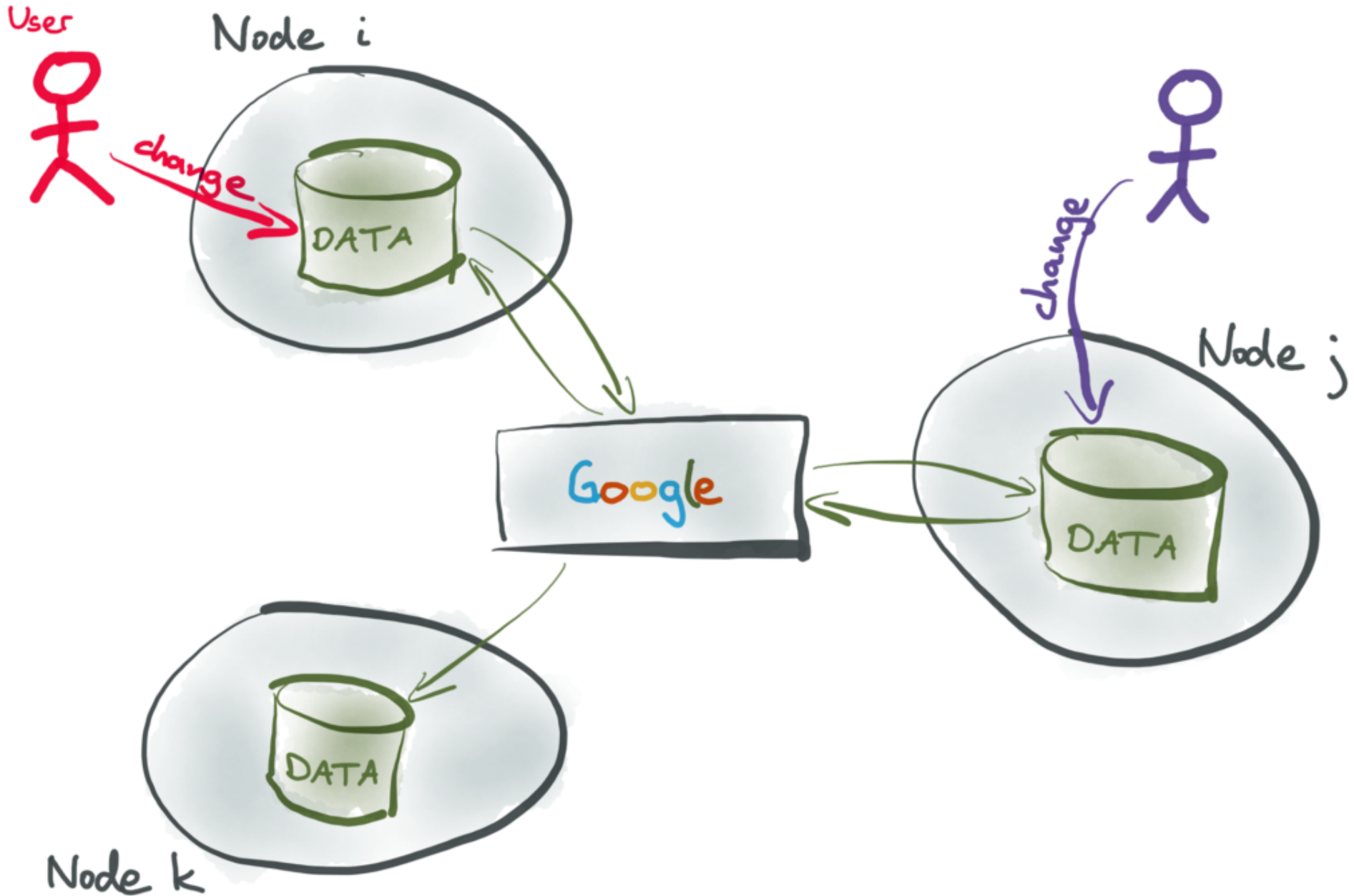
2000

TTF

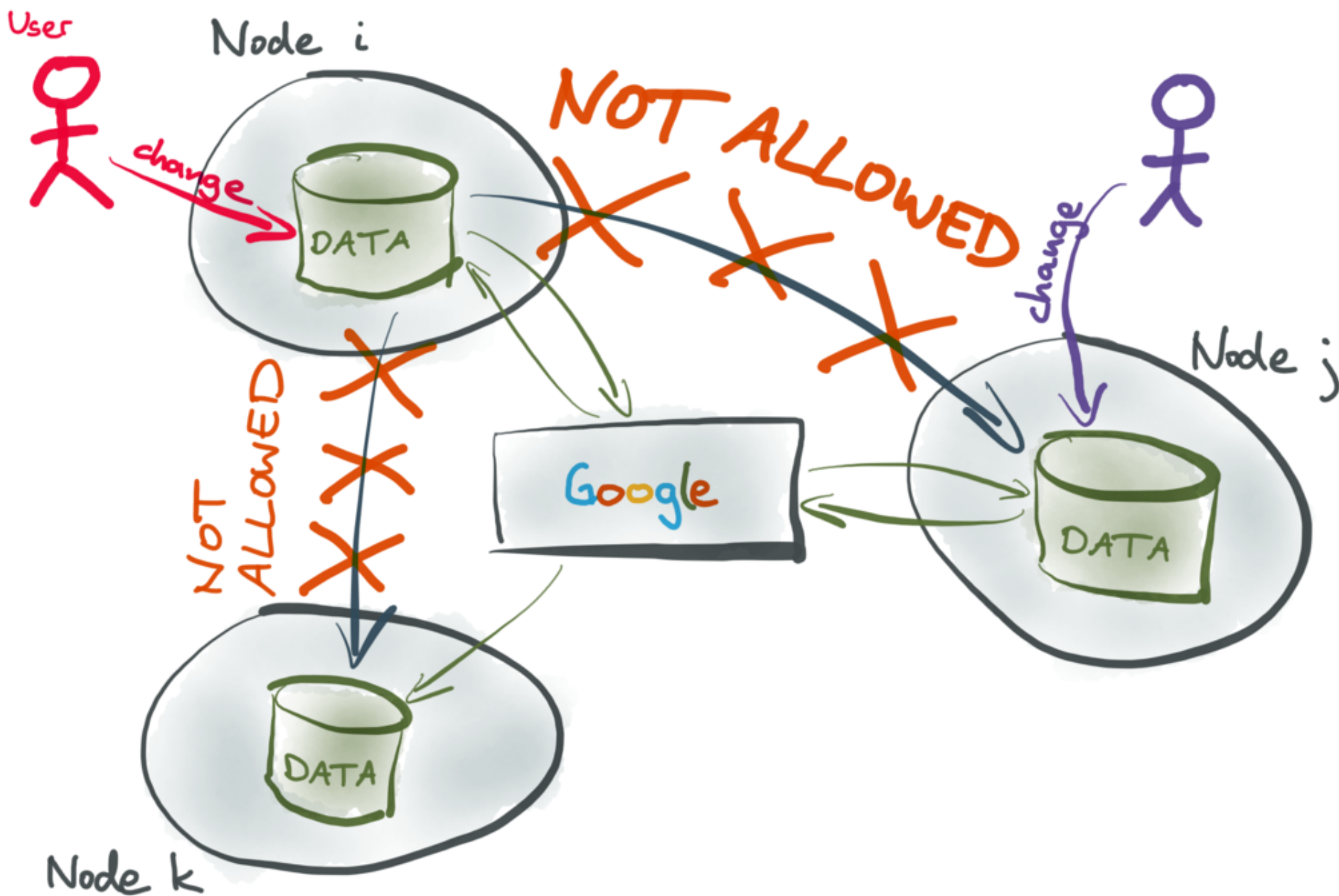
Oster et al.

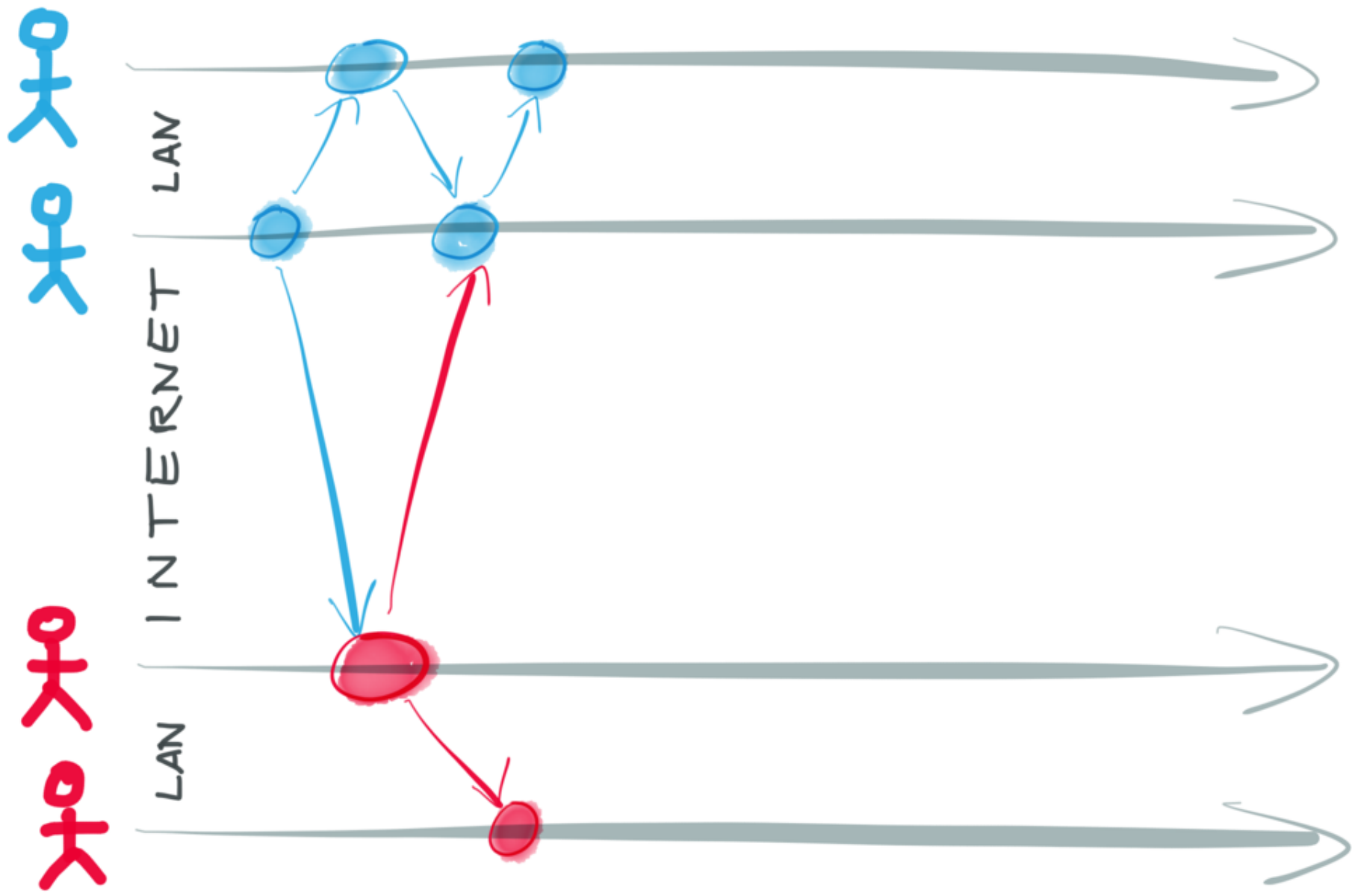
2006

OPERATIONAL TRANSFORMATION IN GOOGLE DOCS.

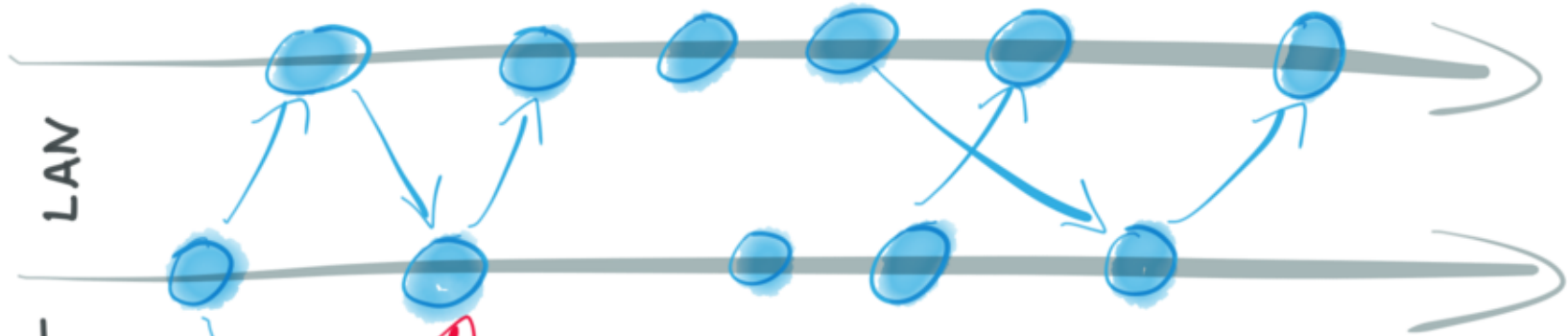


OPERATIONAL TRANSFORMATION IN GOOGLE DOCS.





人 人

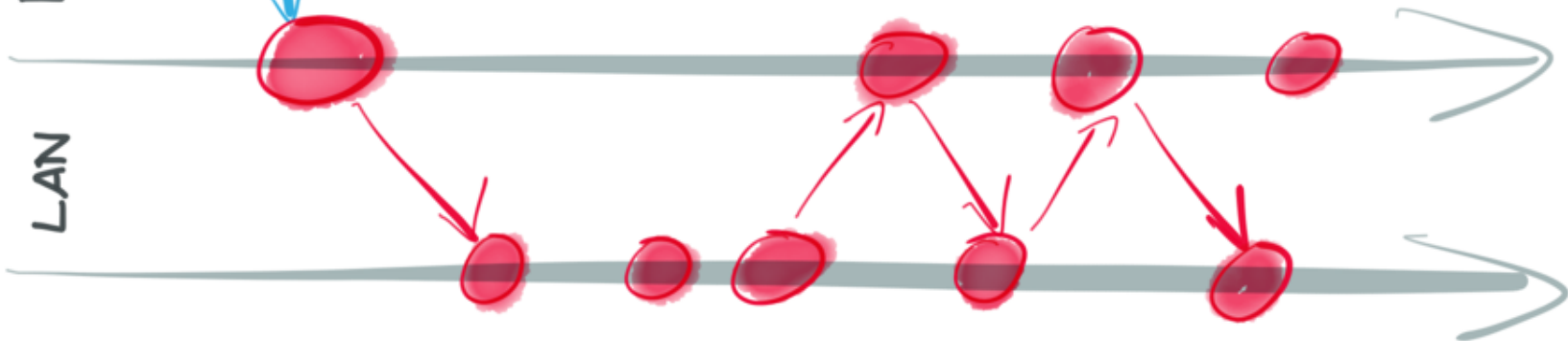


LAN

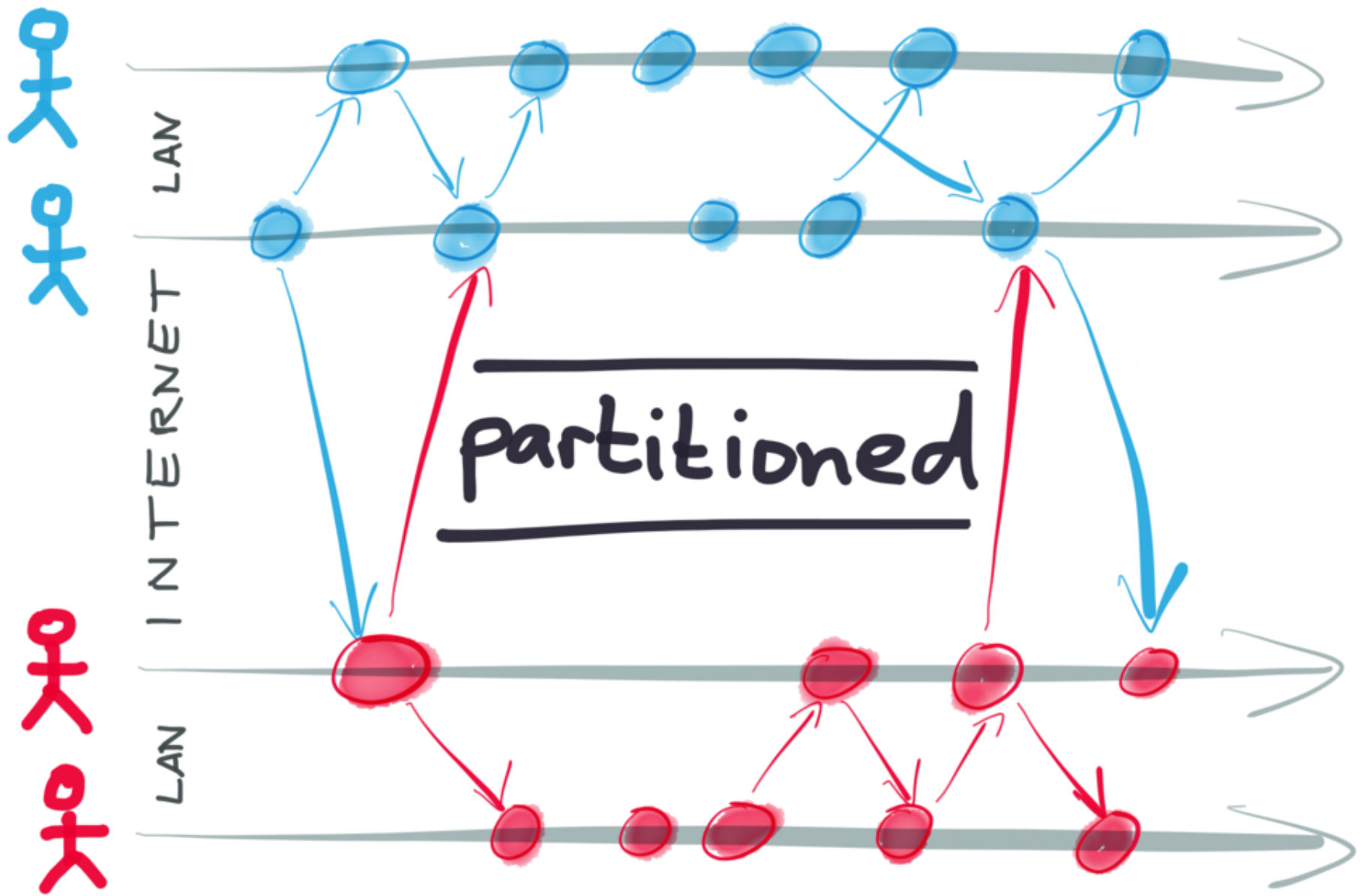
INTERNET

partitioned

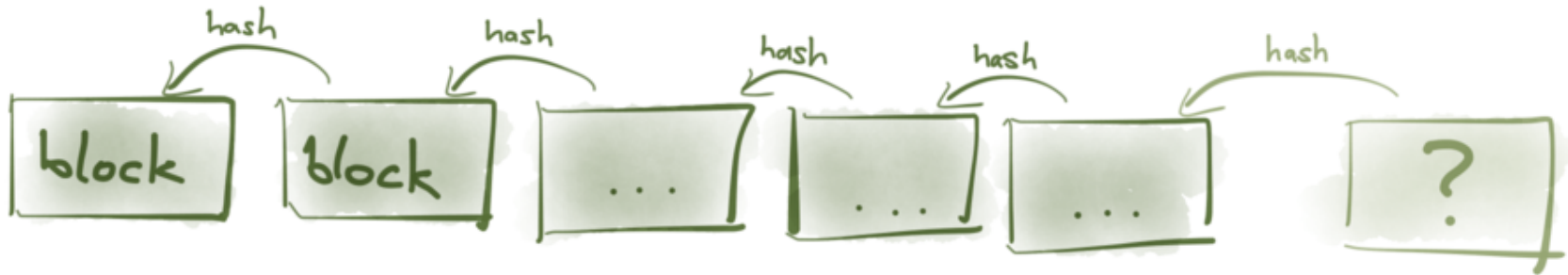
人 人



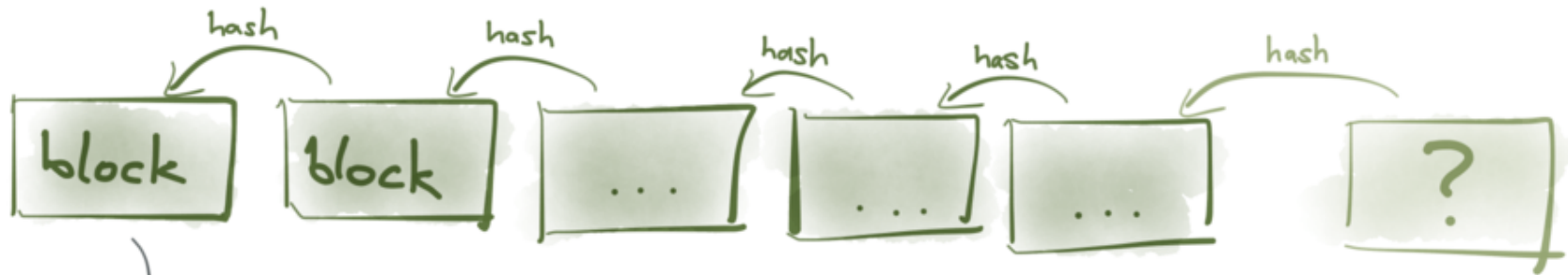
LAN



DECENTRALIZATION = BLOCKCHAINS?

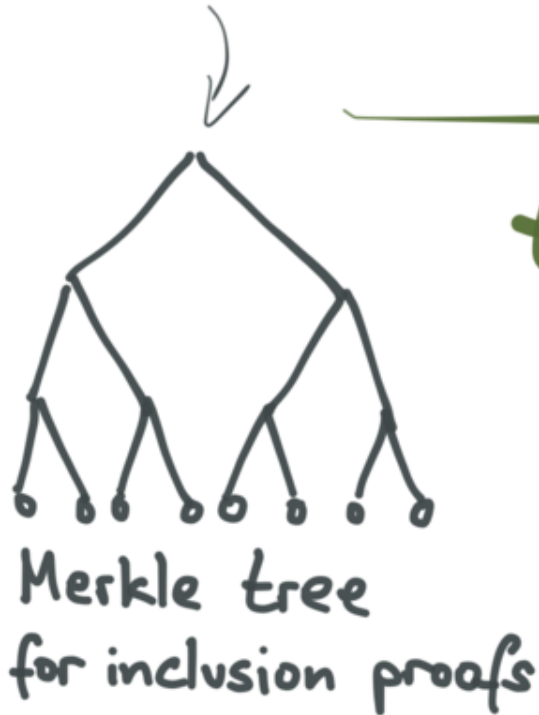
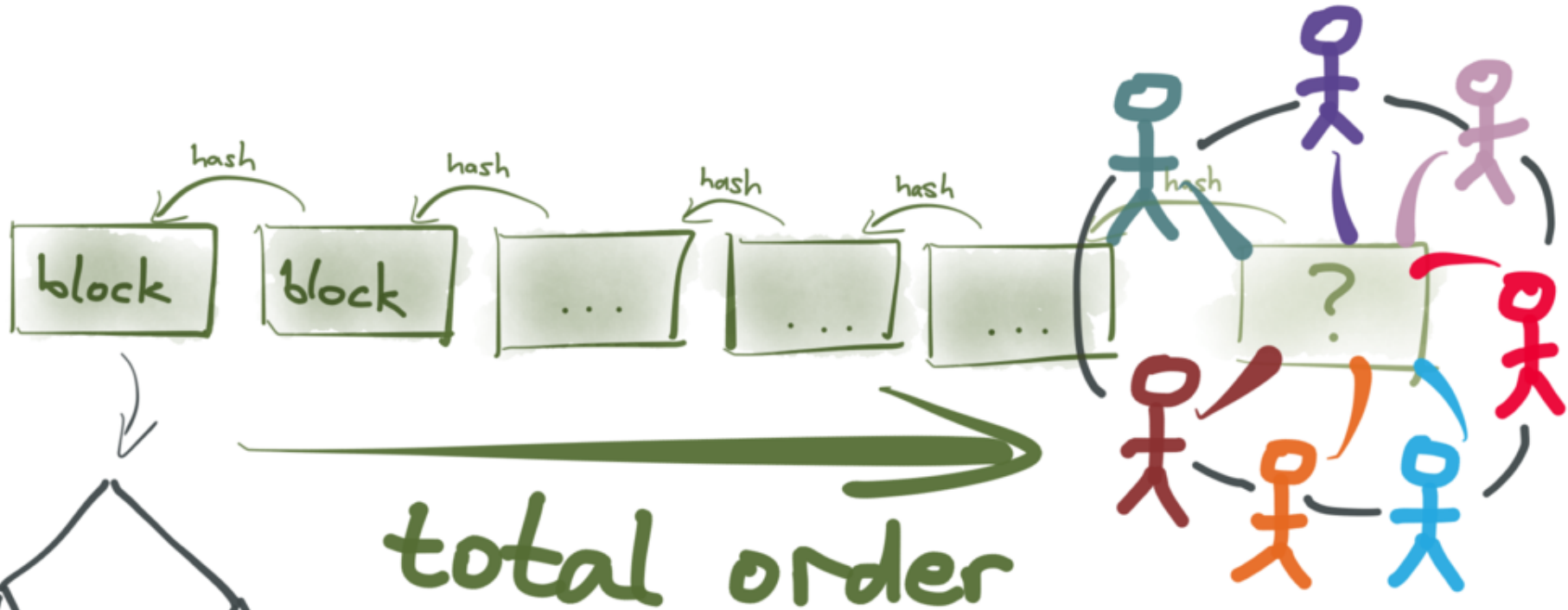


DECENTRALIZATION = BLOCKCHAINS?



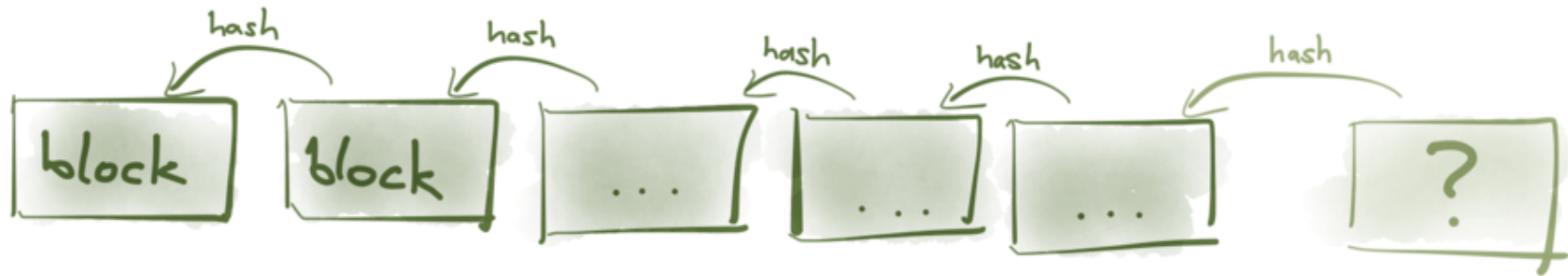
Merkle tree
for inclusion proofs

DECENTRALIZATION = BLOCKCHAINS?



Byzantine
consensus

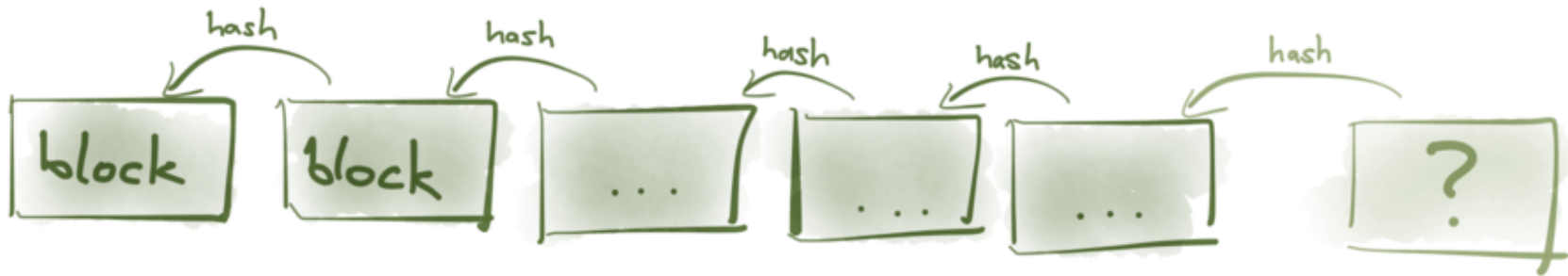
DECENTRALIZATION = BLOCKCHAINS?



Total order

required for cryptocurrencies
(to prevent double-spending)

DECENTRALIZATION = BLOCKCHAINS?



Total order

required for cryptocurrencies
(to prevent double-spending)

Consensus = pick one of several proposed values

Collaboration = keep all edits and merge them

Algorithms for convergence

OPERATIONAL TRANSFORMATION (OT)

- e.g. Google Docs, MS Office Online

1989-2006

CONFLICT-FREE REPLICATED DATA TYPES (CRDTs)

- e.g. Riak, TomTom GPS, Teletype for Atom, ...

2006 - present

PROVING CRDTs CORRECT

RGA

OpSet

Counter

...

STRONG EVENTUAL CONSISTENCY (SEC)

NETWORK MODEL



PROVING CRDTs CORRECT



assumptions

STRONG EVENTUAL CONSISTENCY (SEC)

NETWORK MODEL



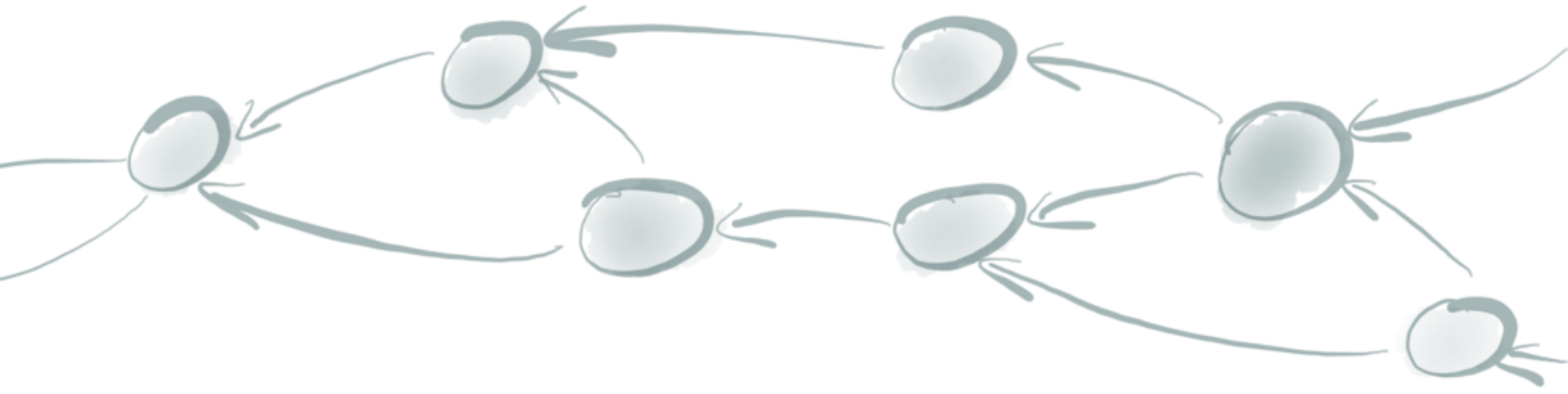
PROVING CRDTs CORRECT



assumptions
always satisfied!

For details, see our paper at <https://doi.org/10.1145/3133933>

Victor B. F. Gomes, Martin Kleppmann, Dominic P. Mulligan, and Alastair R. Beresford:
Verifying Strong Eventual Consistency in Distributed Systems. PACMPL 1(OOPSLA), 2017.



Automerge

<https://github.com/automerge/automerge>

Automerge

(data model / storage)

Apps

Apps

Automerge (data model / storage)

Apps Apps Trellis

Automerge (data model / storage)

Trellis, a Trello clone based on Automerge: <https://github.com/automerge/trellis>
Joint work with Orion Henry, Peter van Hardenberg, Roshan Choxi, and Adam Wiggins.

The Avengers Initiative

PROSPECTS ×

Spiderman



Doctor Strange

Star Lord

Groot

Add a card...

RECRUITING ×

Iron Man



Add a card...

JOINED THE TEAM ×

Black Widow



The Hulk



Add a card...

Apps

Apps

Trellis

Pixelpusher

Automerge (data model / storage)

Pixelpusher, a collaborative pixel art editor: <https://github.com/automerge/pixelpusher>
Created by Javier Valencia, Jeff Peterson, Peter van Hardenberg, and Jim Pick.

PIXEL ART TO CSS

by [JVALEN](#)

Star 324





+    25    50    75    100

NEW

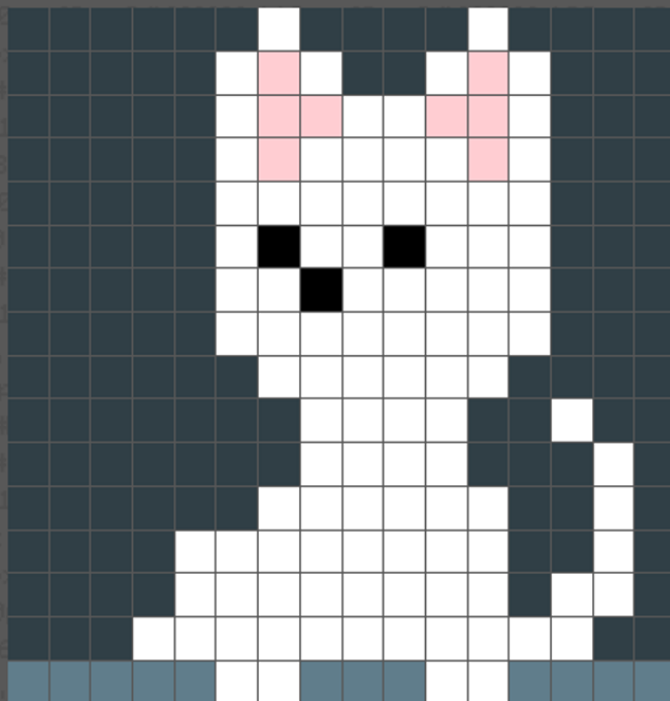
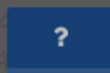
LOAD

SAVE



CSS



PREVIEW

RESET



16 +
-



16 +
-

Pixel Size
10

Duration
1

Preview

Reset



Pixel Size

10

Duration

1



Import Image

Eyeballs



Versions

- Eyeballs**
Thumbnail icons: [OK] [OK] [OK] [OK]
- Potato M...**
Thumbnail icons: [OK] [OK] [OK] [OK]
- Groucho...**
Thumbnail icons: [OK] [OK] [OK] [OK]

Archivers +

0

+

Apps

Apps

Trellis

Pixelpusher

Automerge (data model / storage)

MPL

WebRTC

MPL, a WebRTC network layer for Automerge: <https://github.com/automerge/mpl>
Joint work with Orion Henry, Peter van Hardenberg, Roshan Choxi, and Adam Wiggins.

Apps

Apps

Trellis

Pixelpusher

Automerge (data model / storage)

MPL

Hypermerge

WebRTC

Hypercore / Dat

Hypermerge, a peer-to-peer network layer: <https://github.com/automerge/hypermerge>
Created by Jim Pick, Jeff Peterson, and Peter van Hardenberg.

Apps

Apps

Trellis

Pixelpusher

Automerger (data model / storage)

MPL

Hypermerge

WebSocket
Client / Server

WebRTC

Hypercore / Dat

APPLICATION DATA

(JSON)

```
{ "to-do": [  
  { "title": "buy milk",  
    "done": false },  
  { "title": "water plants",  
    "done": false }  
],  
  "settings": { "alert-sound": "ring", ... }  
}
```


APPLICATION DATA

(JSON)

```
{ "to-do": [
  { "title": "buy milk",
    "done": false },
  { "title": "water plants",
    "done": false }
],
  "settings": { "alert-sound": "ring", ... }
}
```

ordered list

APPLICATION DATA

(JSON)

```
{ "to-do": [
  { "title": "buy milk",
    "done": false },
  { "title": "water plants",
    "done": false }
],
"settings": { "alert-sound": "ring", ... }
}
```

ordered list

nested maps

EDITING OPERATIONS

```
{ "to-do": [  
  { "title": "buy milk",  
    "done": false },  
  { "title": "water plants",  
    "done": false  
           true } value assignment  
],  
  "settings": { "alert-sound": "ring", ... }  
}
```

EDITING OPERATIONS

```
{ "to-do": [  
  { "title": "buy soy milk",  
    "done": false },  
  { "title": "water plants",  
    "done": false }  
],  
  "settings": { "alert-sound": "ring", ... }  
}
```

string editing

value assignment

EDITING OPERATIONS

```
{ "to-do": [  
  { "title": "buy soy milk",  
    "done": false },  
  { "title": "water plants",  
    "done": false  
      true }  
],  
  "settings": { "alert-sound": "ring", ... }  
}
```

string editing

list insertion

value assignment

The diagram shows a JSON object with a 'to-do' array and a 'settings' object. Three editing operations are highlighted: 1. 'string editing' where the word 'soy' is inserted into the title 'buy milk'. 2. 'list insertion' where a new object with 'title' and 'phone num' is added to the 'to-do' array. 3. 'value assignment' where the 'done' value of the second object is changed from 'false' to 'true'.

EDITING OPERATIONS

```
{ "to-do": [  
  { "title": "buy milk",  
    "done": false },  
  { "title": "water plants",  
    "done": false  
      true }  
],  
  "settings": { "alert-sound": "ring", ... }  
}
```

string editing

soy

list insertion

{ "title":
 "phone num",
 done: false }

value assignment

put map key

"background-image": "..."

state = Automerge.change (state, "Add todo item",

(doc) => {

doc.todos.push ({

title: "Buy milk",

done: false

})

})

state is immutable
(Automerge.change() returns
new object)

state = Automerge.change(state, "Add todo item",

(doc) => {

doc.todos.push({

title: "Buy milk",

done: false

})

})

state is immutable
(Automerge.change() returns
new object)

"Commit message"
(optional)

state = Automerge.change(state, "Add todo item",

(doc) => {

doc.todos.push({

title: "Buy milk",

done: false

})

})

state is immutable
(Automerge.change() returns
new object)

"Commit message"
(optional)

state = Automerge.change(state, "Add todo item",

(doc) => {

doc is mutable
only within
this block
(Proxy object)

doc.todos.push({
title: "Buy milk",
done: false
})

})

state is immutable
(Automerge.change() returns
new object)

"Commit message"
(optional)

state = Automerge.change(state, "Add todo item",

(doc) => {

doc is mutable
only within
this block
(Proxy object)

doc.todos.push({
 title: "Buy milk",
 done: false
})

}

Plain old JS
objects and methods

```
doc.todos.push({  
  title: "Buy milk",  
  done: false  
})
```

operation log



```
{action: "makeMap", obj: id1}
```

```
{action: "set", obj: id1, key: "title", value: "Buy milk"}
```

```
{action: "set", obj: id1, key: "done", value: false}
```

```
{action: "ins", obj: todosID, key: prevID, elem: 15}
```

```
{action: "link", obj: todosID, key: elem15, value: id1}
```

≡ TIME TRAVEL ≡

Automerge.getHistory(*state*)

TIME TRAVEL

Automerge.getHistory (state)

⇒ [{ change: { message: "Add todo item", ... },
 snapshot: { todos: [{ title: "Buy milk", ... }, ...] } },
 { change: { message: "Mark item as done", ... },
 snapshot: { todos: [{ title: "Buy milk", ... }, ...] } },
 ...
]

— CONCURRENT CHANGES —

```
{ todos : [  
  { title : "Water plants",  
    done : false }  
]}
```

```
{ todos : [  
  { title : "Water plants",  
    done : false }  
]}
```


CONCURRENT CHANGES

```
{ todos: [  
  { title: "Water plants",  
    done: false }  
]}
```



```
doc.todos.push ({  
  title: "Buy milk",  
  done: false  
})
```

```
{ todos: [  
  { title: "Water plants",  
    done: false }  
]}
```



```
doc.todos[0].done = true
```

— CONCURRENT CHANGES —

```
doc.todos.push ({  
  title: "Buy milk",  
  done: false  
})
```

```
doc.todos[0].done = true
```

CONCURRENT CHANGES

```
doc.todos.push ({  
  title: "Buy milk",  
  done: false  
})
```



```
{ todos: [  
  { title: "Water plants",  
    done: false },  
  { title: "Buy milk",  
    done: false }  
]}
```

```
doc.todos[0].done = true
```



```
{ todos: [  
  { title: "Water plants",  
    done: true }  
]}
```

CONCURRENT CHANGES

```
doc.todos.push ({  
  title: "Buy milk",  
  done: false  
})
```

```
doc.todos[0].done = true
```

network communication

```
{ todos: [  
  { title: "Water plants",  
    done: true },  
  { title: "Buy milk",  
    done: false }  
]}
```

```
{ todos: [  
  { title: "Water plants",  
    done: true },  
  { title: "Buy milk",  
    done: false }  
]}
```



Convergence guarantee:

Any two nodes have seen the
same set of operations
(but maybe in a different order!)



They are in the same state

— CONCURRENT CHANGES —

```
doc.todos[0].title =  
  "Buy soy milk"
```

```
doc.todos[0].title =  
  "Buy almond milk"
```

CONCURRENT CHANGES

doc.todos[0].title =
"Buy soy milk"

doc.todos[0].title =
"Buy almond milk"

network communication

{ todos: [
 { title: "Buy soy milk",
 done: false,
 _conflicts: { title: {
 1234: "Buy almond milk"
 } } }
]

{ todos: [
 { title: "Buy soy milk",
 done: false,
 _conflicts: { title: {
 1234: "Buy almond milk"
 } } }
]

NODE A:

Hey guys!

NODE B:

Hey guys!

NODE A:

Hey guys!

↙ edit

Hey ~~guys~~ everyone!

NODE B:

Hey guys!

↙ edit

Hey ~~guys~~ folks!

Node A:

Hey guys!

edit

Hey ~~guys~~ everyone!

Node B:

Hey guys!

edit

Hey ~~guys~~ folks!

Hey everyone folks!

Hey everyone folks!



ORDERED LIST CRPT (NUTSHELL)

NODE A:



NODE B:



ORDERED LIST CRPT (NUTSHELL)

NODE A:



edit ID:4a



NODE B:



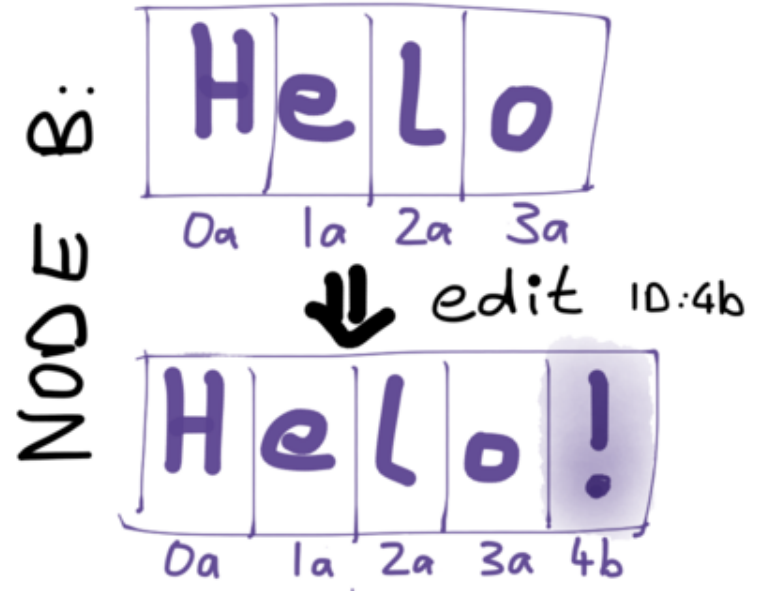
edit ID:4b



ORDERED LIST CRDT (NUTSHELL)



insert "l"
with id 4a
after id 2a



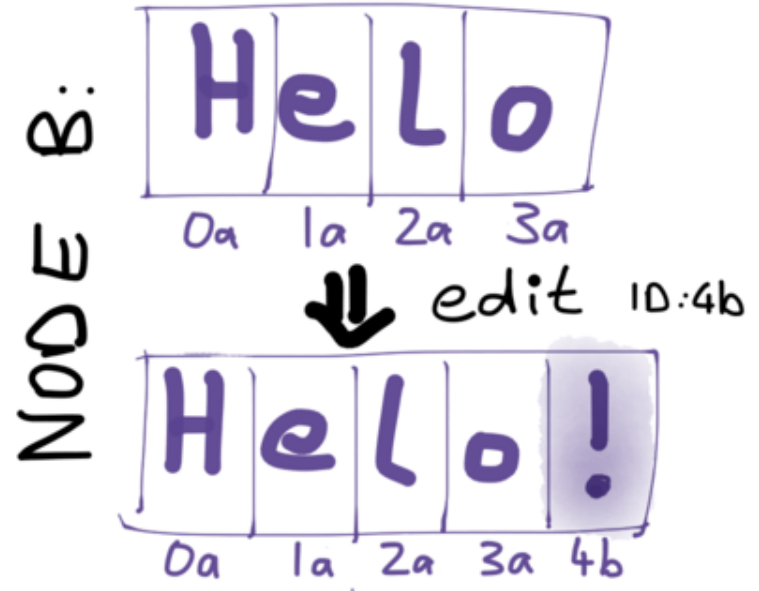
insert "!" with
id 4b after id 3a



ORDERED LIST CRDT (NUTSHELL)



insert "L"
with id 4a
after id 2a



insert "!" with
id 4b after id 3a

insert "L" with
id 4a after id 2a



ORDERED LIST CRDT (NUTSHELL)

NODE A:



edit ID:4a



insert "L" with id 4a after id 2a

NODE B:



edit ID:4b



insert "!" with id 4b after id 3a



insert "!" with id 4b after id 3a



insert "L" with id 4a after id 2a



ORDERED LIST CRDT (NUTSHELL)

NODE A:



edit ID: 4a



insert "l" with id 4a after id 2a



insert "!" with id 4b after id 3a



NODE B:



edit ID: 4b



insert "!" with id 4b after id 3a

insert "l" with id 4a after id 2a



INSERTING IN THE SAME PLACE

a	b	c
1a	2a	3a

a	b	c
1a	2a	3a

INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit
4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit
5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	b	c
---	---	---

1a 2a 3a

INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit 4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit 5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	b	c
---	---	---

1a 2a 3a

edit 4b

a	p	b	c
---	---	---	---

1a 4b 2a 3a

edit 5b

a	p	q	b	c
---	---	---	---	---

1a 4b 5b 2a 3a

INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit 4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit 5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	p	x	y	b	c
---	---	---	---	---	---

1a 4b 4a 5a 2a 3a

insert "p"
with id 4b
after id 1a

a	b	c
---	---	---

1a 2a 3a

edit 4b

a	p	b	c
---	---	---	---

1a 4b 2a 3a

edit 5b

a	p	q	b	c
---	---	---	---	---

1a 4b 5b 2a 3a

INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit 4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit 5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	p	x	y	b	c
---	---	---	---	---	---

1a 4b 4a 5a 2a 3a

a	p	q	x	y	b	c
---	---	---	---	---	---	---

1a 4b 5b 4a 5a 2a 3a

insert "p"
with id 4b
after id 1a

a	b	c
---	---	---

1a 2a 3a

edit 4b

a	p	b	c
---	---	---	---

1a 4b 2a 3a

edit 5b

a	p	q	b	c
---	---	---	---	---

1a 4b 5b 2a 3a

insert "q"
with id 5b
after id 4b

INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit 4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit 5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	p	x	y	b	c
---	---	---	---	---	---

1a 4b 4a 5a 2a 3a

a	p	q	x	y	b	c
---	---	---	---	---	---	---

1a 4b 5b 4a 5a 2a 3a

insert "x"
with id 4a
after id 1a

a	b	c
---	---	---

1a 2a 3a

edit 4b

a	p	b	c
---	---	---	---

1a 4b 2a 3a

edit 5b

a	p	q	b	c
---	---	---	---	---

1a 4b 5b 2a 3a

a	p	q	x	b	c
---	---	---	---	---	---

1a 4b 5b 4a 2a 3a



INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit 4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit 5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	p	x	y	b	c
---	---	---	---	---	---

1a 4b 4a 5a 2a 3a

a	p	q	x	y	b	c
---	---	---	---	---	---	---

1a 4b 5b 4a 5a 2a 3a

insert "x"
with id 4a
after id 1a

a	b	c
---	---	---

1a 2a 3a

edit 4b

a	p	b	c
---	---	---	---

1a 4b 2a 3a

edit 5b

a	p	q	b	c
---	---	---	---	---

1a 4b 5b 2a 3a

a	p	q	x	b	c
---	---	---	---	---	---

1a 4b 5b 4a 2a 3a

4b > 4a

5b > 4a

2a < 4a

Skip over
any existing
list elements
with greater id

INSERTING IN THE SAME PLACE

a	b	c
---	---	---

1a 2a 3a

edit 4a

a	x	b	c
---	---	---	---

1a 4a 2a 3a

edit 5a

a	x	y	b	c
---	---	---	---	---

1a 4a 5a 2a 3a

a	p	x	y	b	c
---	---	---	---	---	---

1a 4b 4a 5a 2a 3a

a	p	q	x	y	b	c
---	---	---	---	---	---	---

1a 4b 5b 4a 5a 2a 3a

a	b	c
---	---	---

1a 2a 3a

edit 4b

a	p	b	c
---	---	---	---

1a 4b 2a 3a

edit 5b

a	p	q	b	c
---	---	---	---	---

1a 4b 5b 2a 3a

a	p	q	x	b	c
---	---	---	---	---	---

1a 4b 5b 4a 2a 3a

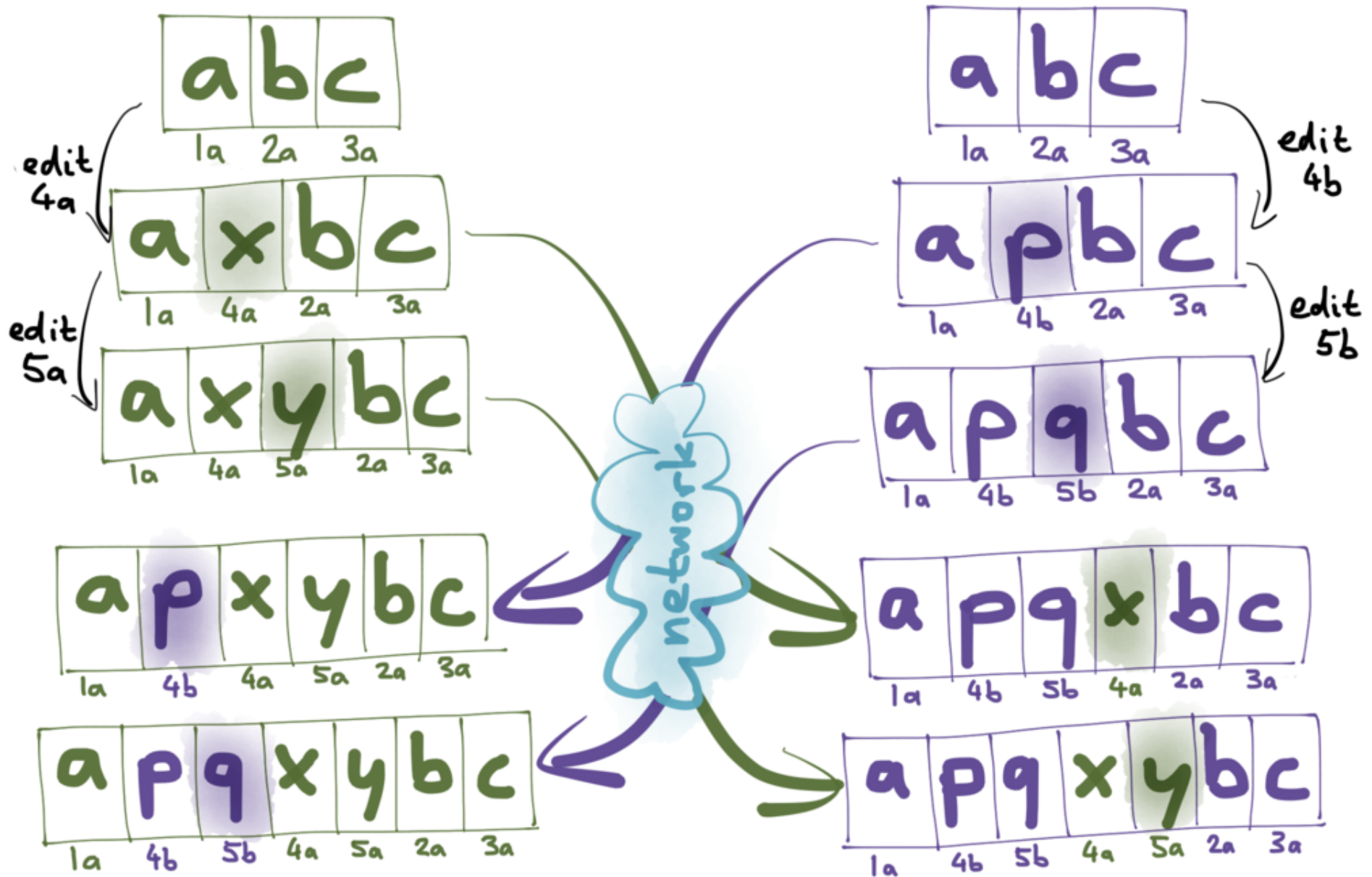
a	p	q	x	y	b	c
---	---	---	---	---	---	---

1a 4b 5b 4a 5a 2a 3a

insert "x"
with id 4a
after id 1a

insert "y"
with id 5a
after id 4a

INSERTING IN THE SAME PLACE



Resources

- Automerge: <https://github.com/automerge/automerge>
- Trellis: <https://github.com/automerge/trellis>
- Pixelpusher: <https://github.com/automerge/pixelpusher>
- MPL (WebRTC layer): <https://github.com/automerge/mpl>
- Hypermerge: <https://github.com/automerge/hypermerge>
- Dat / Hypercore: <https://datproject.org/>
- Proving CRDTs correct: <https://doi.org/10.1145/3133933>
- JSON CRDT: <http://arxiv.org/abs/1608.03960>
- My book: <http://dataintensive.net/>

dataintensive.net



@martinkl