

Automerge

Replicated data structures for
peer-to-peer collaboration

Martin Kleppmann, University of Cambridge @martinkl

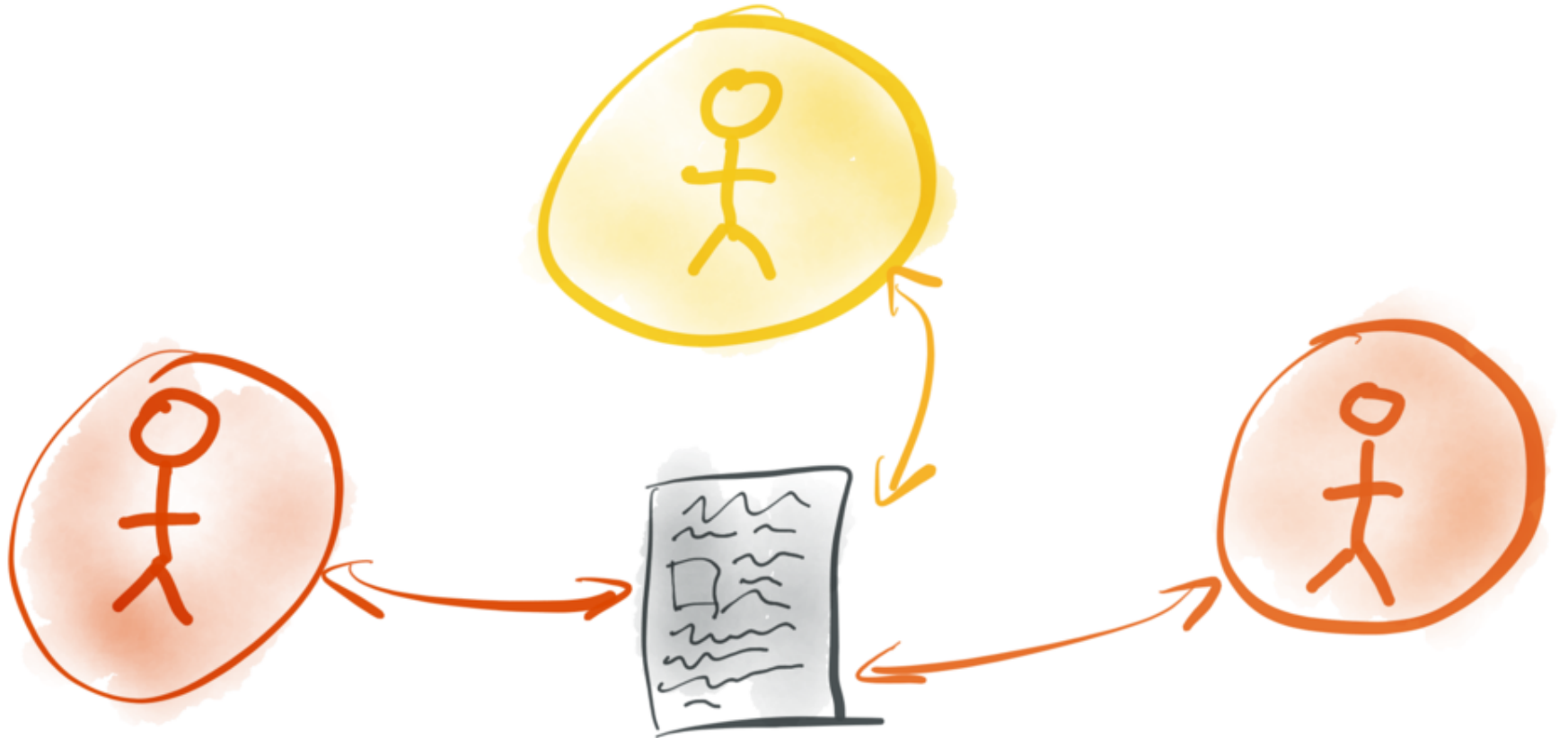
Joint work with Victor Gomes, Dominic Mulligan, Alastair Beresford, Peter van Hardenberg, Orion Henry, Adam Wiggins, Roshan Chori, Jeff Peterson, Jim Pick, & more contributors

dataintensive.net



@martinkl

COLLABORATIVE APPLICATIONS



Example: Text editing



Example: Text editing

insert "World"
after "Hello"



"Hello!"

"Hello World!"

time

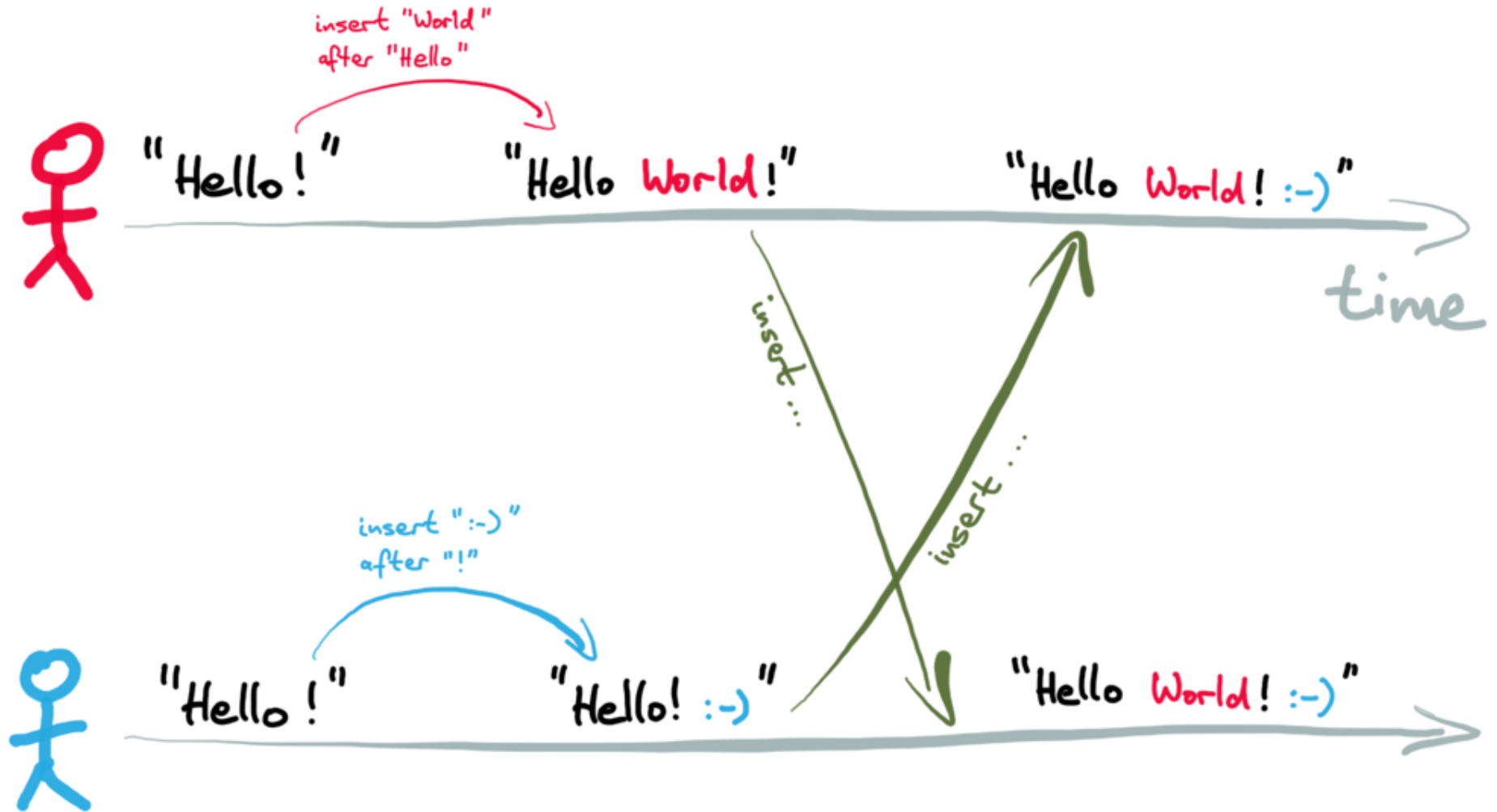
insert ":-)"
after "!"



"Hello!"

"Hello! :-)"

Example: Text editing



Algorithms for convergence

OPERATIONAL TRANSFORMATION (OT)

- e.g. Google Docs,
MS Office Online

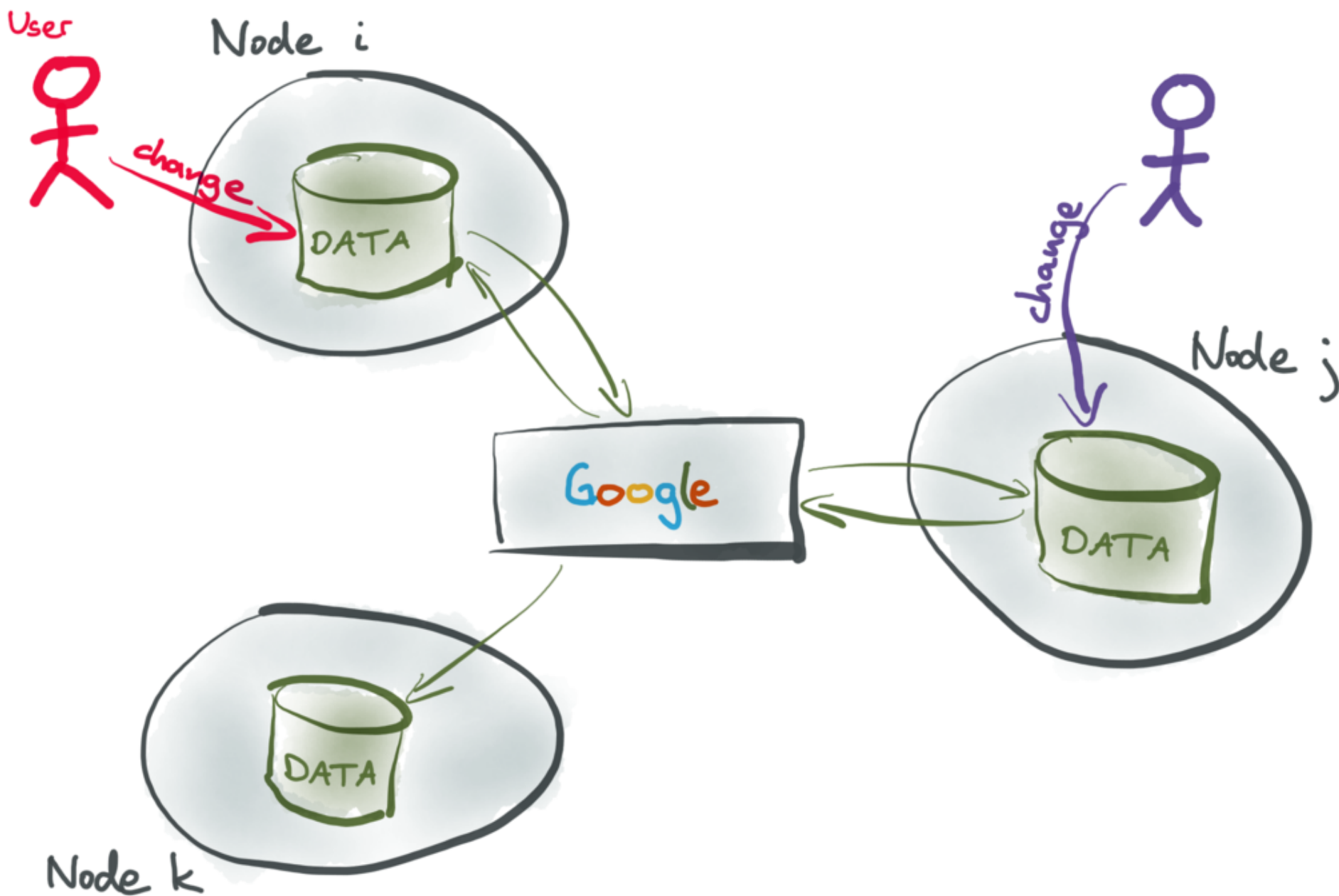
1989-2006

CONFLICT-FREE REPLICATED DATA TYPES (CRDTs)

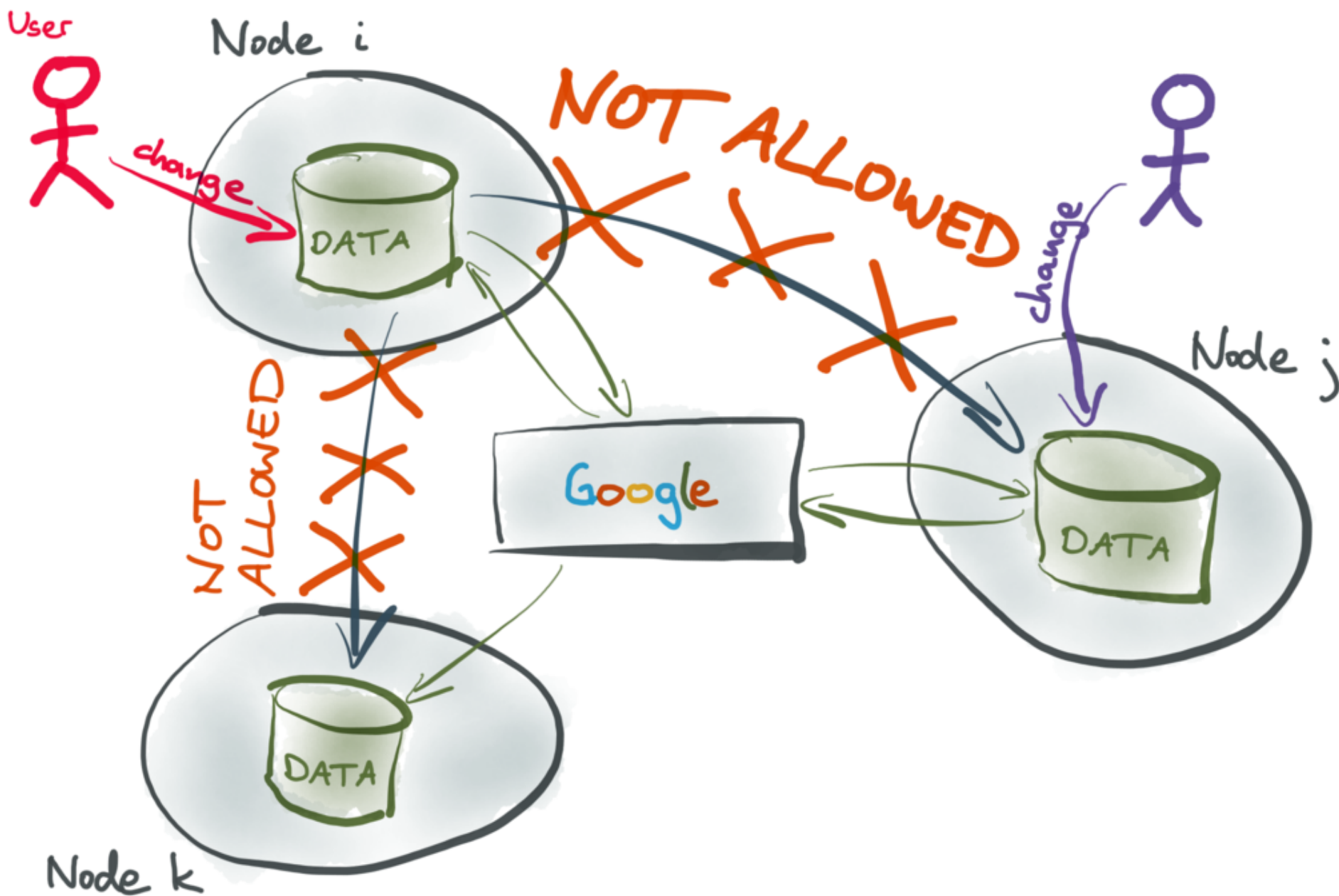
- e.g. Riak, TomTom GPS,
Teletype for Atom, ...

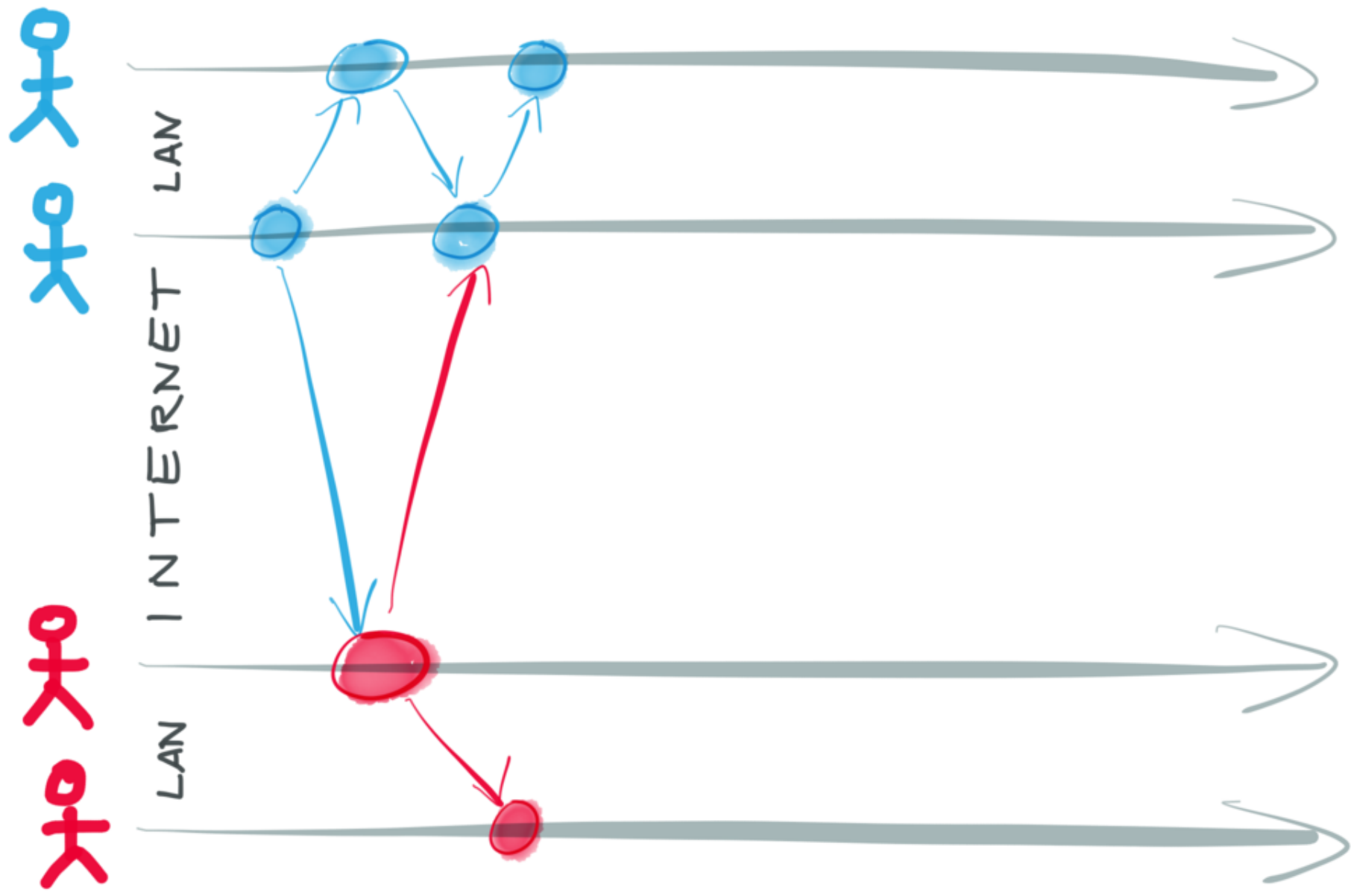
2006 - present

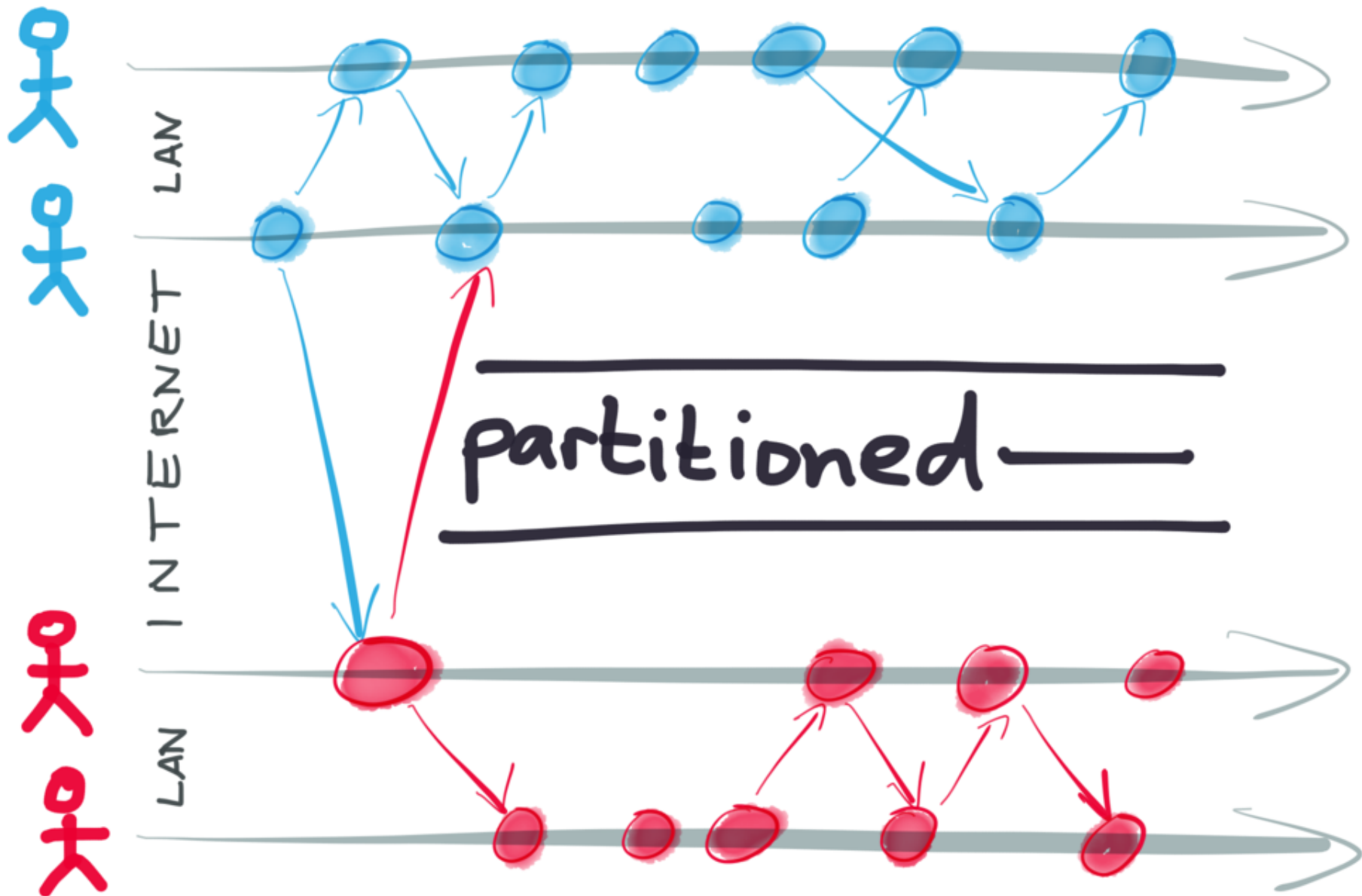
OPERATIONAL TRANSFORMATION IN GOOGLE DOCS.

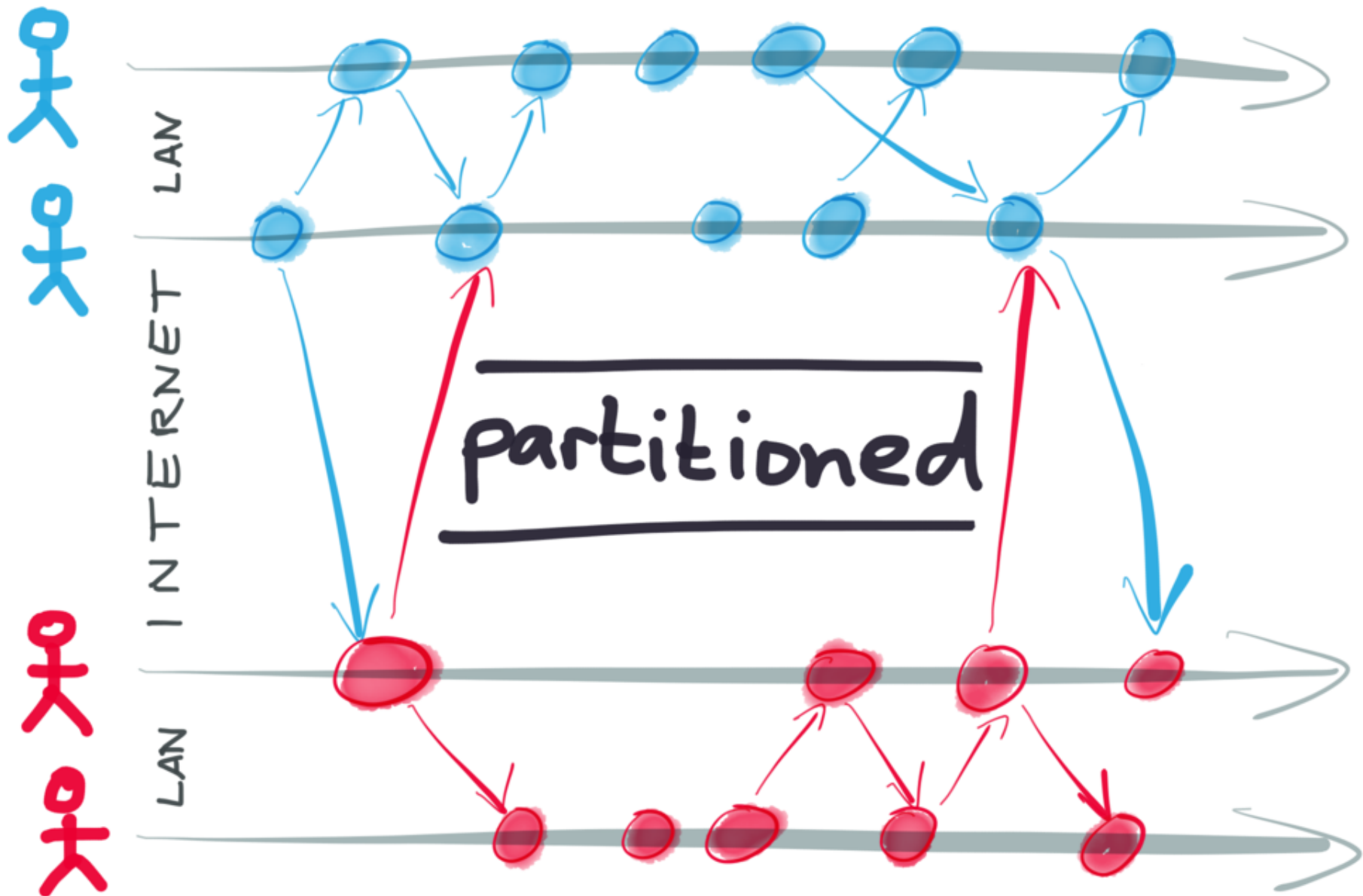


OPERATIONAL TRANSFORMATION IN GOOGLE DOCS.









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

ORSet

Counter

...

STRONG EVENTUAL CONSISTENCY (SEC)

NETWORK MODEL



PROVING CRDTs CORRECT



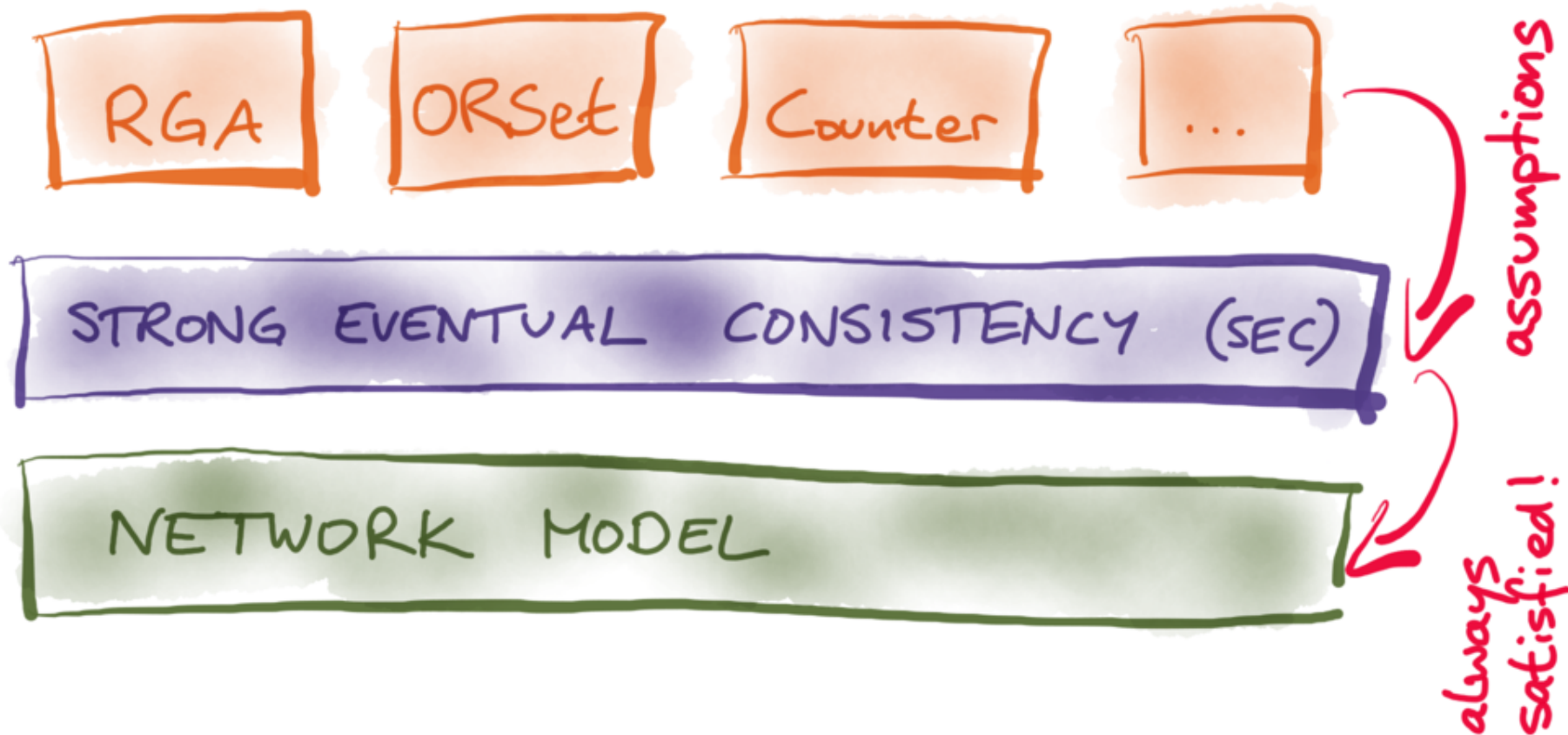
assumptions

STRONG EVENTUAL CONSISTENCY (SEC)

NETWORK MODEL



PROVING CRDTs CORRECT



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.



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

Automerge (data model / storage)

Apps

Apps

Automerge (data model / storage)



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

The Avengers Initiative

PROSPECTS



Spiderman



1

Doctor Strange

Star Lord

Groot

Add a card...

RECRUITING



Iron Man



Add a card...

JOINED THE TEAM



Black Widow



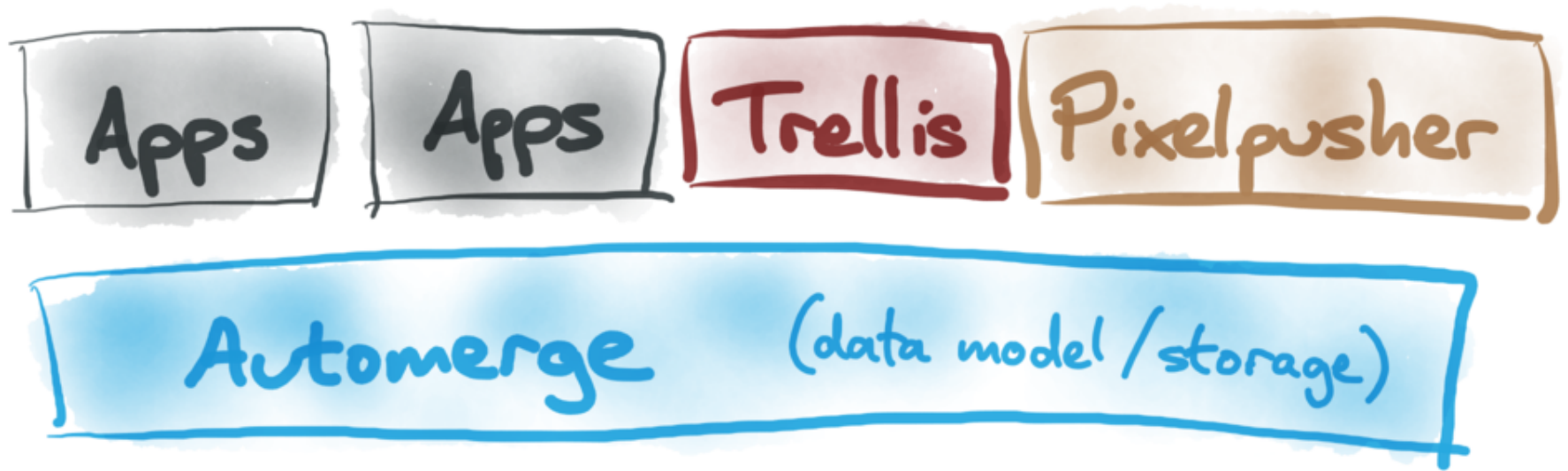
The Hulk



1



Add a card...



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



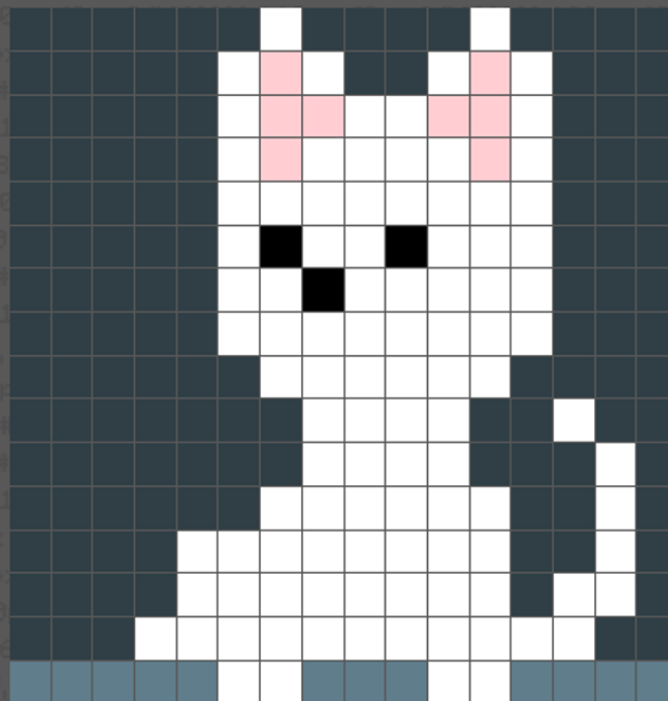
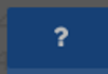
NEW

LOAD

SAVE



CSS



PREVIEW

RESET

16 + -

16 + -

Pixel Size
10

Duration
1

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 } value assignment  
             true
```

```
],
```

```
"settings": { "alert-sound": "ring", ... }
```

```
}
```

EDITING OPERATIONS

string editing

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

value assignment

EDITING OPERATIONS

string editing

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

list insertion

value assignment

soy

true

EDITING OPERATIONS

string editing

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

list insertion

value assignment

put map key

soy

phone num,

background-image": "..."

The diagram illustrates various JSON editing operations on a sample object. The object is: { "to-do": [{ "title": "buy milk", "done": false }, { "title": "water plants", "done": false }], "settings": { "alert-sound": "ring", ... } }. Annotations include: 'string editing' in orange above the first object; 'soy' in orange above 'buy'; 'list insertion' in blue above the second object, with a blue arrow pointing to the insertion point between the two objects; 'value assignment' in red above the 'done' field of the second object, with a red arrow pointing to the change from 'false' to 'true'; and 'put map key' in green above the 'settings' field, with a green arrow pointing to the 'background-image' key being added to the settings 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)

`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}
```




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 —

```
{ 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 }  
]}
```

— 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"
 } }
] }

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", ... }, ...] } },
 ...
]

Resources

- Martin's email: mk428@cl.cam.ac.uk
- Martin on Twitter: [@martinkl](https://twitter.com/martinkl)
- 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>
- Martin's book: <http://dataintensive.net/>