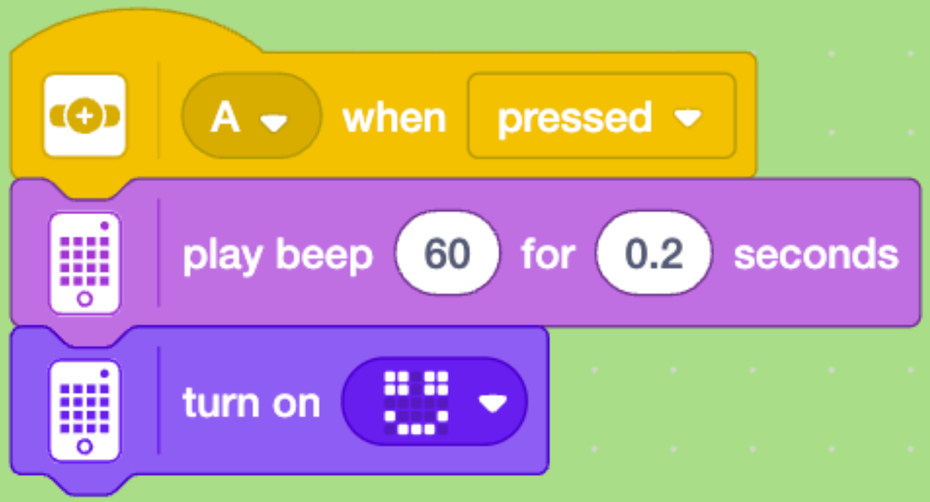
# You Might Not Need A CRDT

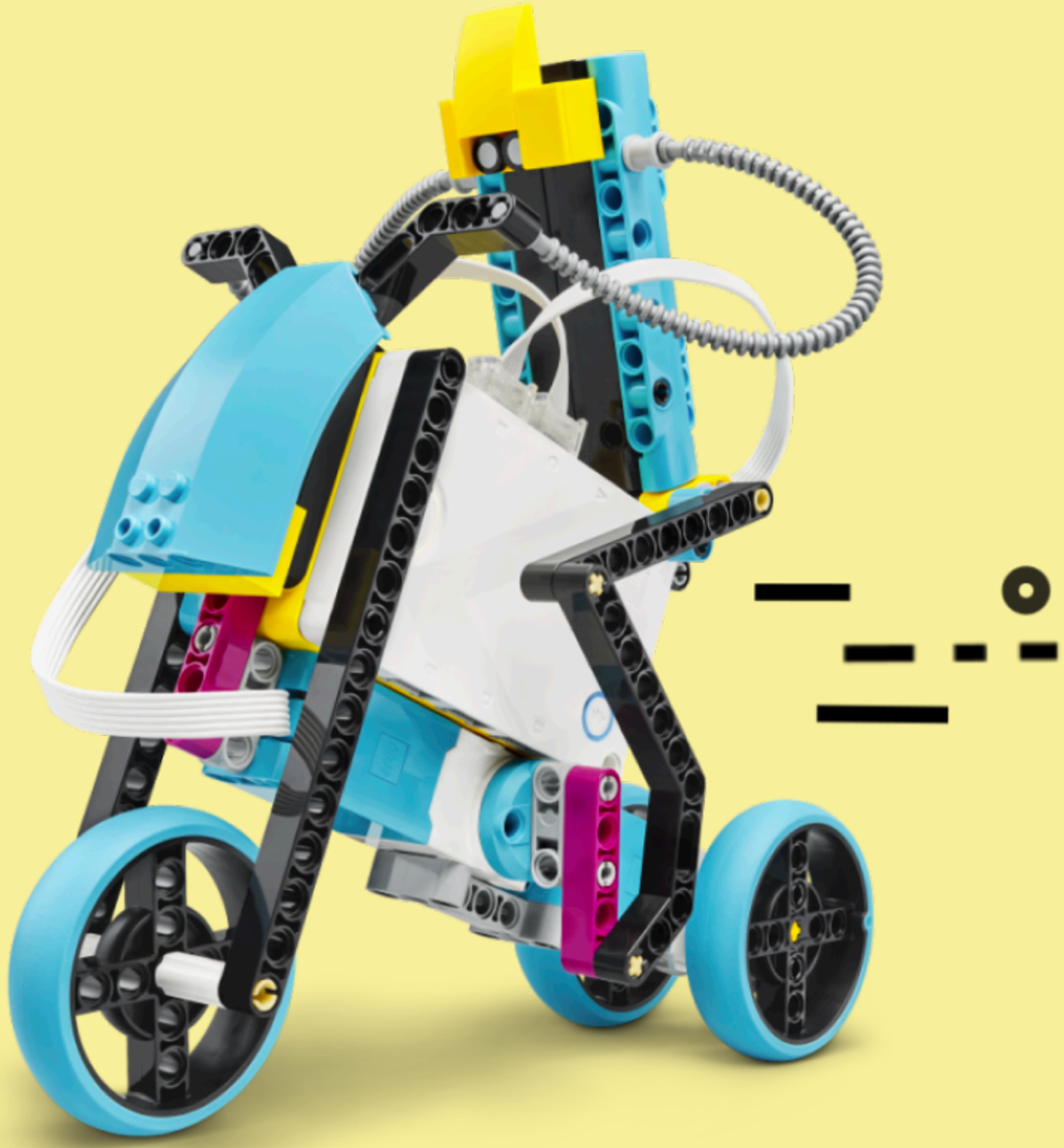
# Robots and Collaborative Coding

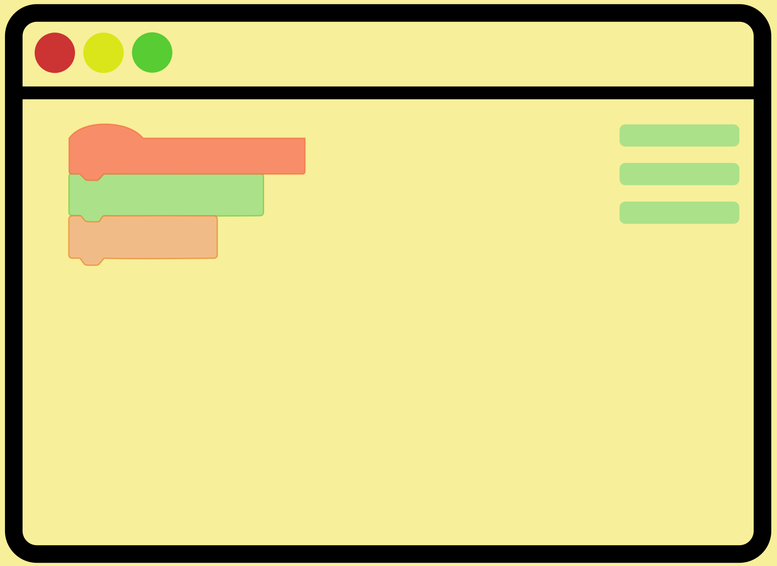
## Or: Stuff I Learned While Playing with Phoenix

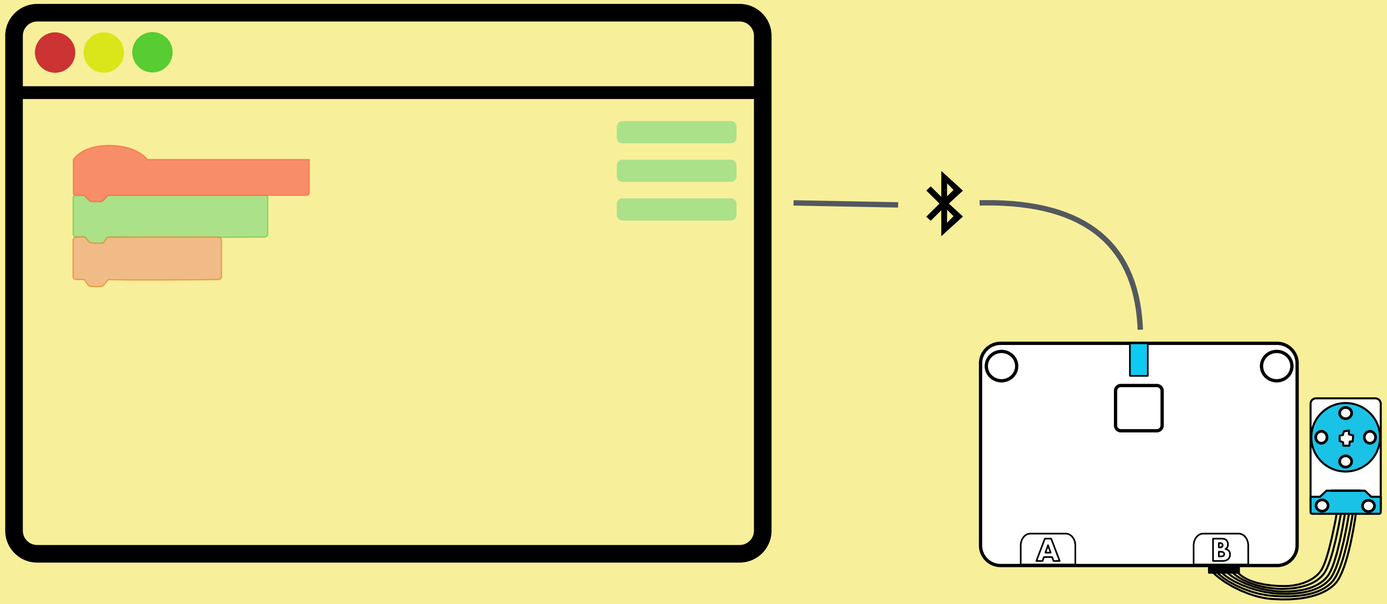

A
when colour is
start moving
right: 30

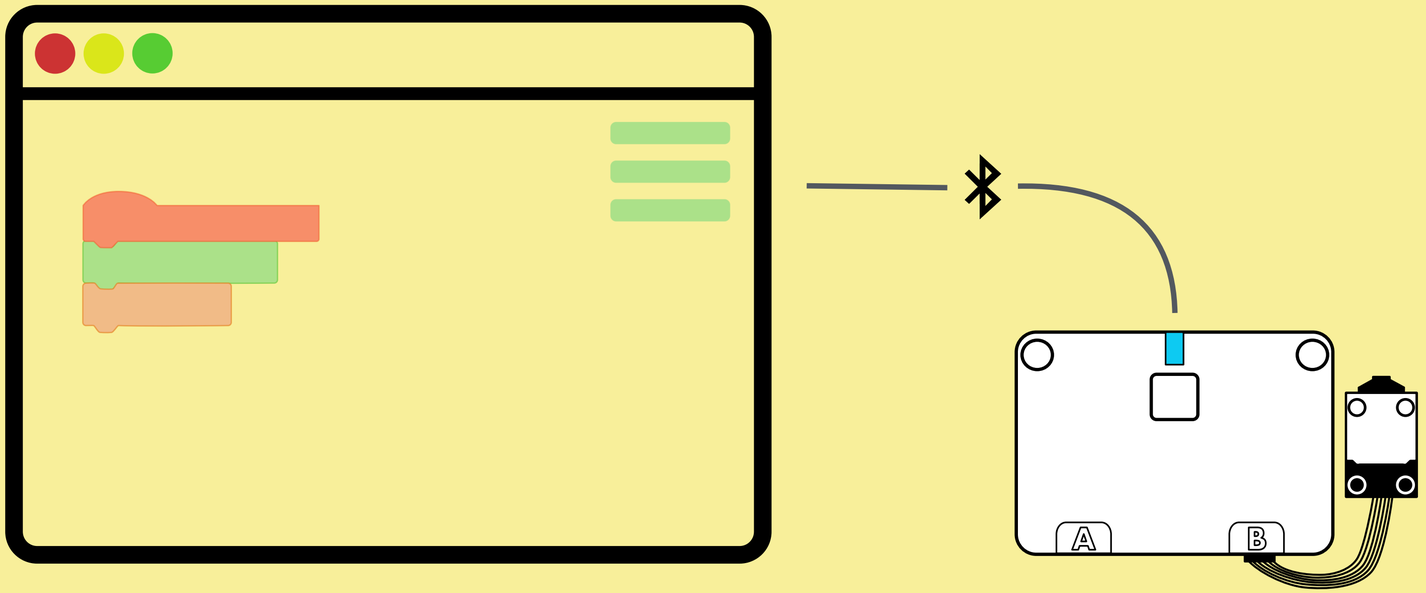

A
when colour is
start moving
left: -30


A
when
pressed
play beep
60
for
0.2
seconds
turn on

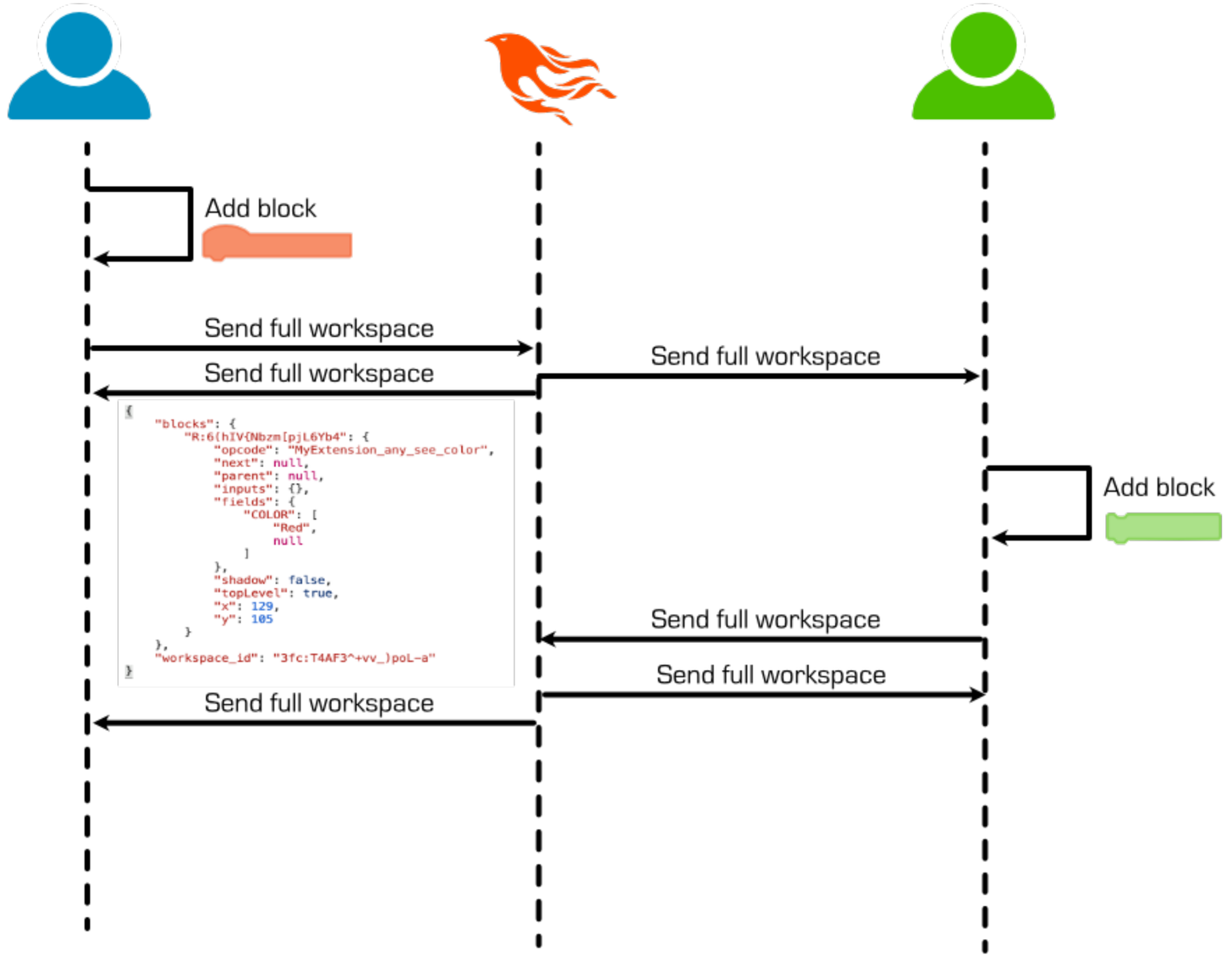


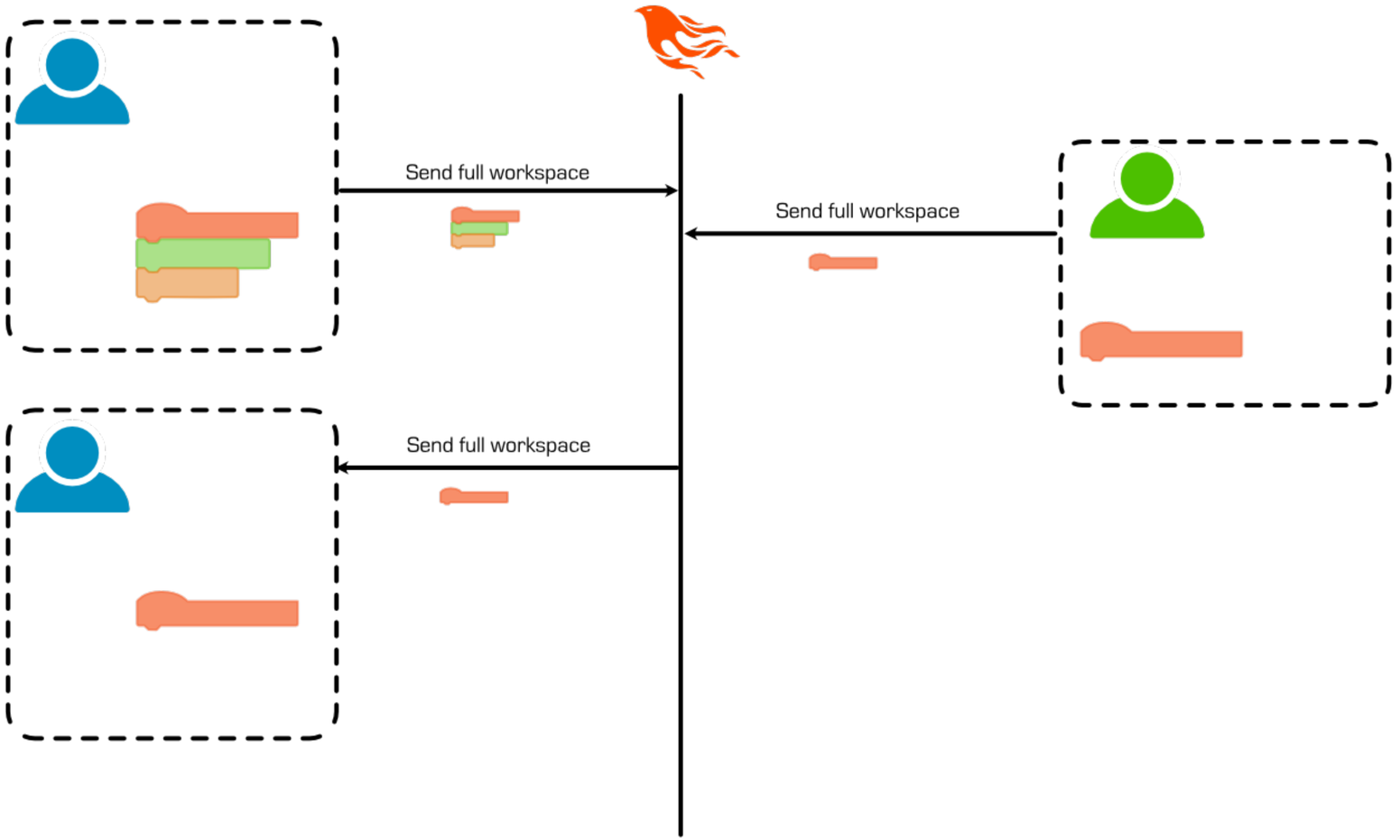



A
B


A
B

# The Edit Flow



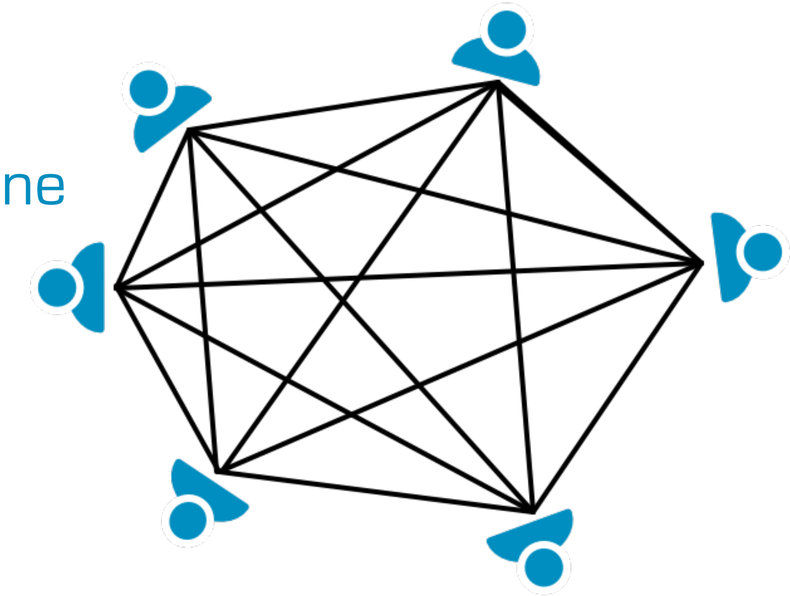

Send full workspace
Send full workspace
Send full workspace

# What then?

# CRDT: Conflict-Free Replicated Data Type

# Some properties of a nice CRDT

- Convergence - if everyone gets all the edits everyone ends up with the same document
- Preserves user intention - the document ends up reflecting what the user wanted to do
- Local first - local edits are applied instantly
- Supports offline editing - you don’t have to be constantly online to participate
- No central server needed

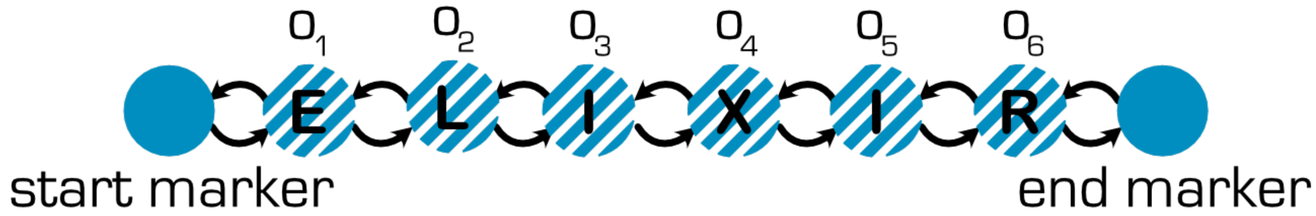


# An Example CRDT: YATA

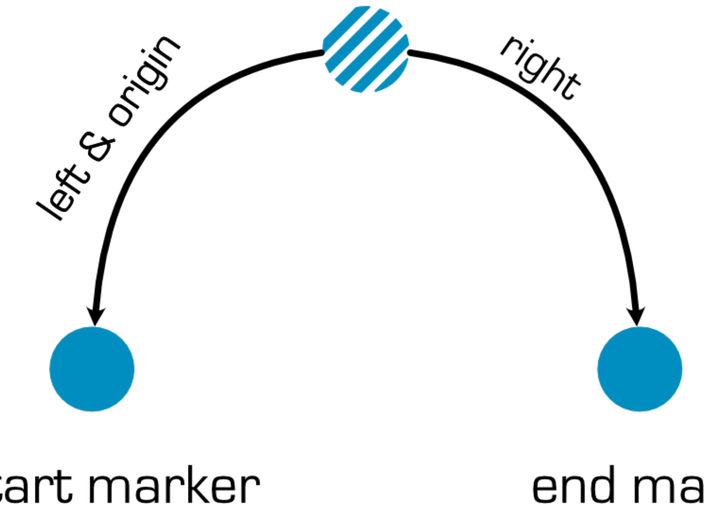
From: Near Real-Time Peer-to-Peer Shared Editing on Extensible Data Types

Article link

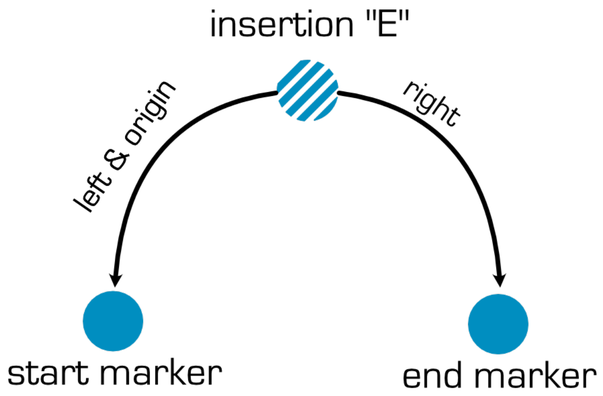
# YATA: It's Linked List

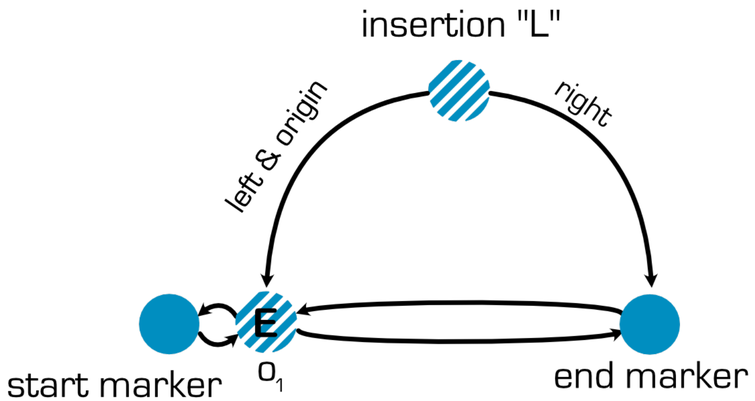


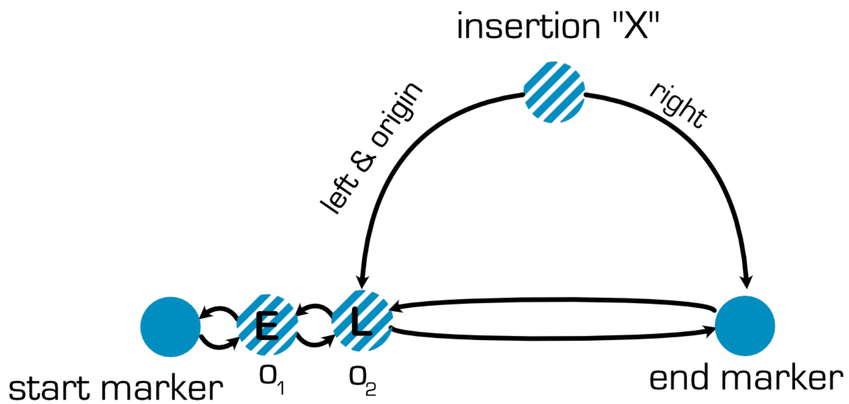

insertion(id, origin, left, right, is_deleted, content)



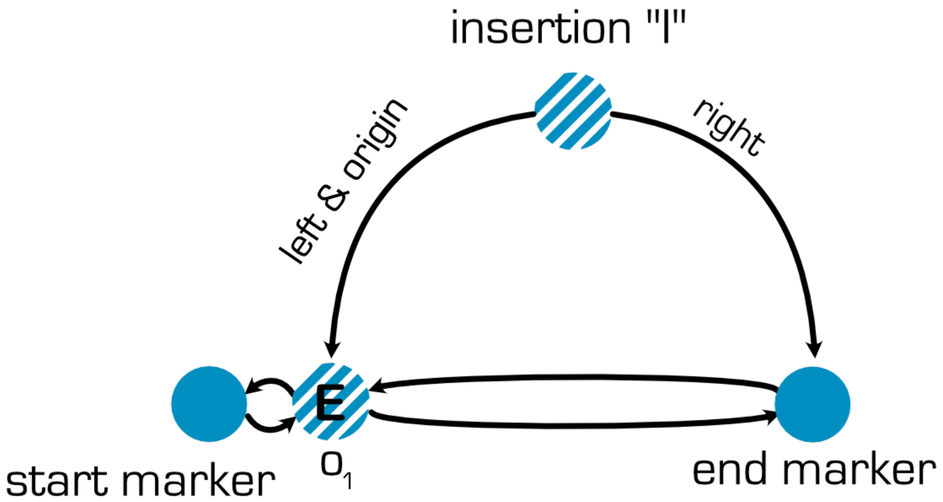
# User 1: Inserts E, L, X, I, R



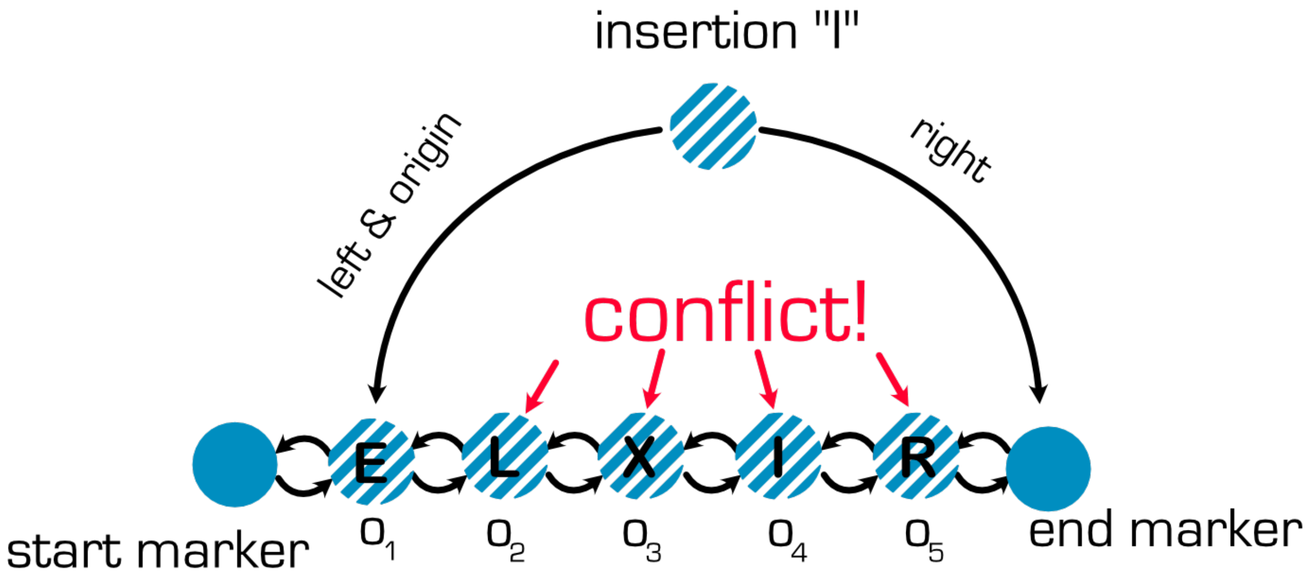




# User 2: Has “E” inserts “I”

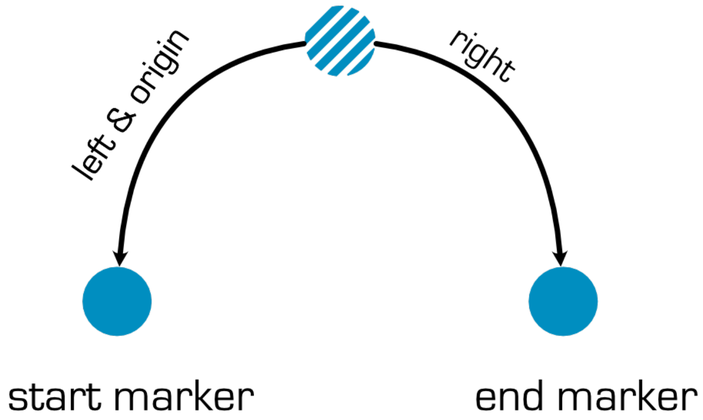


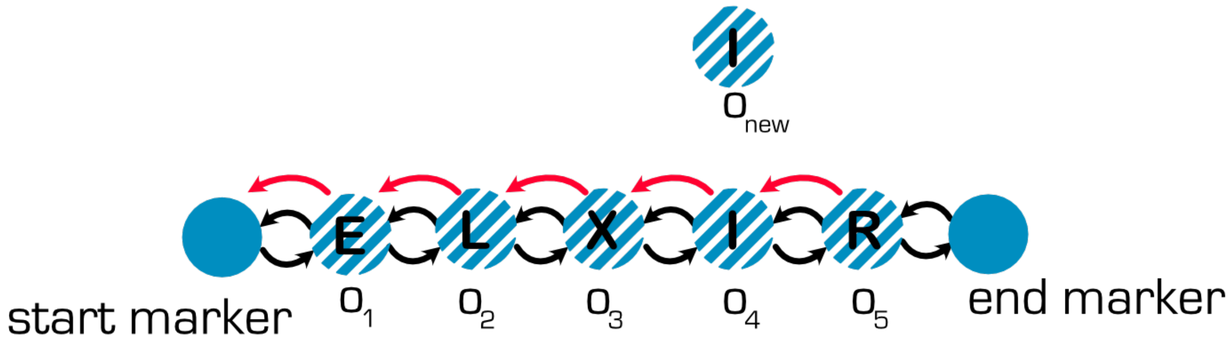
# Conflict


insertion "I"
left & origin
right
conflict!
E
L
X
I
R
start marker
end marker
o1
o2
o3
o4
o5

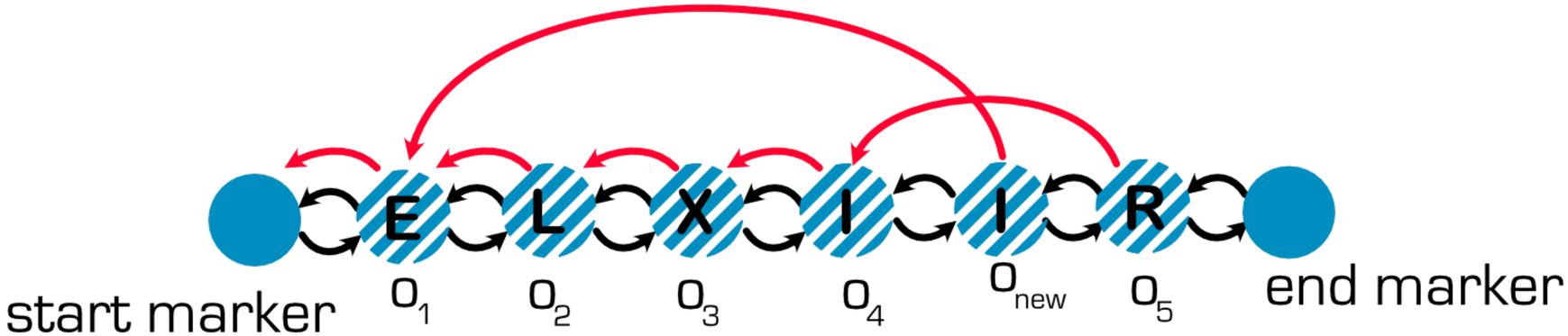
# The Origin Links

insertion(id, origin, left, right, is_deleted, content)



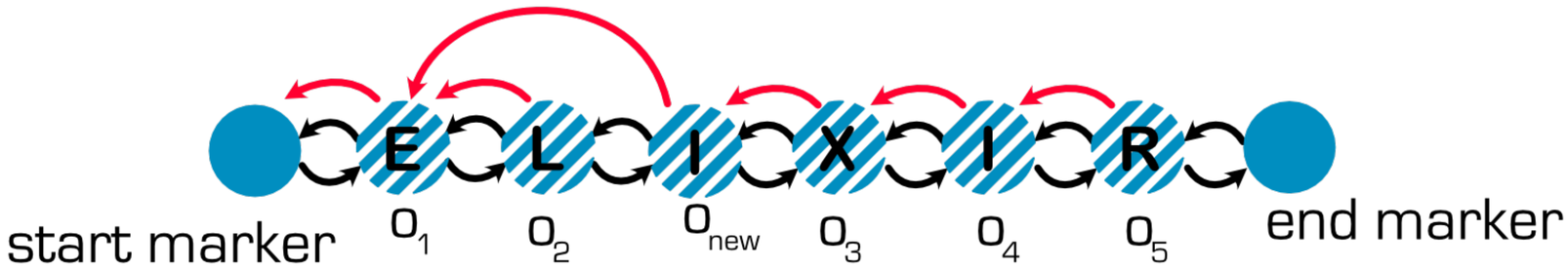



# Conflict Resolution Rule 1: No crossing origin links



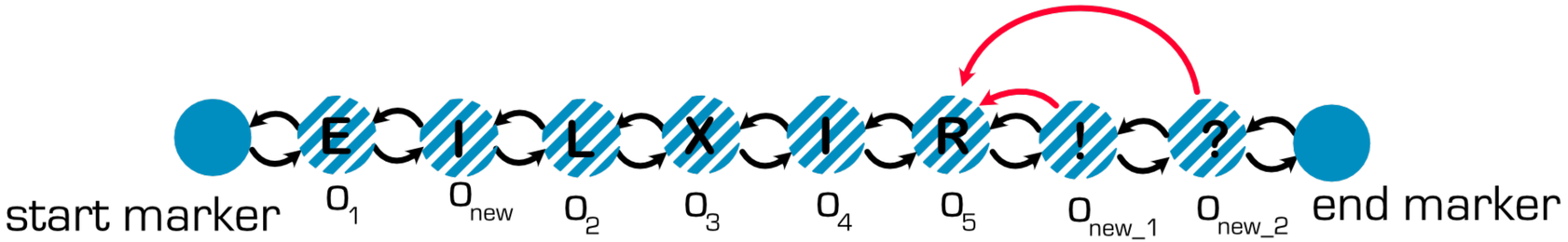
# Conflict Resolution Rule 1: Options

# Conflict Resolution Rule 2: As close to origin as possible

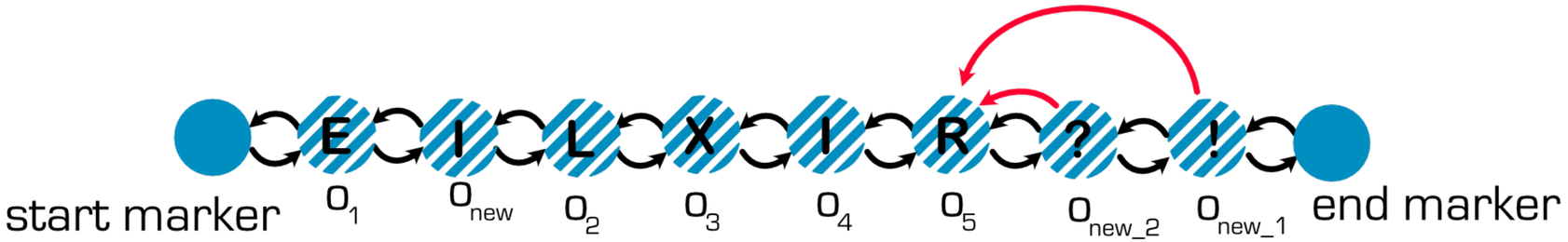


# It’s a tie

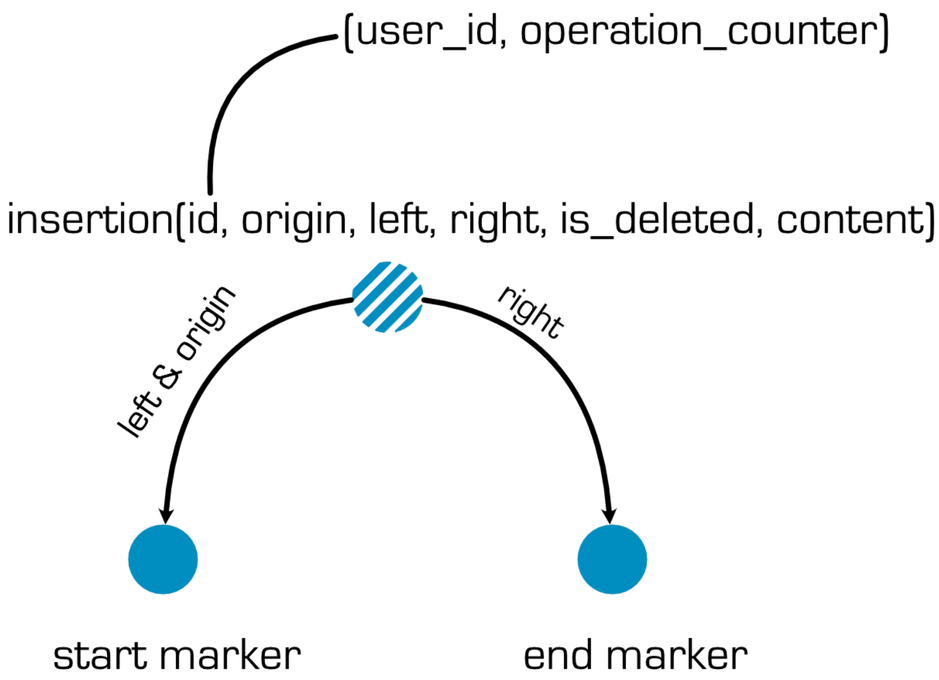
User 1 wants to append “!” and User 2 wants to append “?”



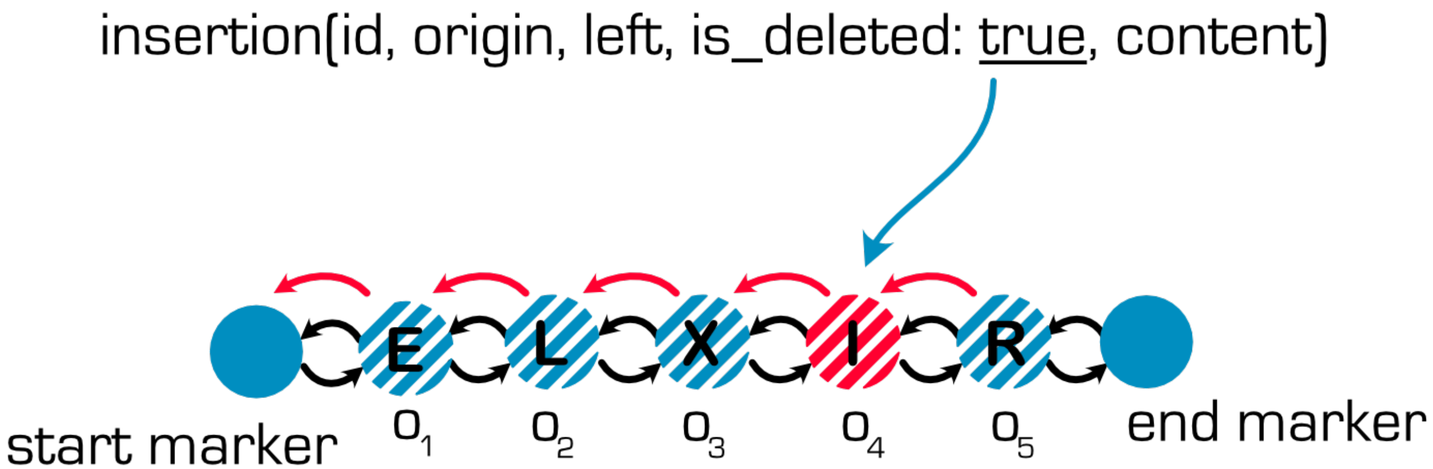

Or



# Conflict Resolution Rule 3: In case of a tie lowest user_id wins

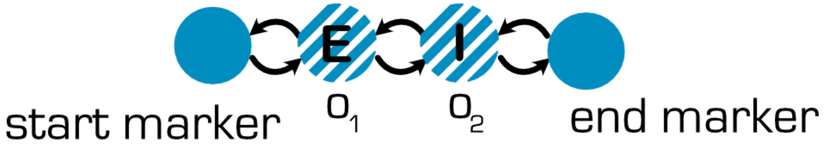


# Tombstone Approach to Deletions

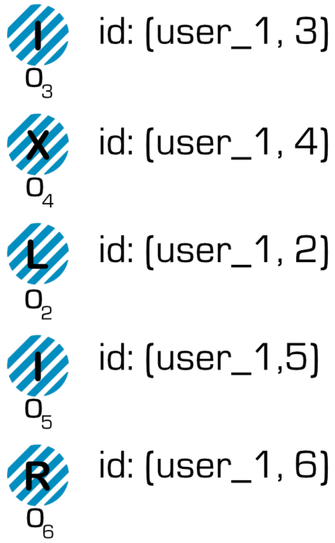


# Why the Operation ID?

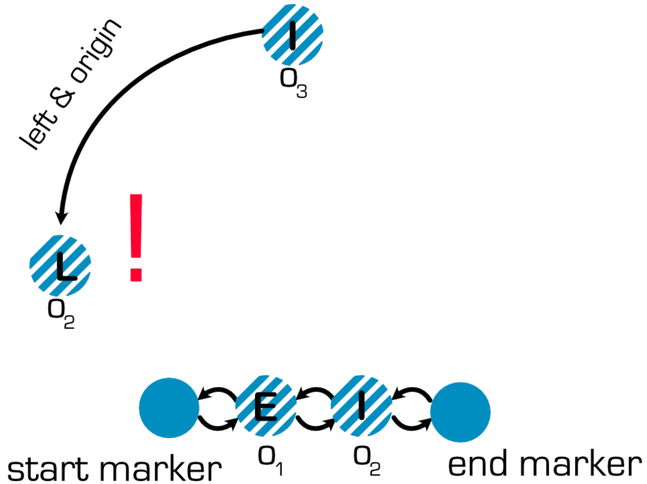
## Current State



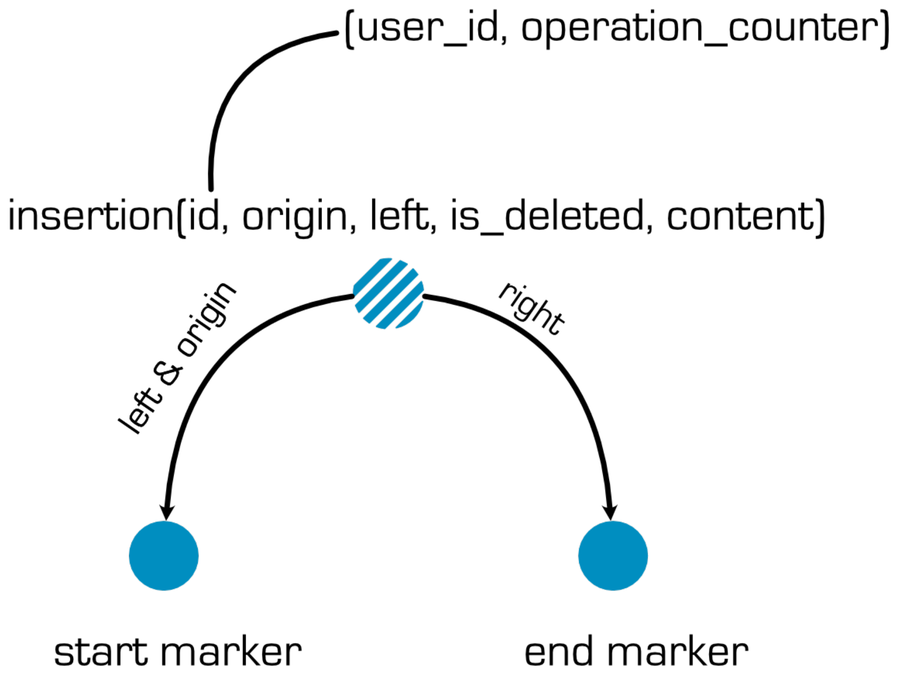

## Incoming operations




## Oh no!



Each client has to keep track of the operation counter for each user



# Generalisation

Maps:

- Keys are Documents
- Values are Documents

XML:

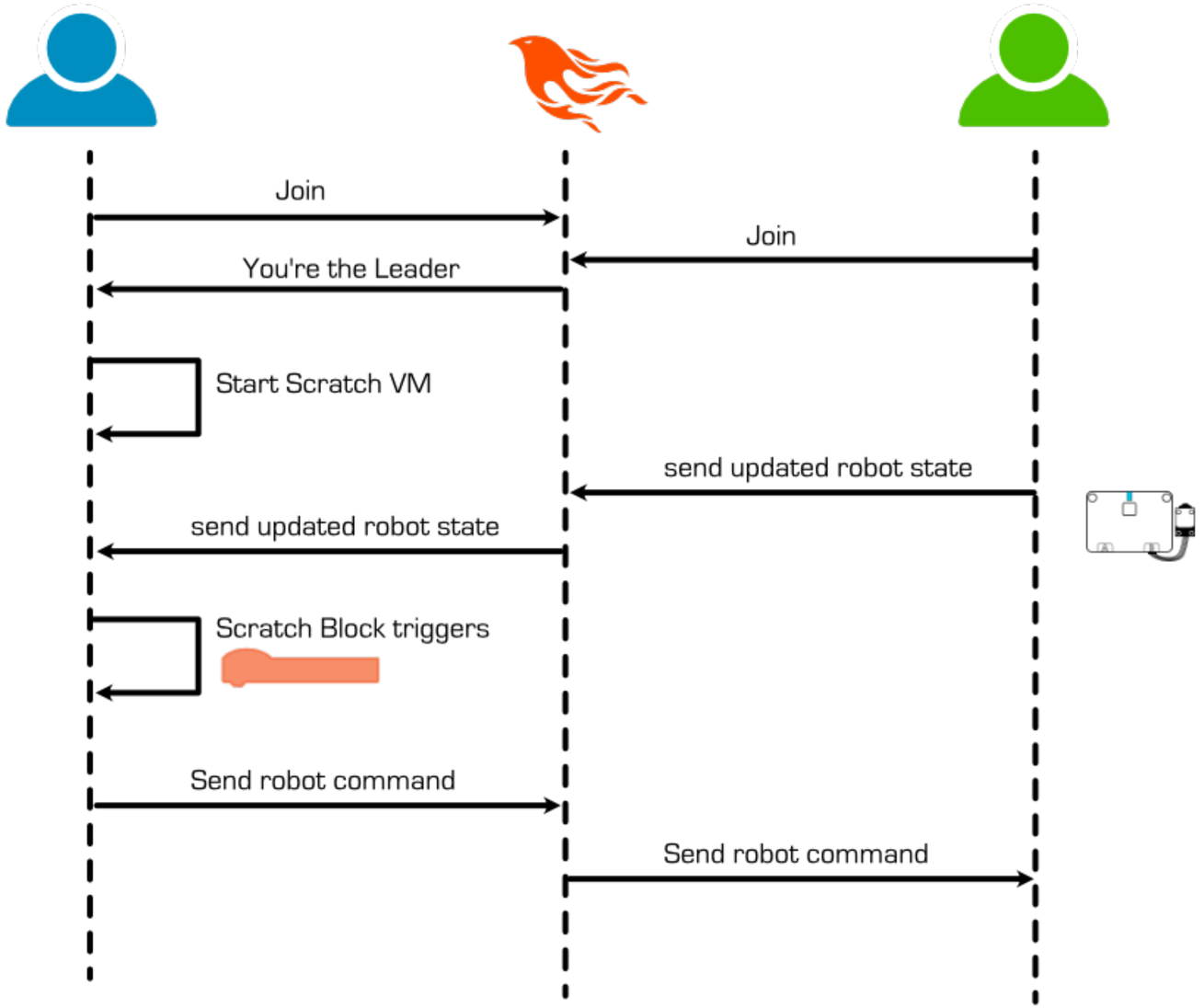
- It’s just a map

# Seems simple enough!

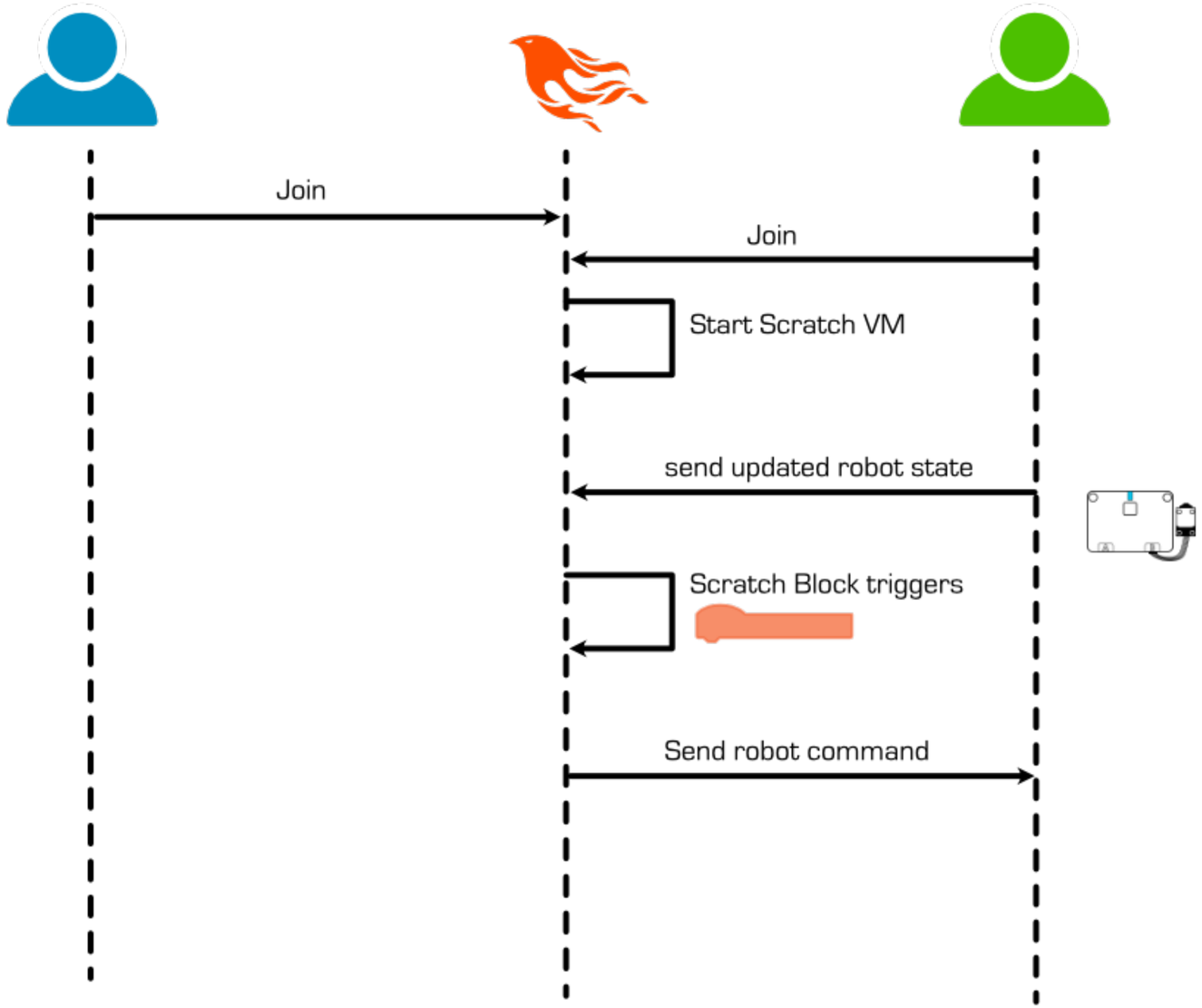
And there is already a JS implementation (y.js)

I'll port y.js to Elixir. Easy!

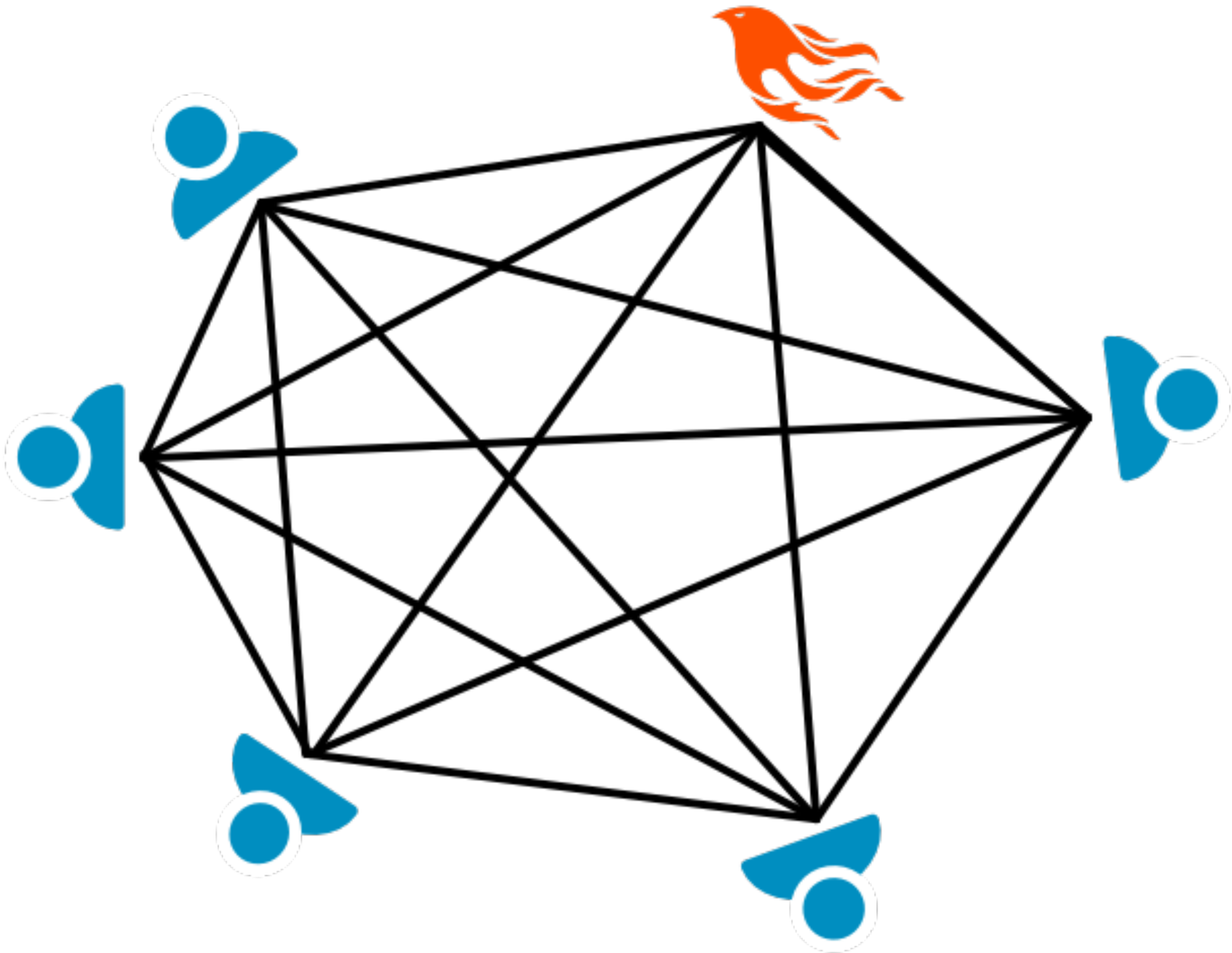
# How the Scratch program runs



# How I'd like the Scratch programs to run



# Just another client



# Not so quick!*

# y.js is ~ 20_000 LOC

*This is hyperbole for dramatic effect: There's are rust implementations you could just use

October 16, 2019

# How Figma's multiplayer technology works

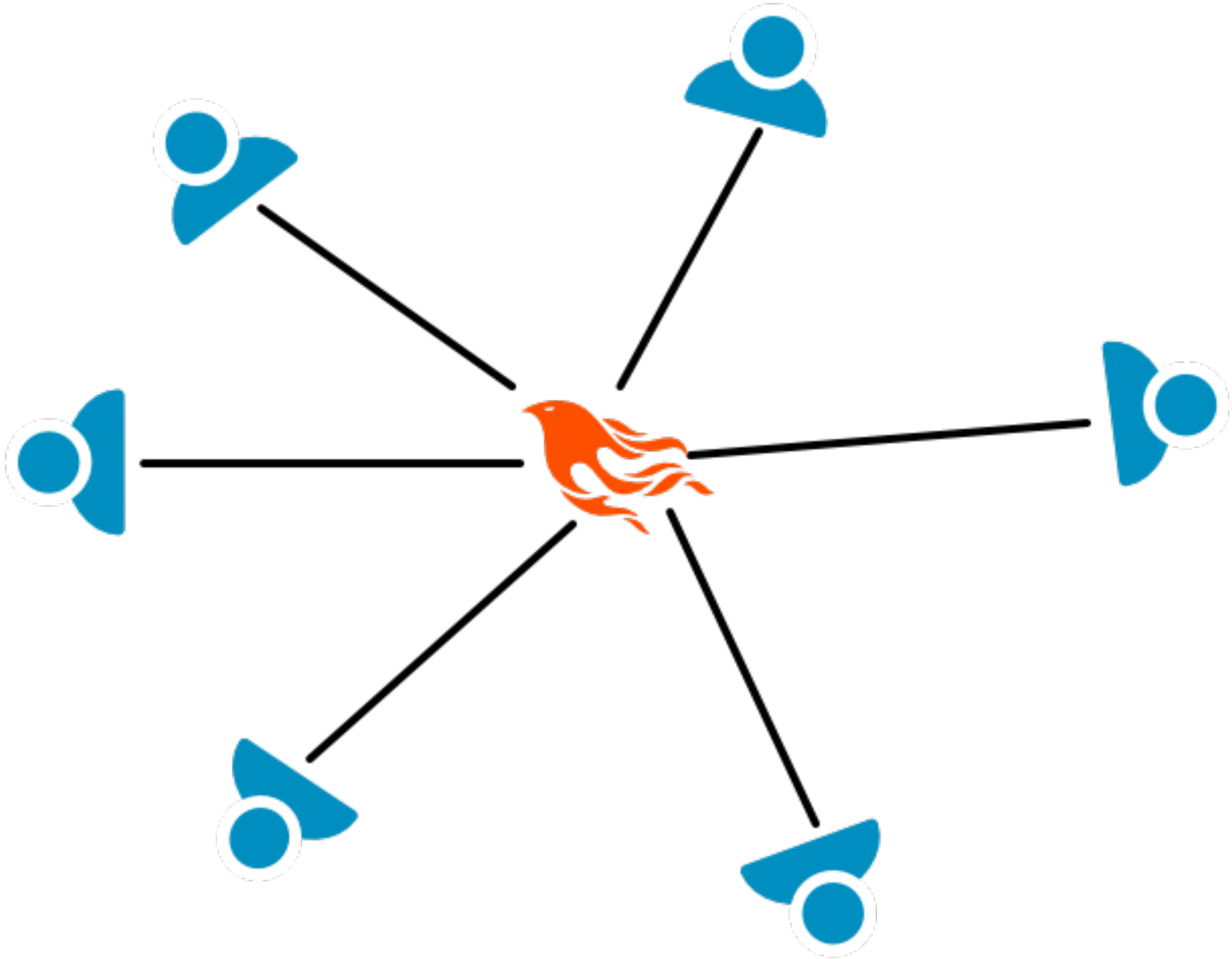
**Evan Wallace** Co-founder, Figma

Inside Figma | Engineering | Behind the scenes | •••

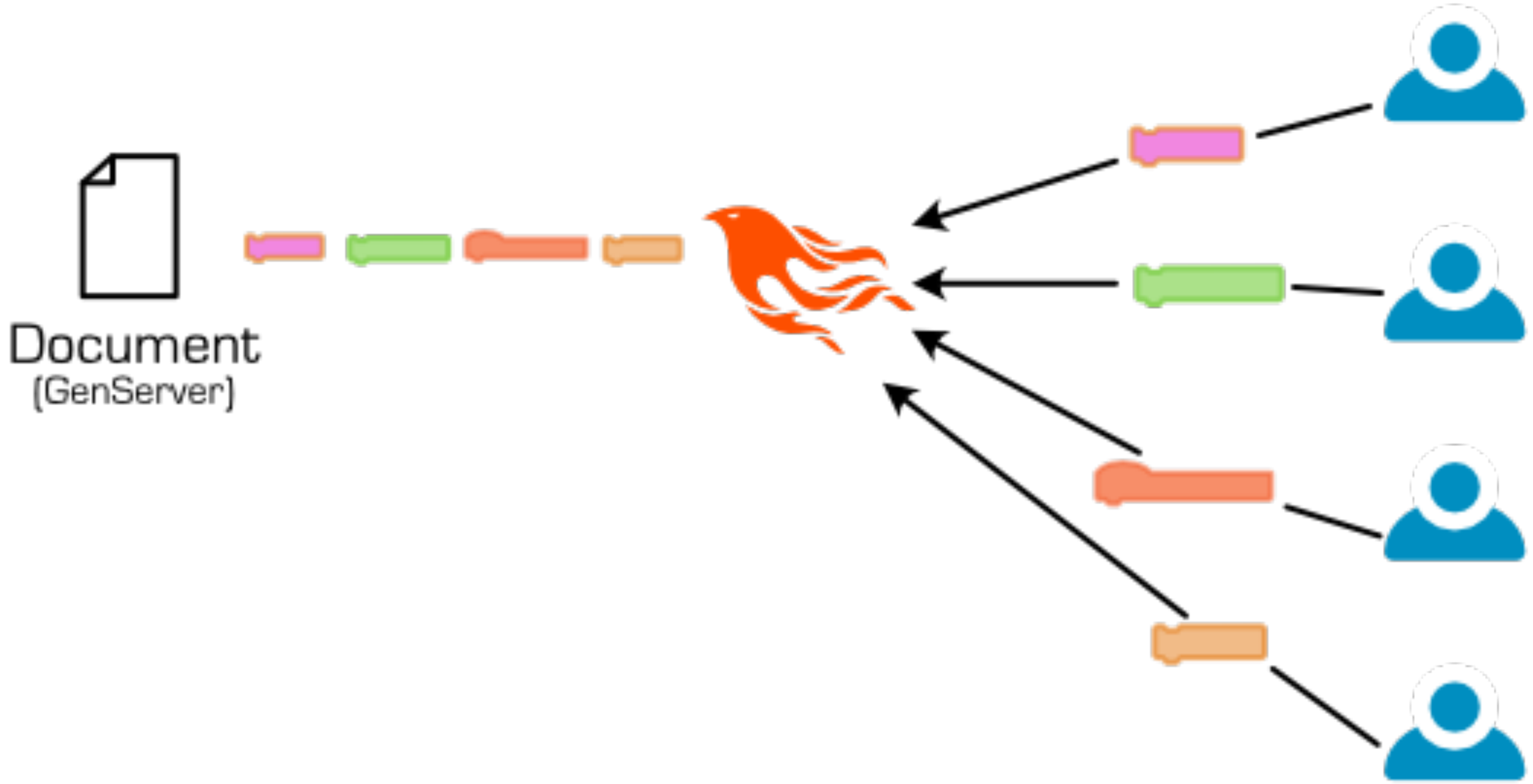
Article link

A peek into the homegrown solution we built as the

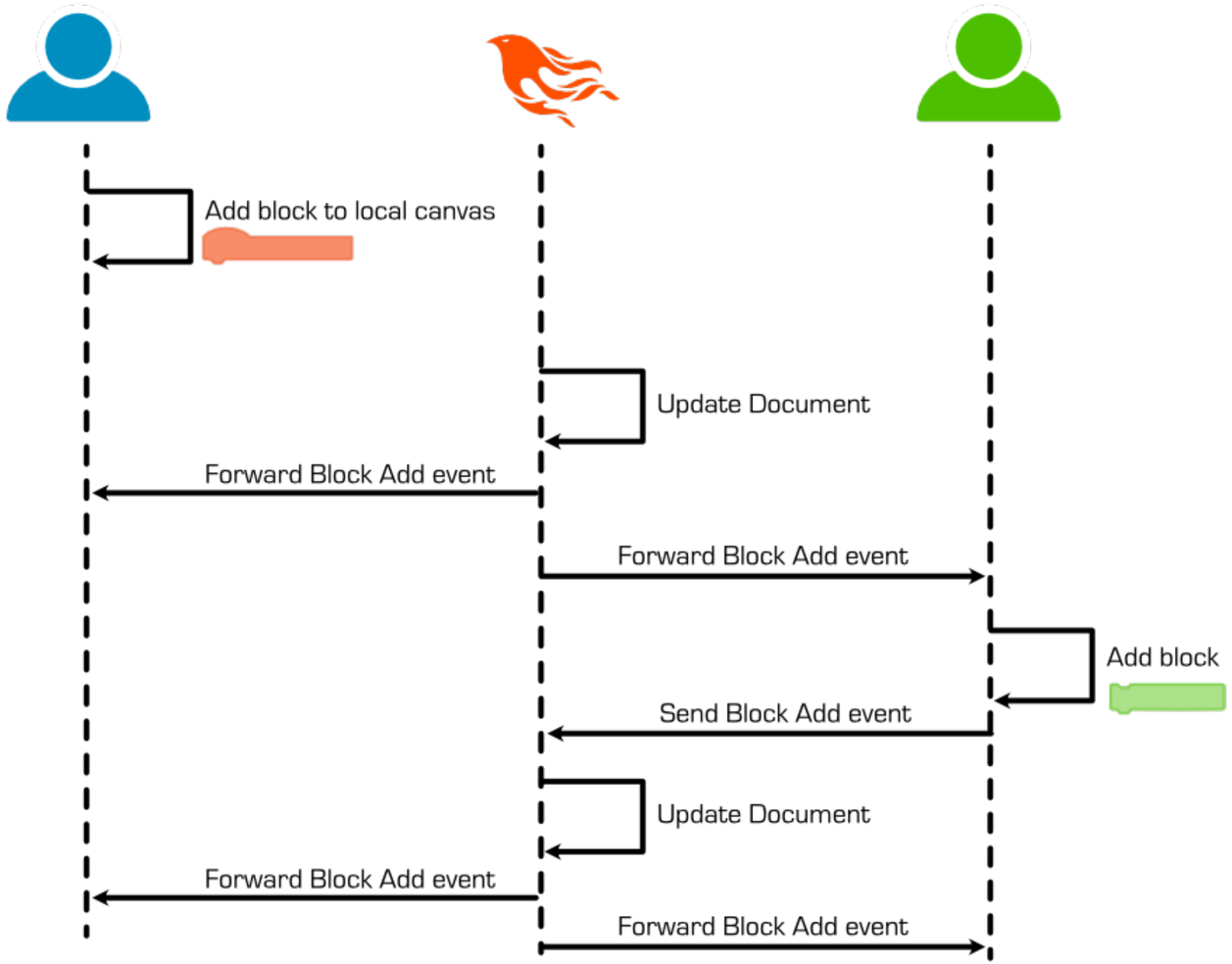
# It’s not a mesh!



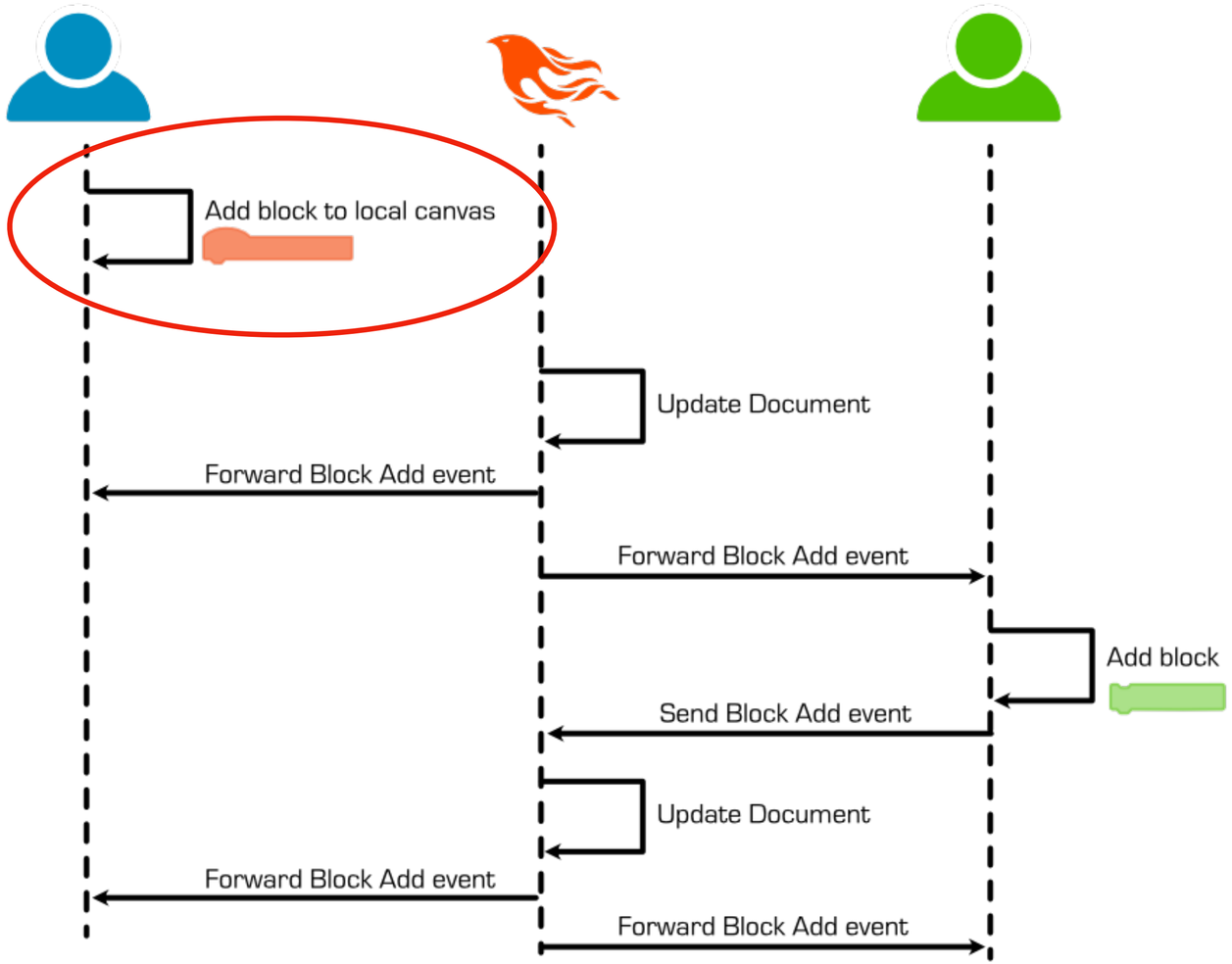
# Ordered Inbox



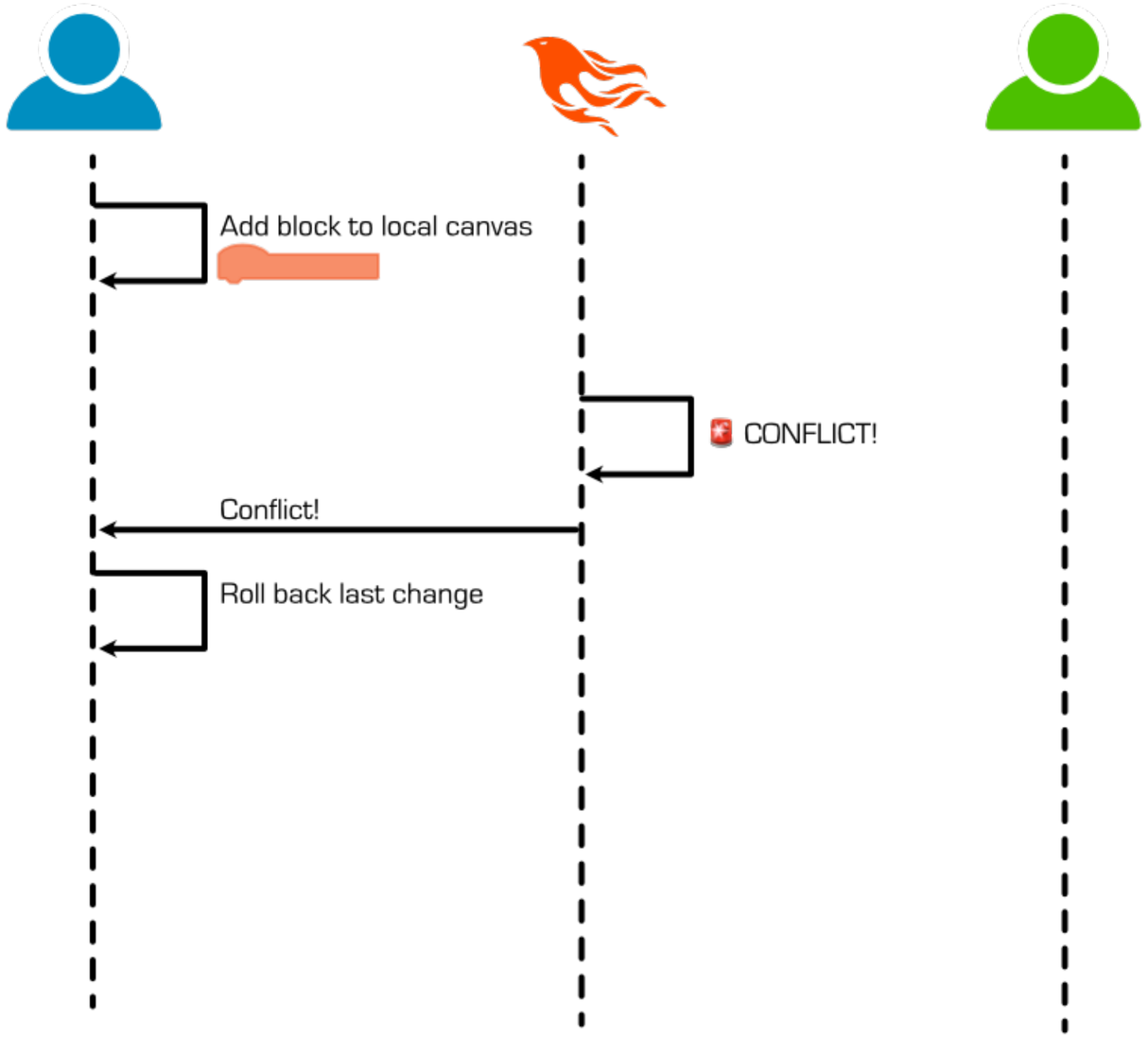
# The New Flow


Add block to local canvas
Update Document
Forward Block Add event
Forward Block Add event
Add block
Send Block Add event
Update Document
Forward Block Add event
Forward Block Add event

# The New Flow


Add block to local canvas
Update Document
Forward Block Add event
Forward Block Add event
Add block
Send Block Add event
Update Document
Forward Block Add event
Forward Block Add event

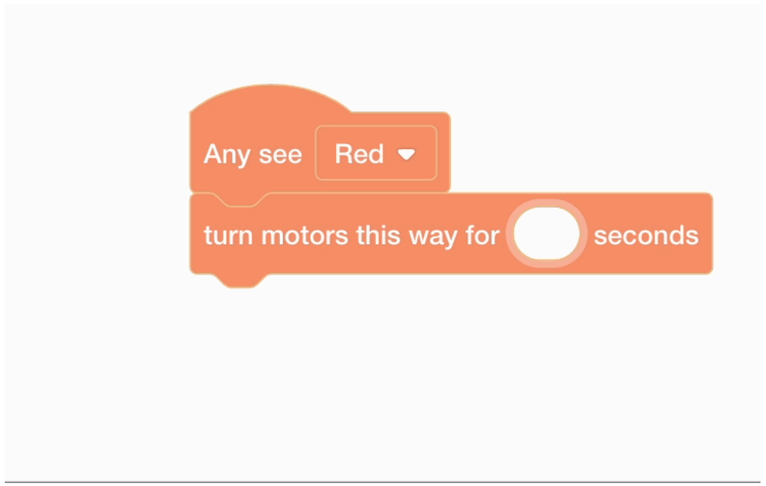
# The Conflict Flow

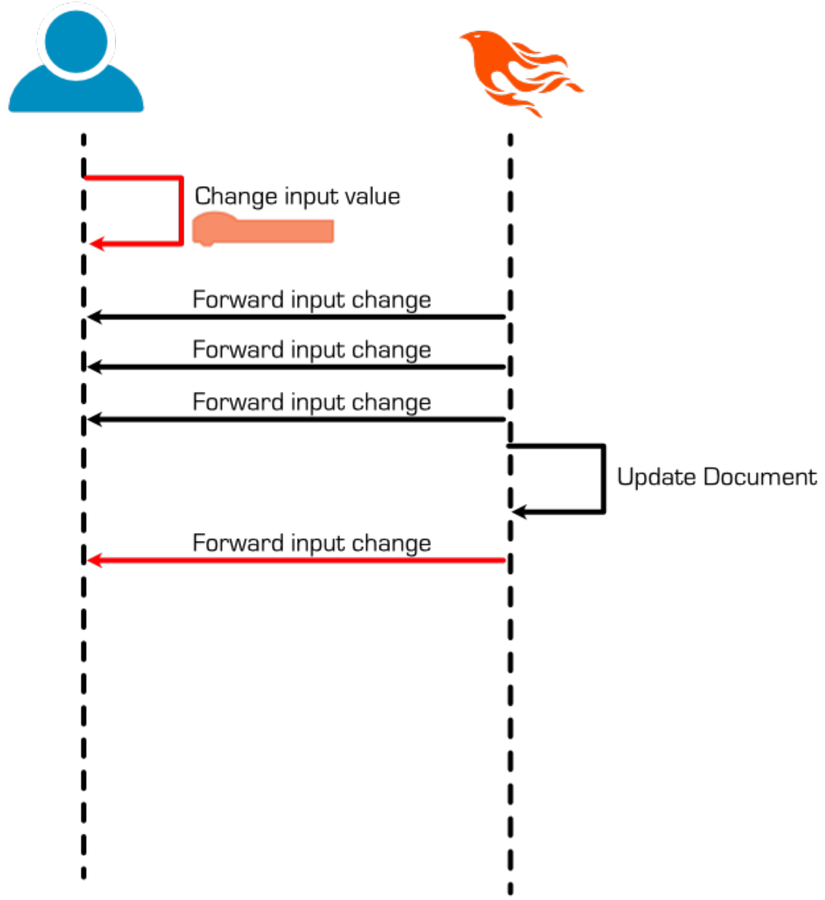


# What Can’t Conflict? Last Edit Wins

- Creating blocks
- Deleting blocks
- Moving blocks
- Changing a input value to a block

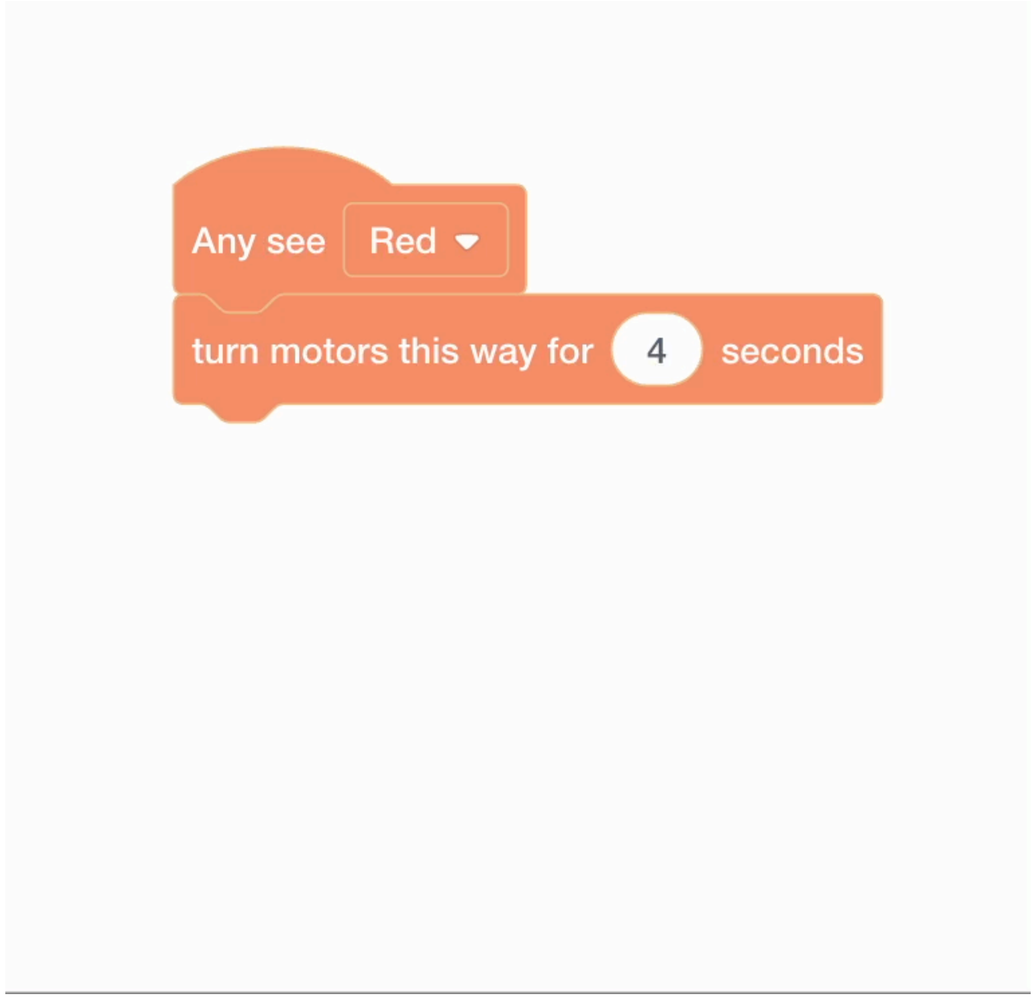
# Flickering input values



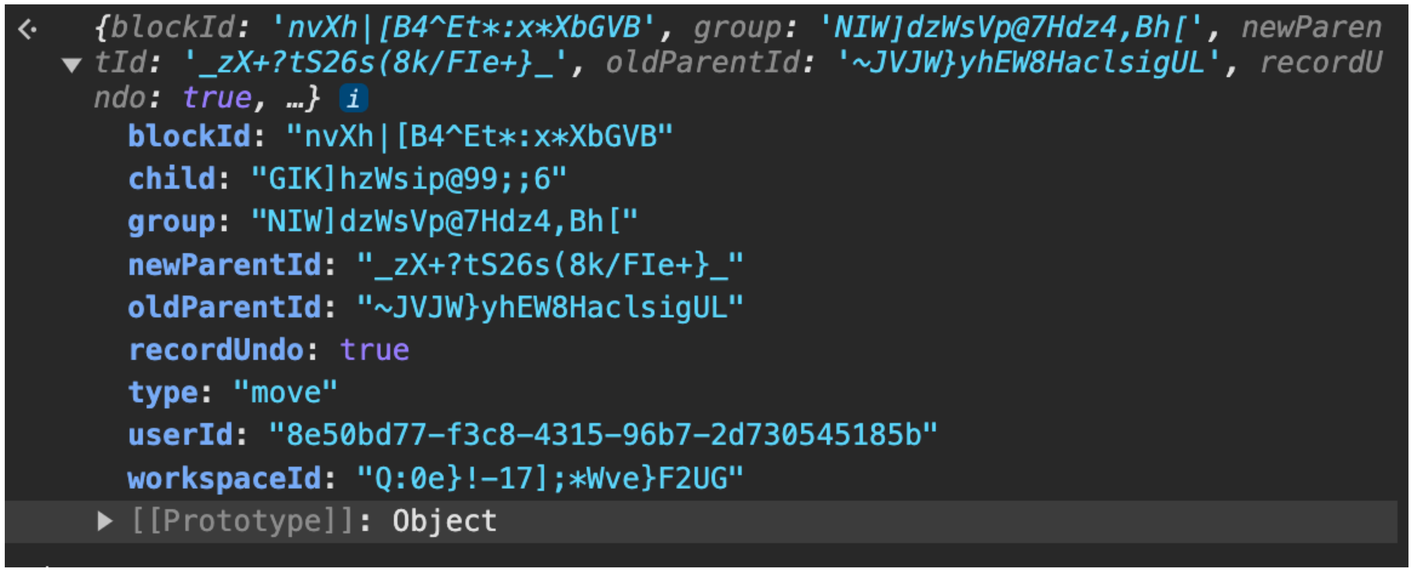



# What Can Conflict?

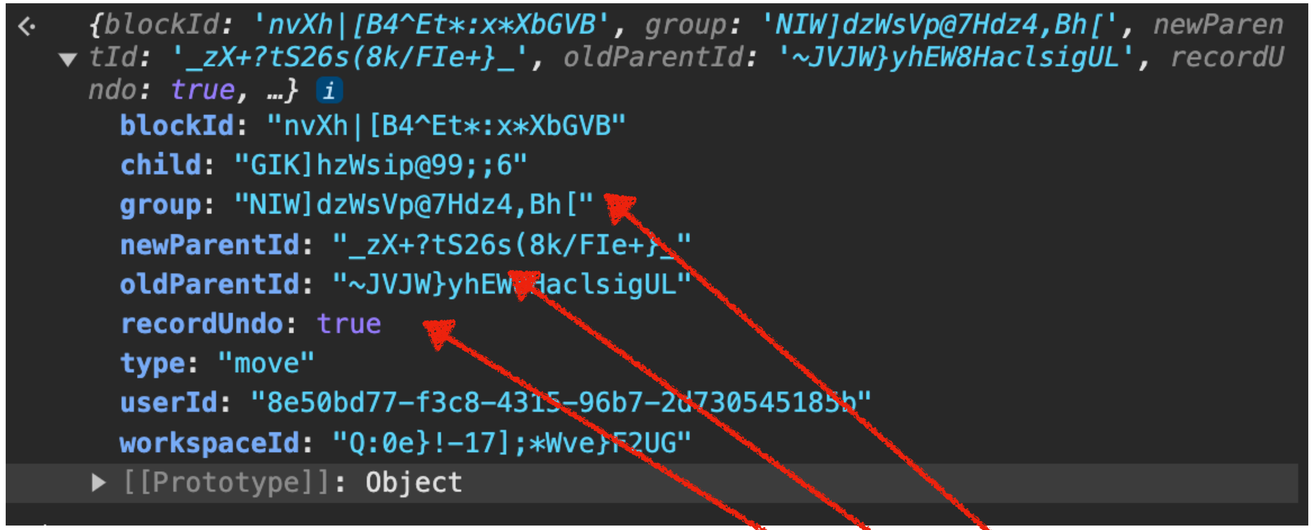
- Reparenting



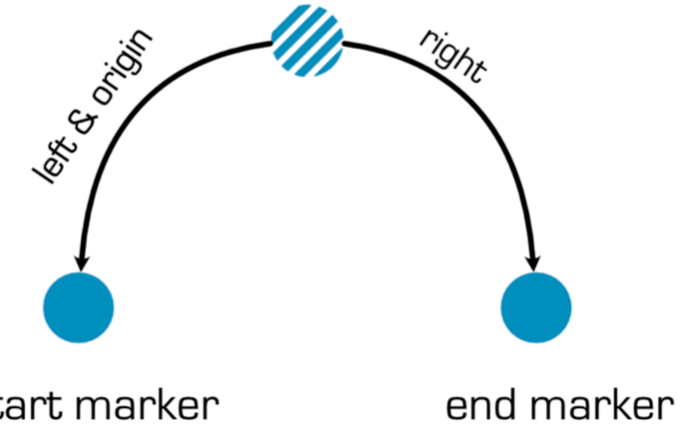
# Reparenting Conflict Resolution



# Reparenting Conflict Resolution



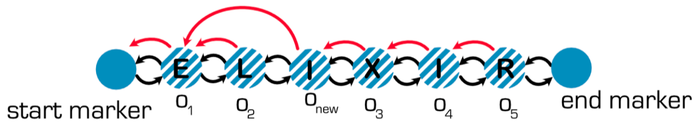

insertion(id, origin, left, right, is_deleted, content)



# Reparenting Conflict Resolution

- Rule 1: Place after new parent
- Rule 2: If new parent is gone, place just before child
- Rule 3: Child gone? Create new stack

Conflict Resolution Rule 2: As close to origin as possible



# And there you have it - faking local first conflict free editing

# Pros and Cons

Pros:

- Simpler algorithm
- Much easier testing
- Tailored to your app

Cons:

- Not offline edit capable
- You have to do the work
- Multi-node support not there