Grokking #9

Building an offline & real-time editing service with Couchbase

Oliver N.

Software Engineer

In this talk

You'll walk away with

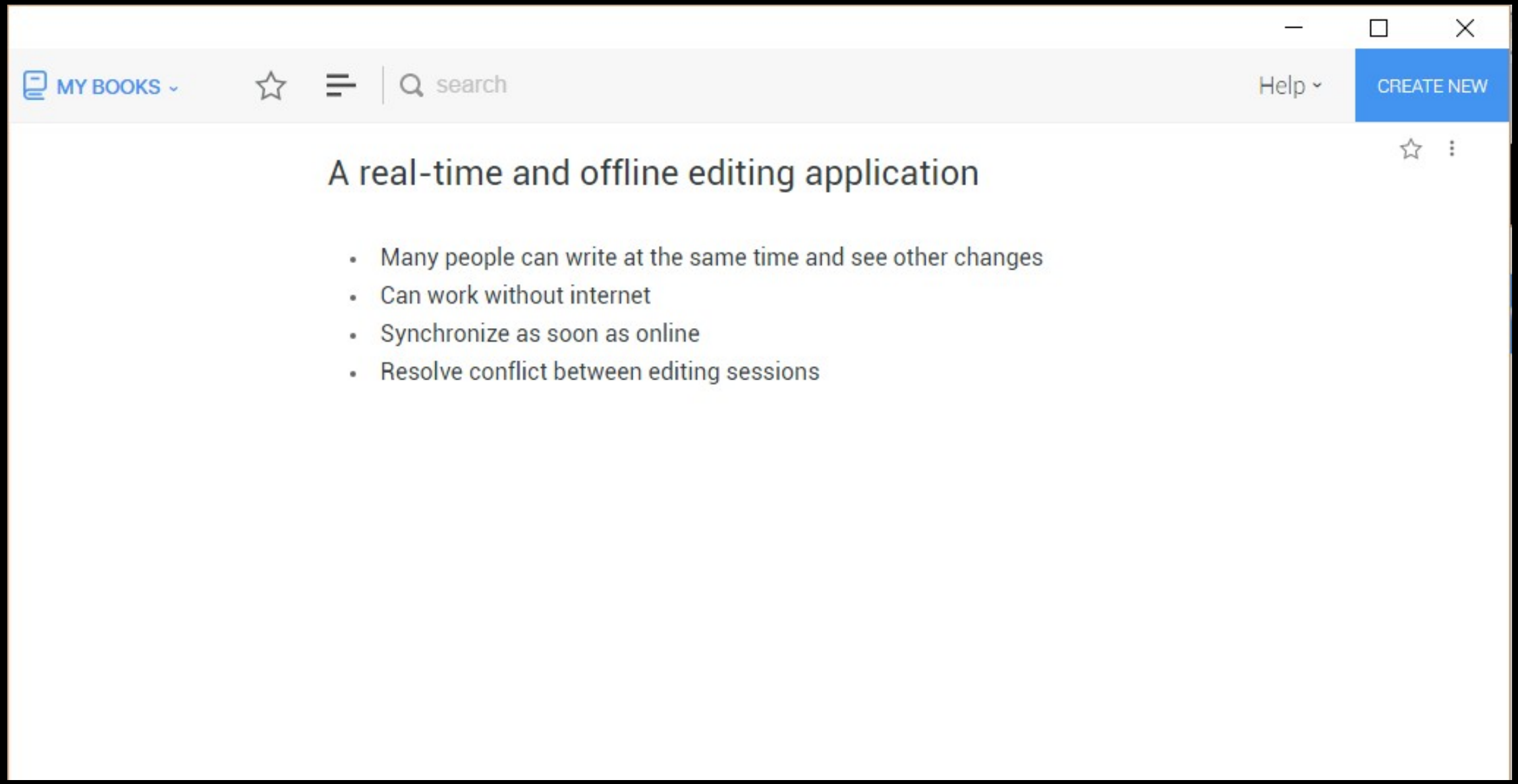
- Our approach for an unknown architecture
- How do we design architecture for real-time & offline
- Couchbase and other real-time solutions

Talk structure

1. Problems of a real-time & offline editing service
2. Evaluate real-time frameworks and databases
3. First prototype with Firebase
4. Second version with Couchbase Mobile
5. Time to make our own one!
6. Why Couchbase?

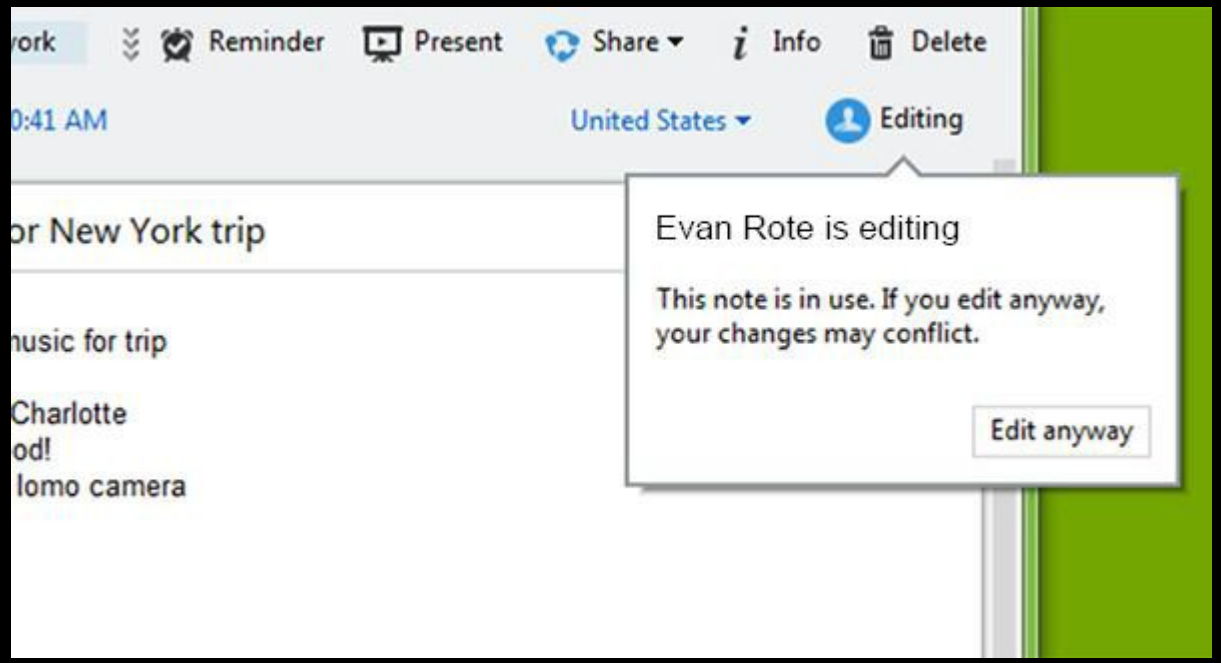
A real-time and offline editing service

What is it?

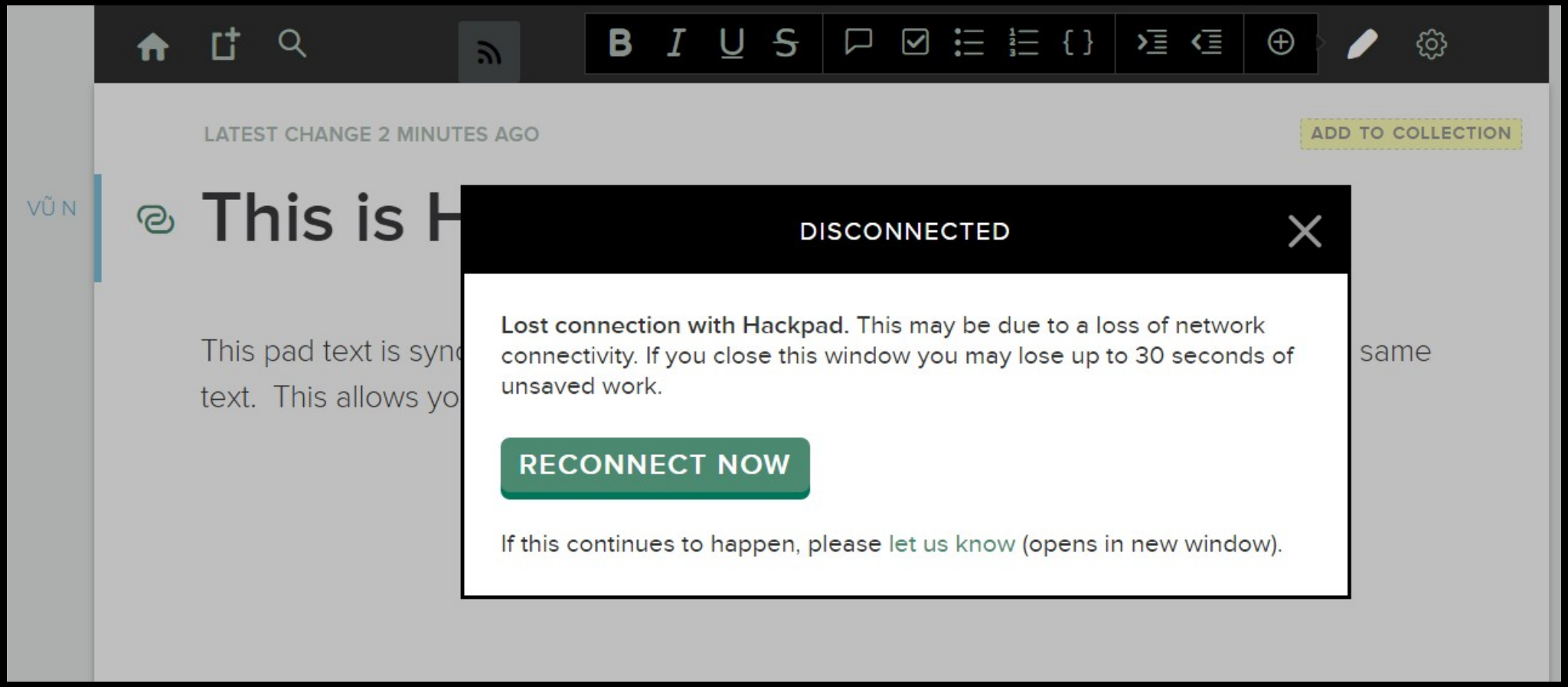

MY BOOKS
search
Help
CREATE NEW
A real-time and offline editing application
Many people can write at the same time and see other changes
Can work without internet
Synchronize as soon as online
Resolve conflict between editing sessions

A real-time and offline editing services

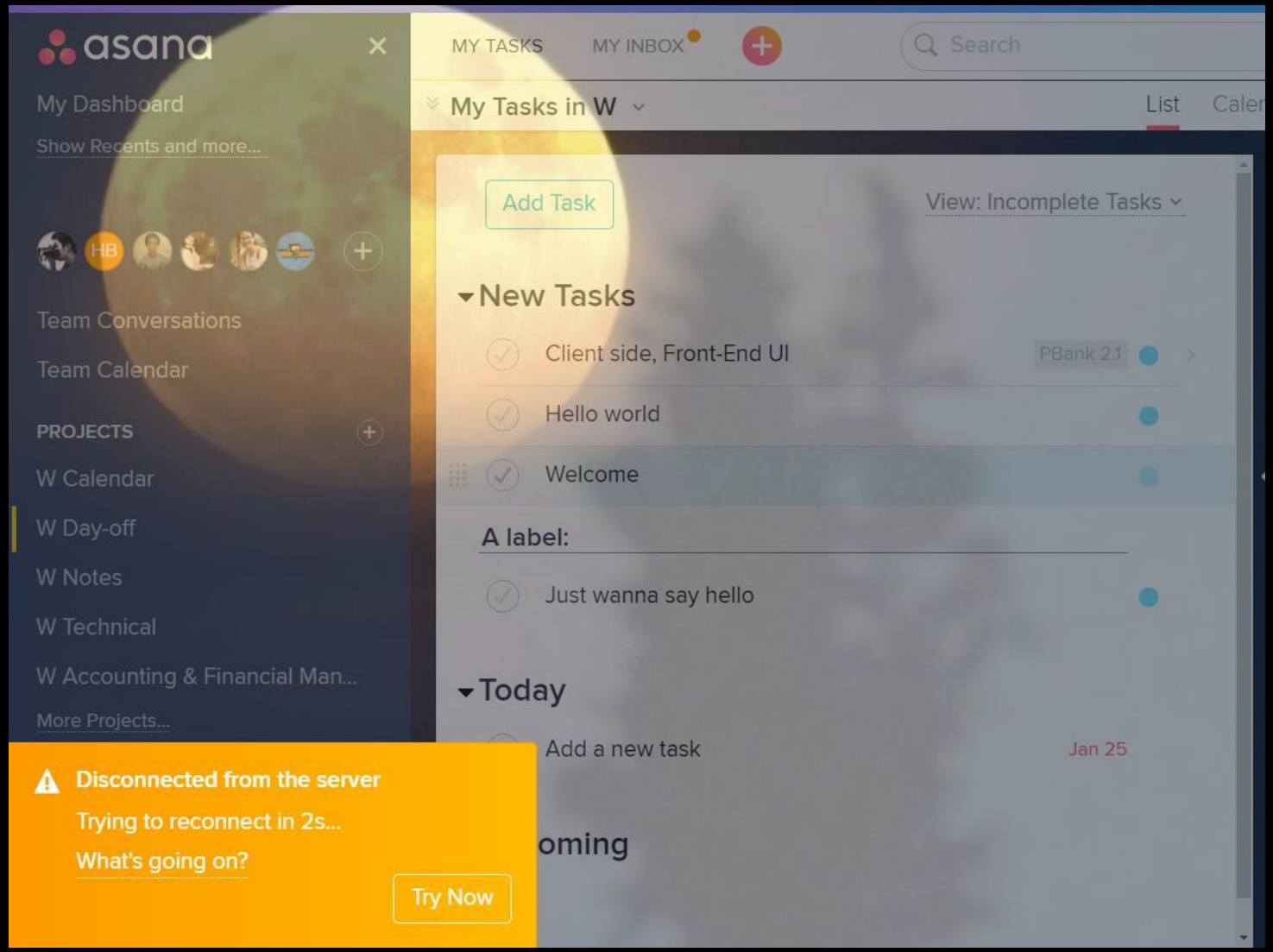
1. Users can edit at the same time
2. Can work without the internet
3. Synchronize as soon as online
4. Resolve conflict between editing sessions



Evernote does not resolve conflict!



Hackpad disables editing while offline!


asana
My Dashboard
Show Recents and more...
Team Conversations
Team Calendar
PROJECTS
W Calendar
W Day-off
W Notes
W Technical
W Accounting & Financial Man...
More Projects...
MY TASKS
MY INBOX
Search
My Tasks in W
List
Add Task
View: Incomplete Tasks
New Tasks
Client side, Front-End UI
PBank 2.1
Hello world
Welcome
A label:
Just wanna say hello
Today
Add a new task
Jan 25
oming
Disconnected from the server
Trying to reconnect in 2s...
What's going on?
Try Now

And Asana!

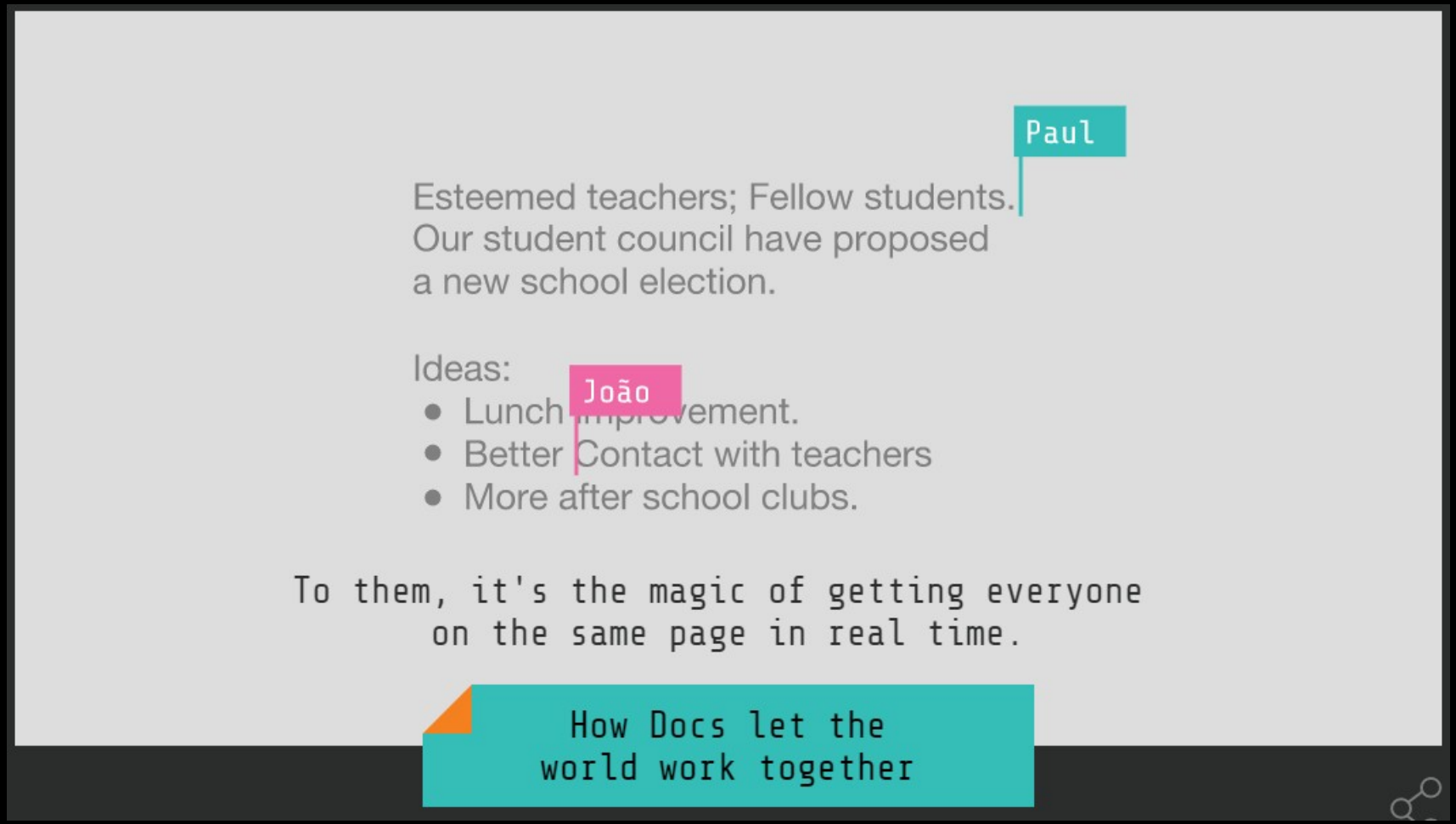
First problem:

How to handle conflict?

A real-time and offline editing service

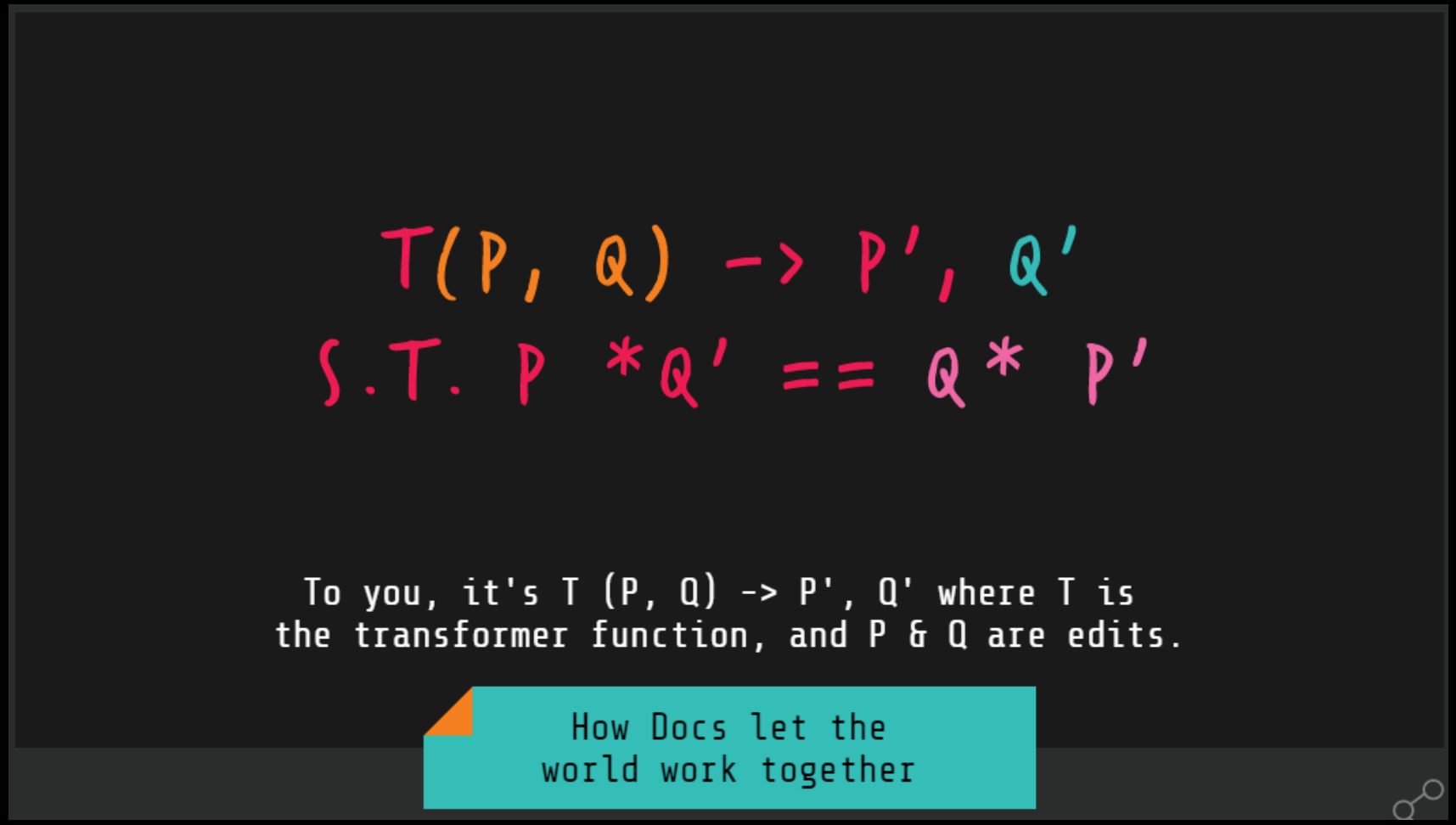
Google Docs

Google Docs



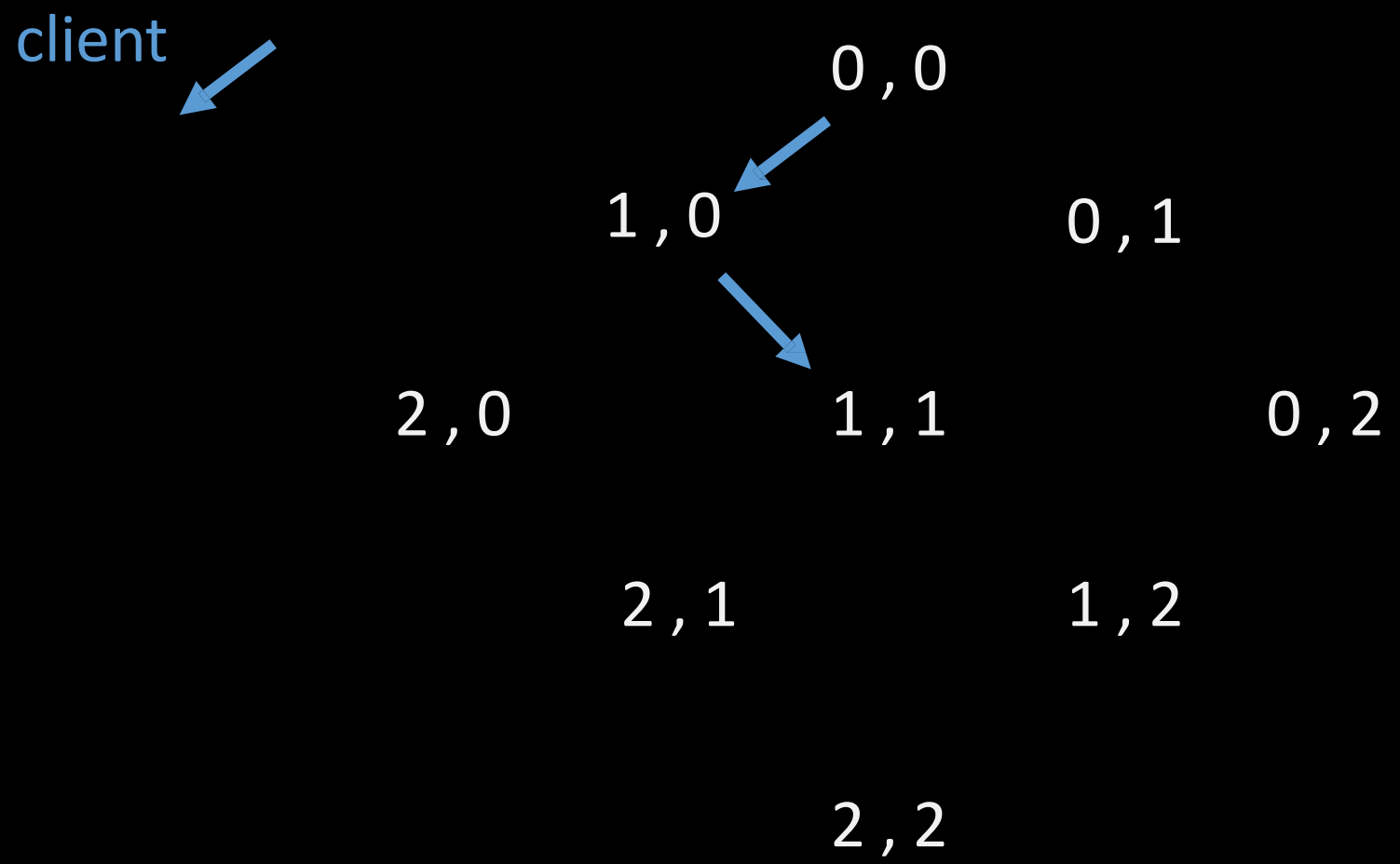
http://bar.foo

Google Docs

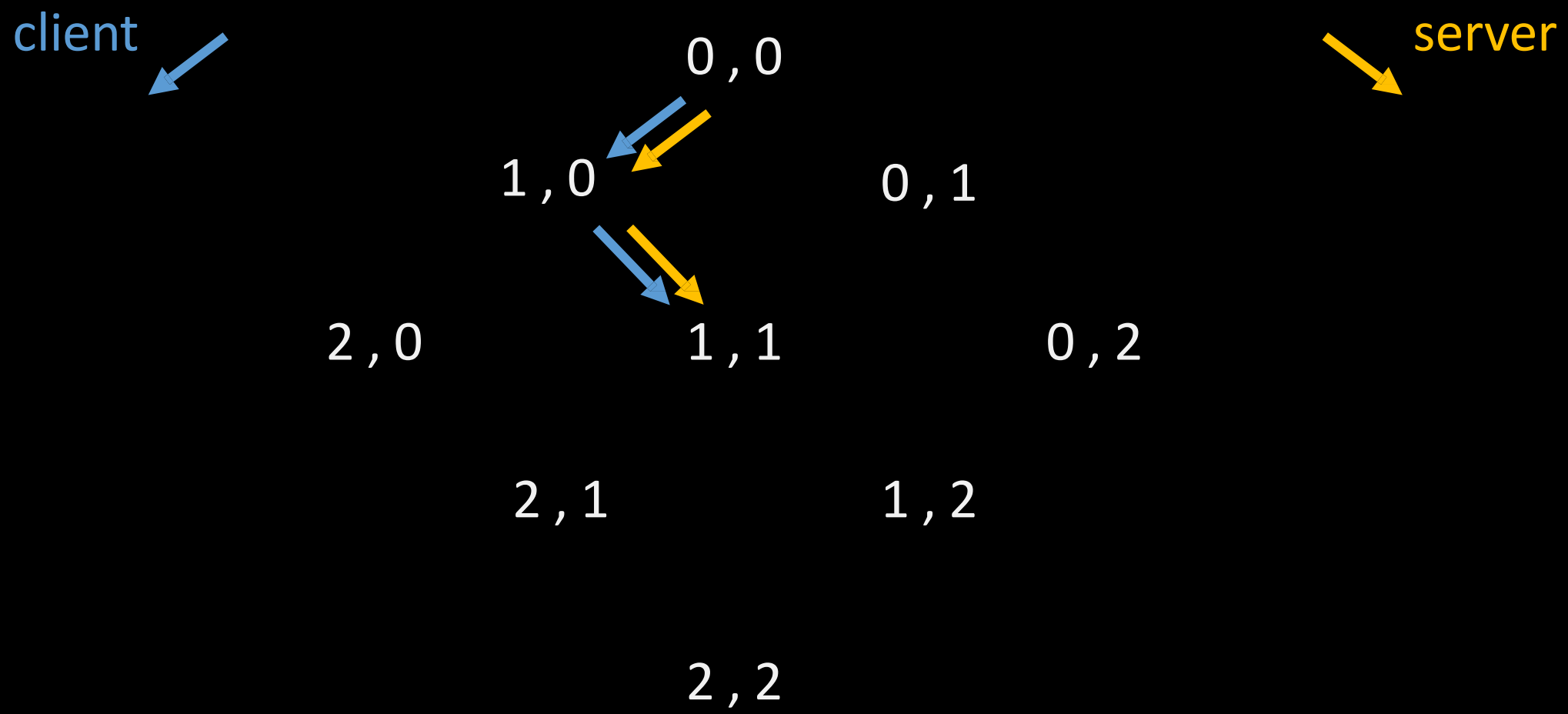


http://bar.foo

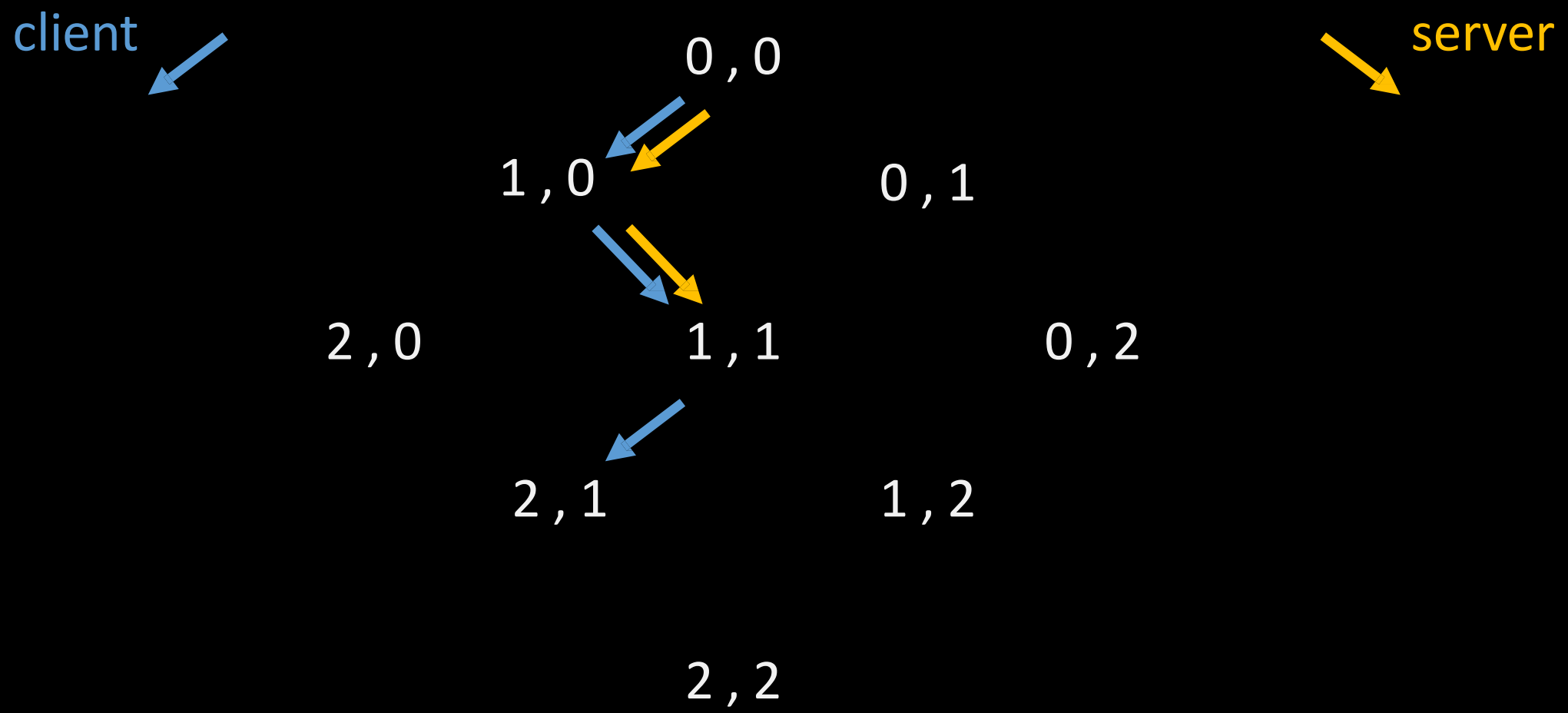
Operational transformation



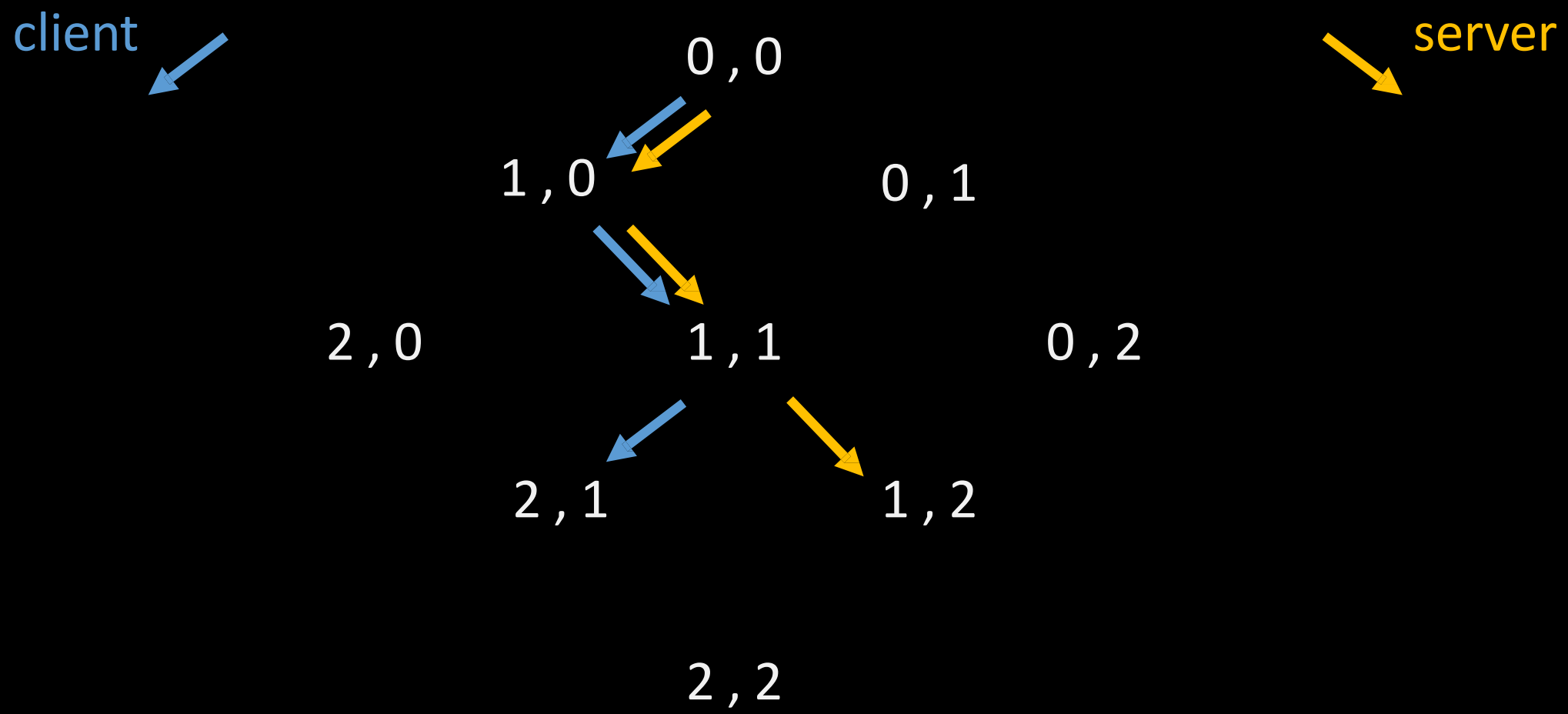
Operational transformation



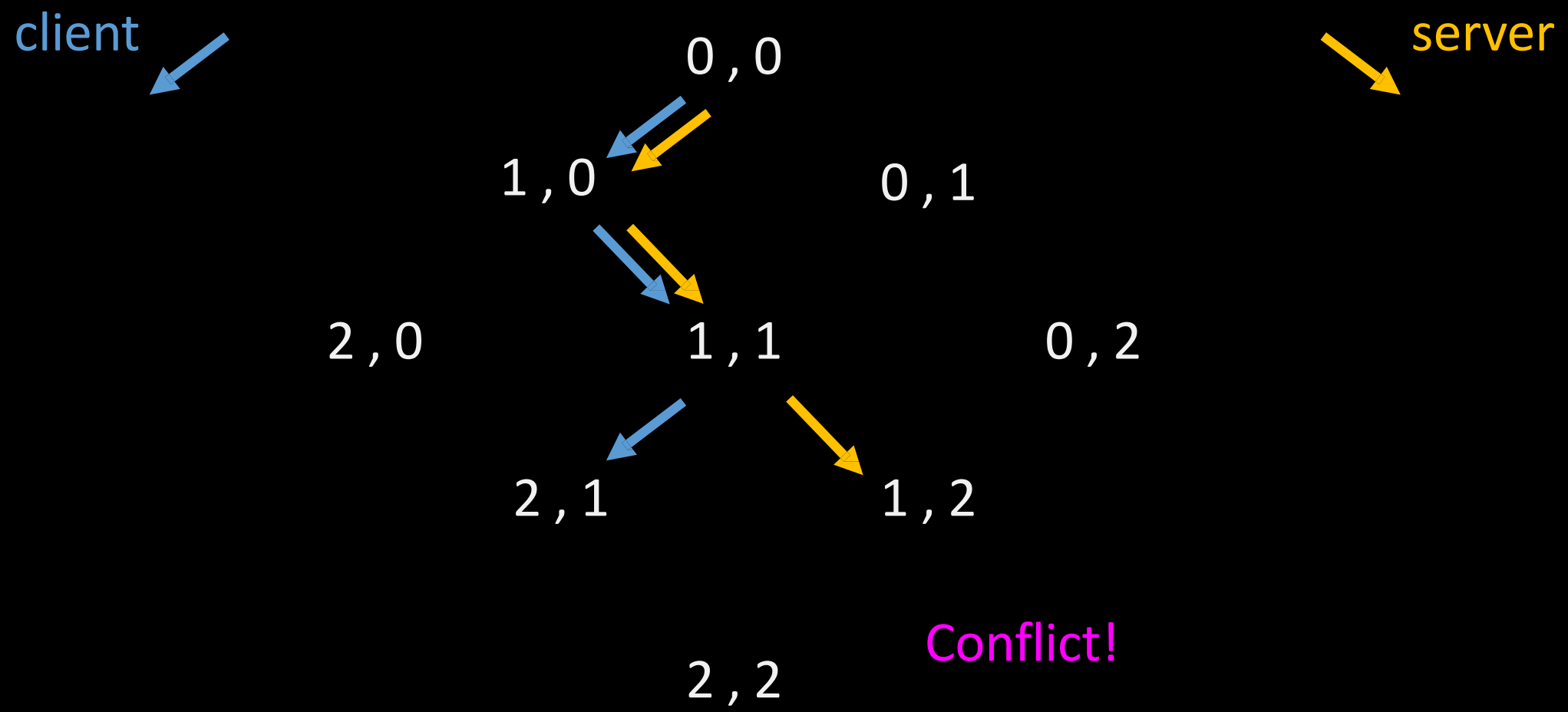
Operational transformation



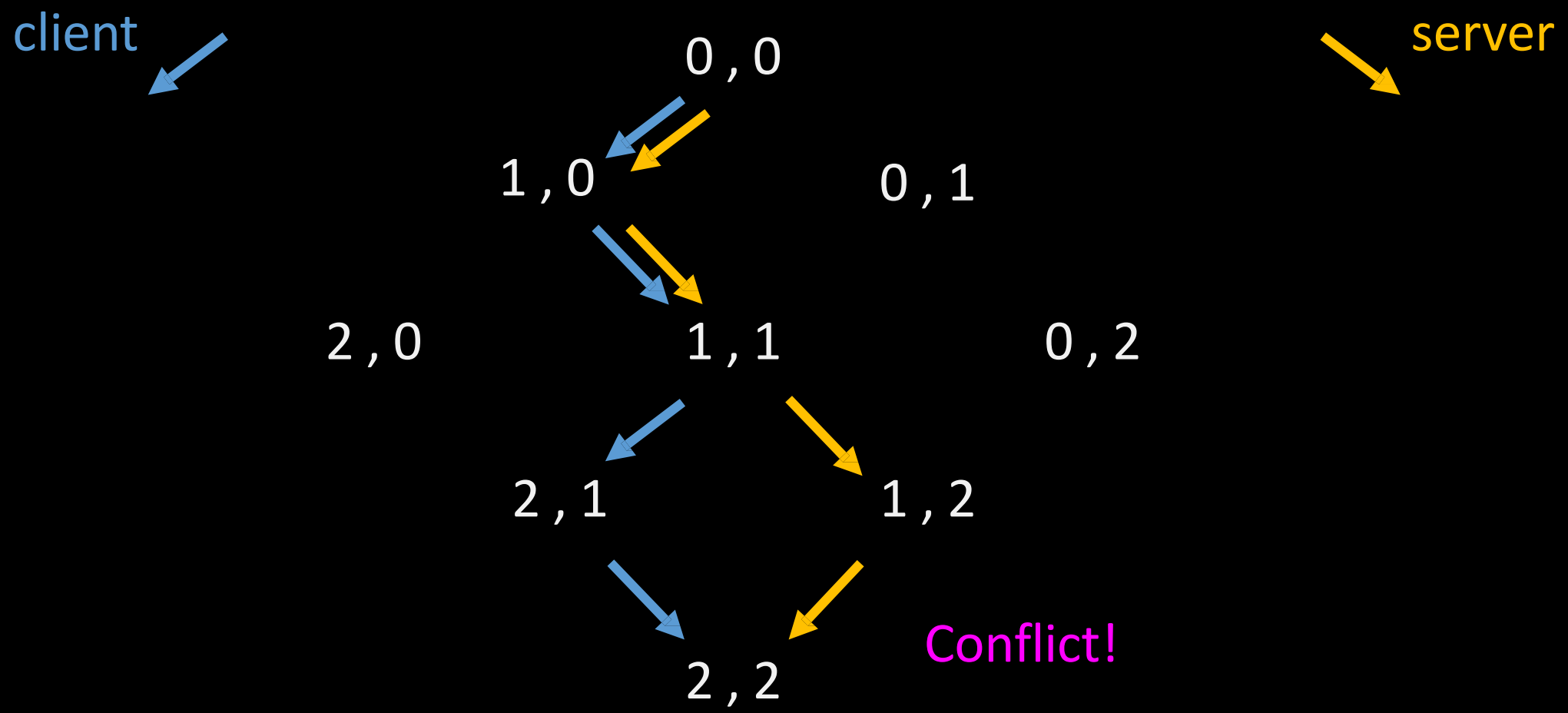
Operational transformation



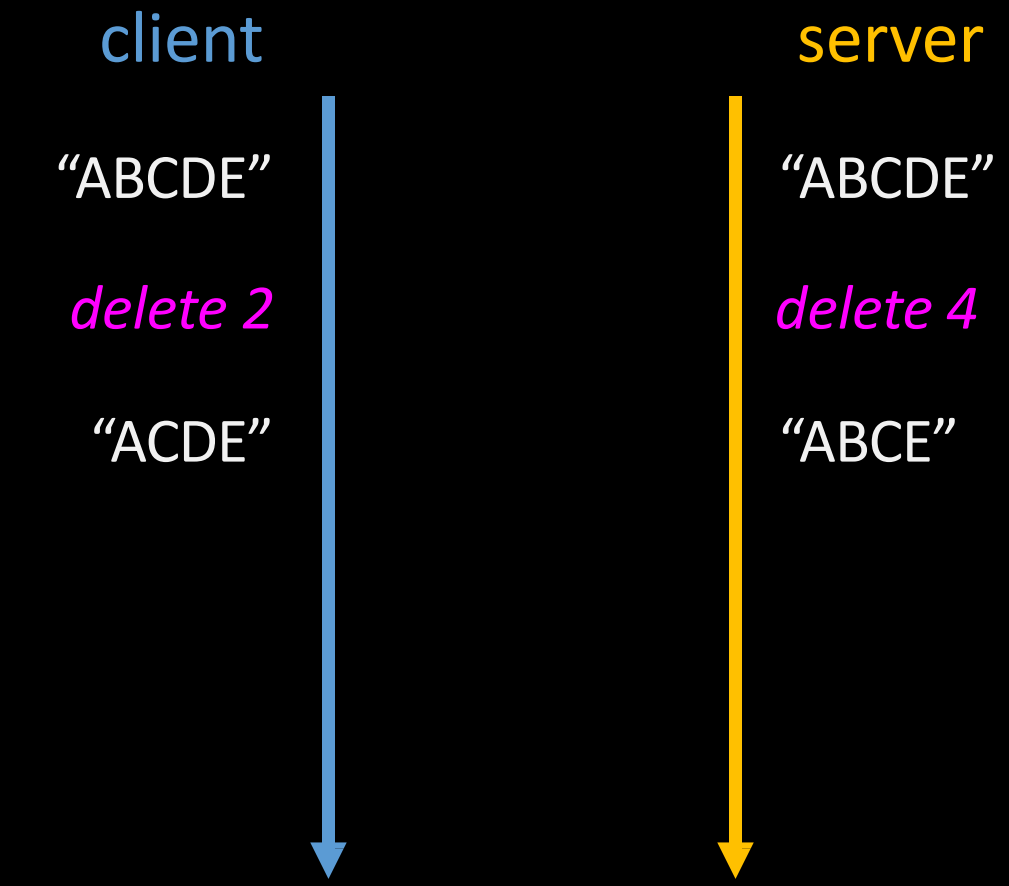
Operational transformation



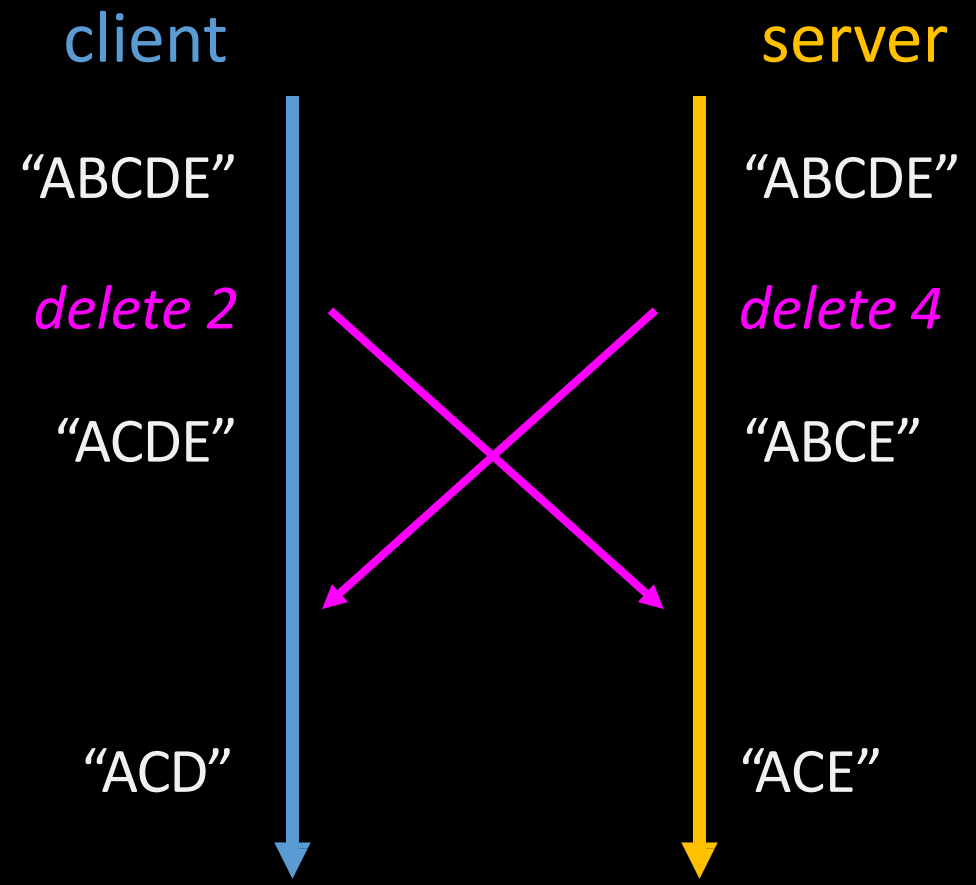
Operational transformation



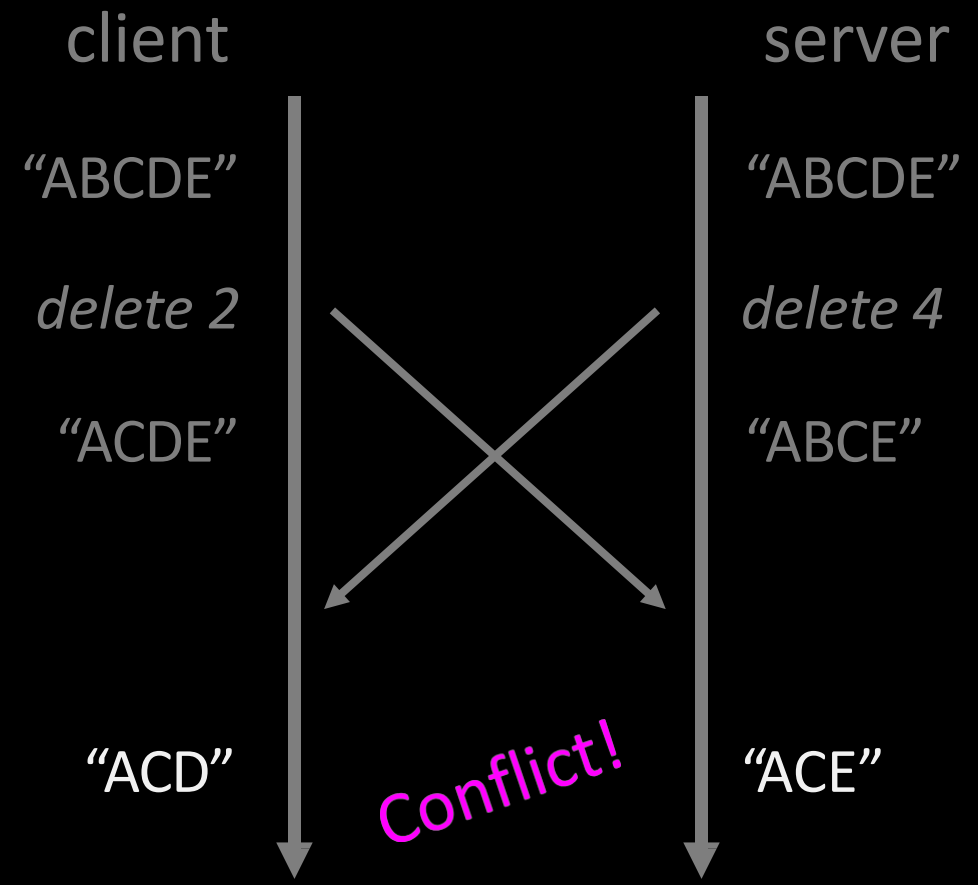
Example for operational transformation



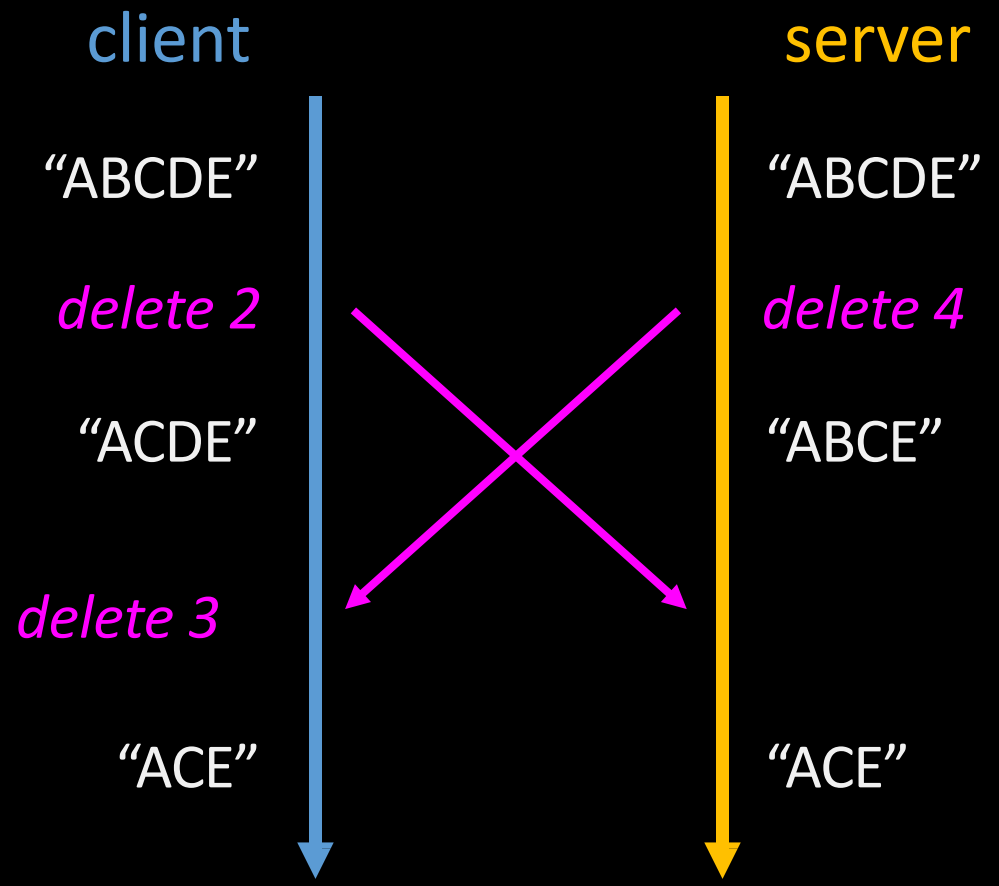
Example for operational transformation



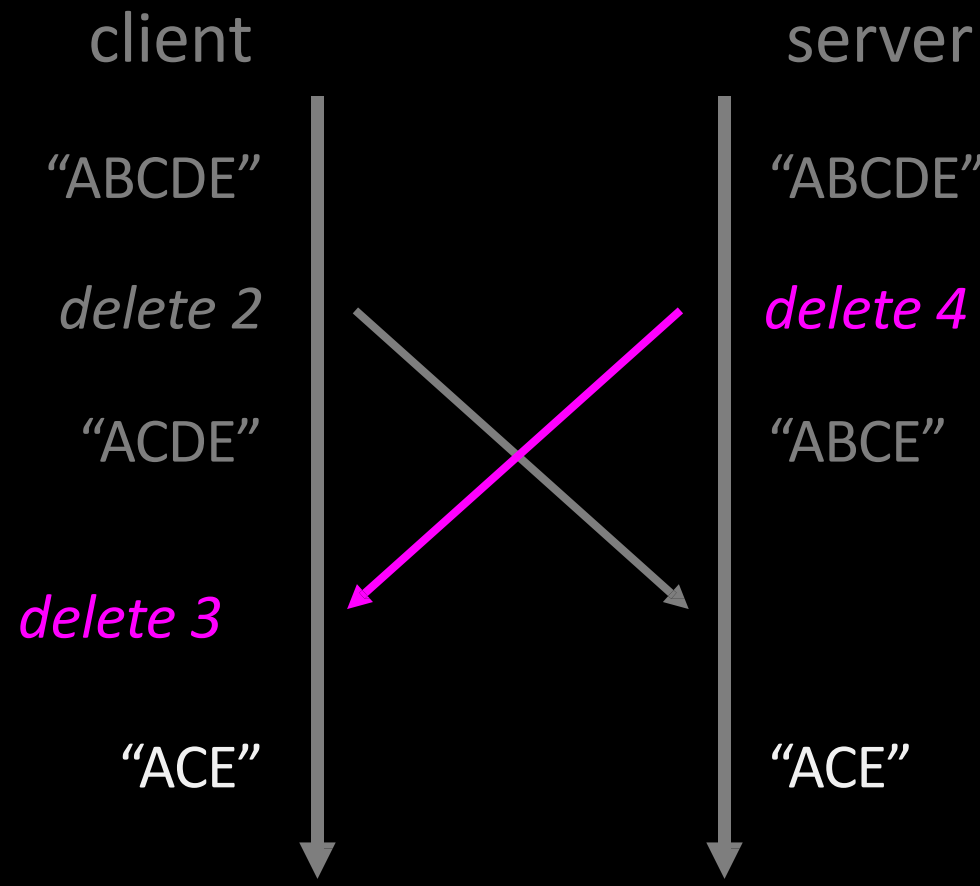
Example for operational transformation



Example for operational transformation



Example for operational transformation



“delete 4” must be transformed to “delete 3”

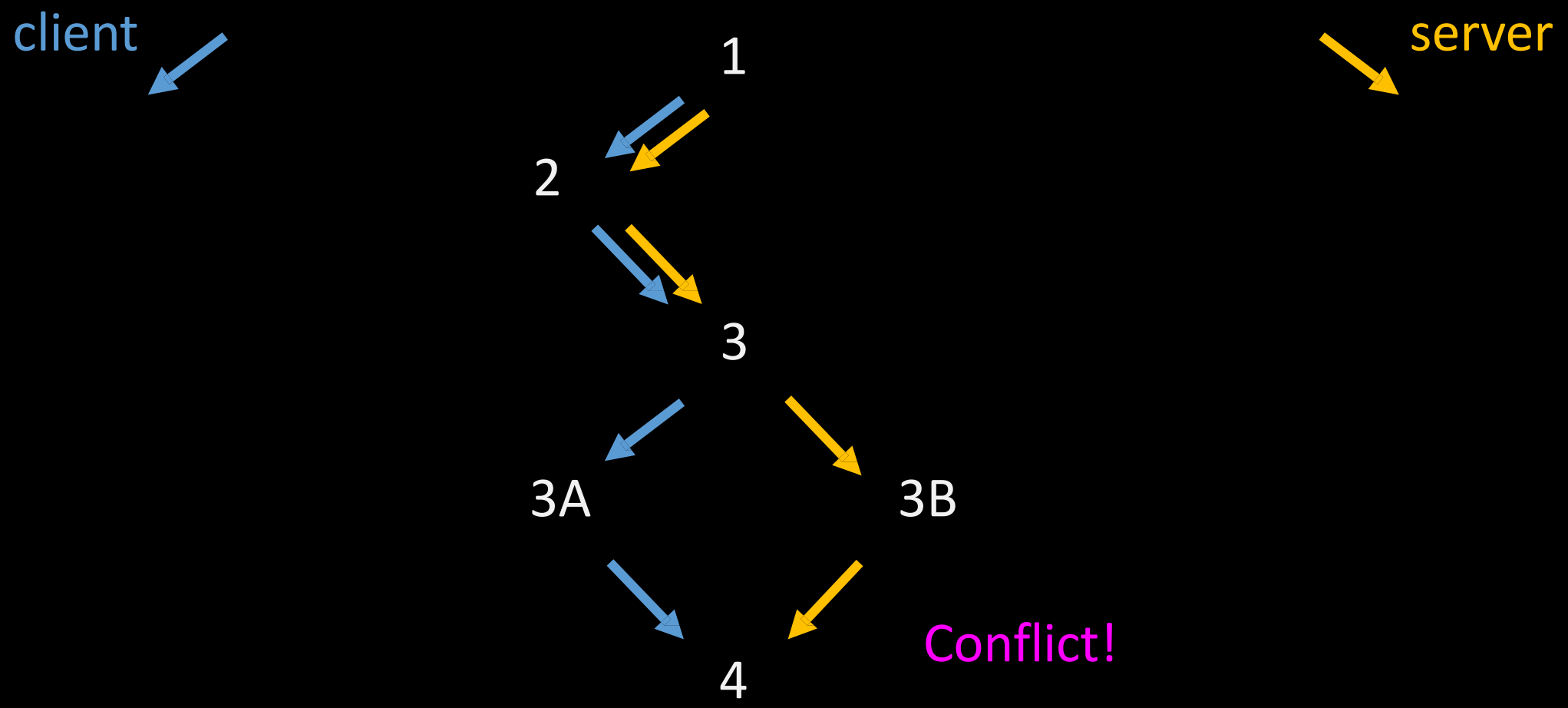
Google Docs

1. Apply Operation Transformation
2. Client and server exchange “change action”
3. Same code runs on both client and server
    - Google Web Toolkit

A real-time and offline editing service

Git

Commit & Revision



Git

1. Change unit is “line”
2. Commit and revision
3. Possible actions
   - Create new file / line
   - Edit a file / line
   - Delete a file / line
   - Move a file
4. Manual conflict resolving

Second problem:

Which real-time & offline solution?

Real-time frameworks and databases

1. Firebase
2. Deepstream.io
3. RethinkDB
4. CouchDB
5. PostgreSQL, PinelineDB, … (?)

Real-time frameworks and databases

1. Firebase
2. Deepstream.io
3. RethinkDB
4. CouchDB
5. PostgreSQL, PinelineDB, ... (2)

Let's try!

Other problems

1. Work on web and mobile (Android, iOS)
2. Keep client in sync with server
   - Can resume from a specific point in time
3. Edit history
   - Can rollback to a specific version

First version

Firebase

Why Firebase?

1. Real-time database (as a service)
2. Work on web and mobile (Android, iOS)
3. Store value as a big JSON
4. Permission on sub-tree
5. Quickly make a prototype

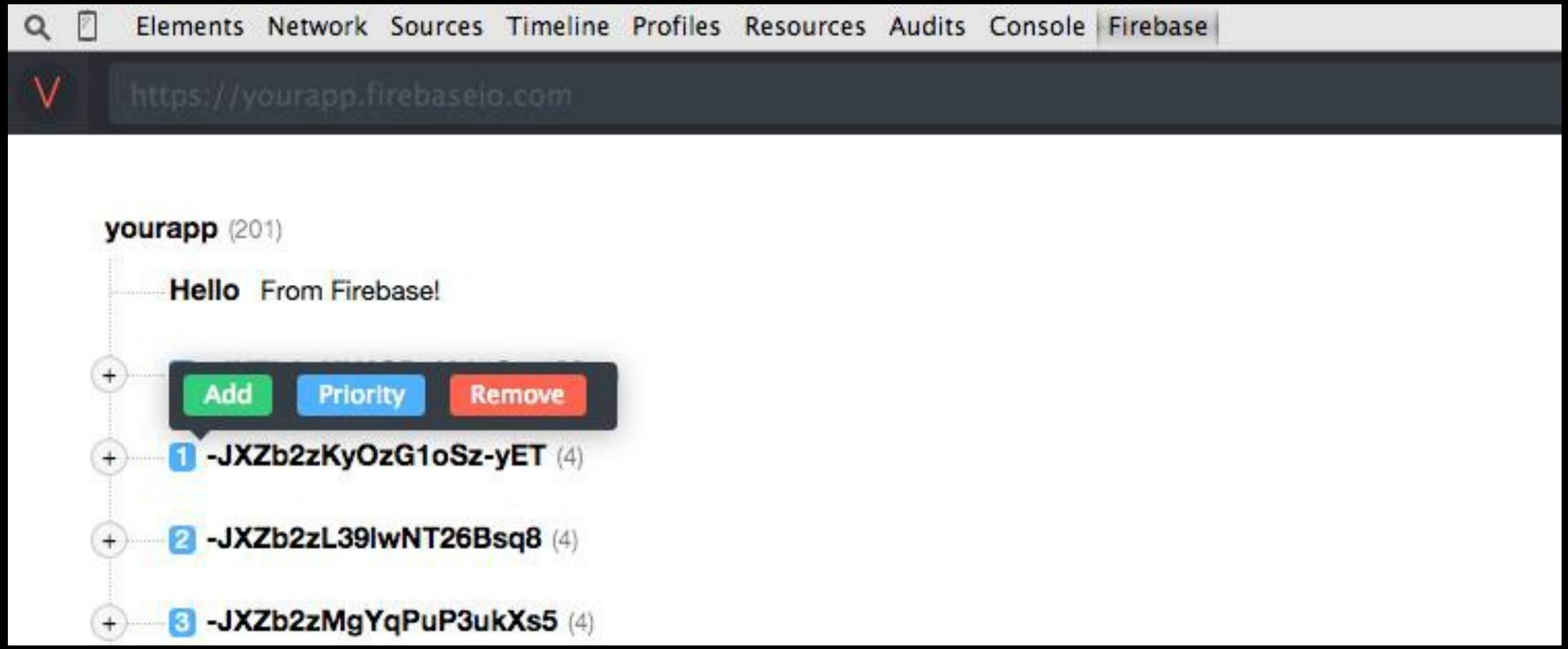
Sample code

```
var ref = new Firebase("https://<MY-FIREBASE-APP>.firebaseio.com");

ref.set({ name: "Alex Wolfe" });

ref.on("value", function(data) {
  var name = data.val().name;
  alert("My name is " + name);
});
```

Hello world

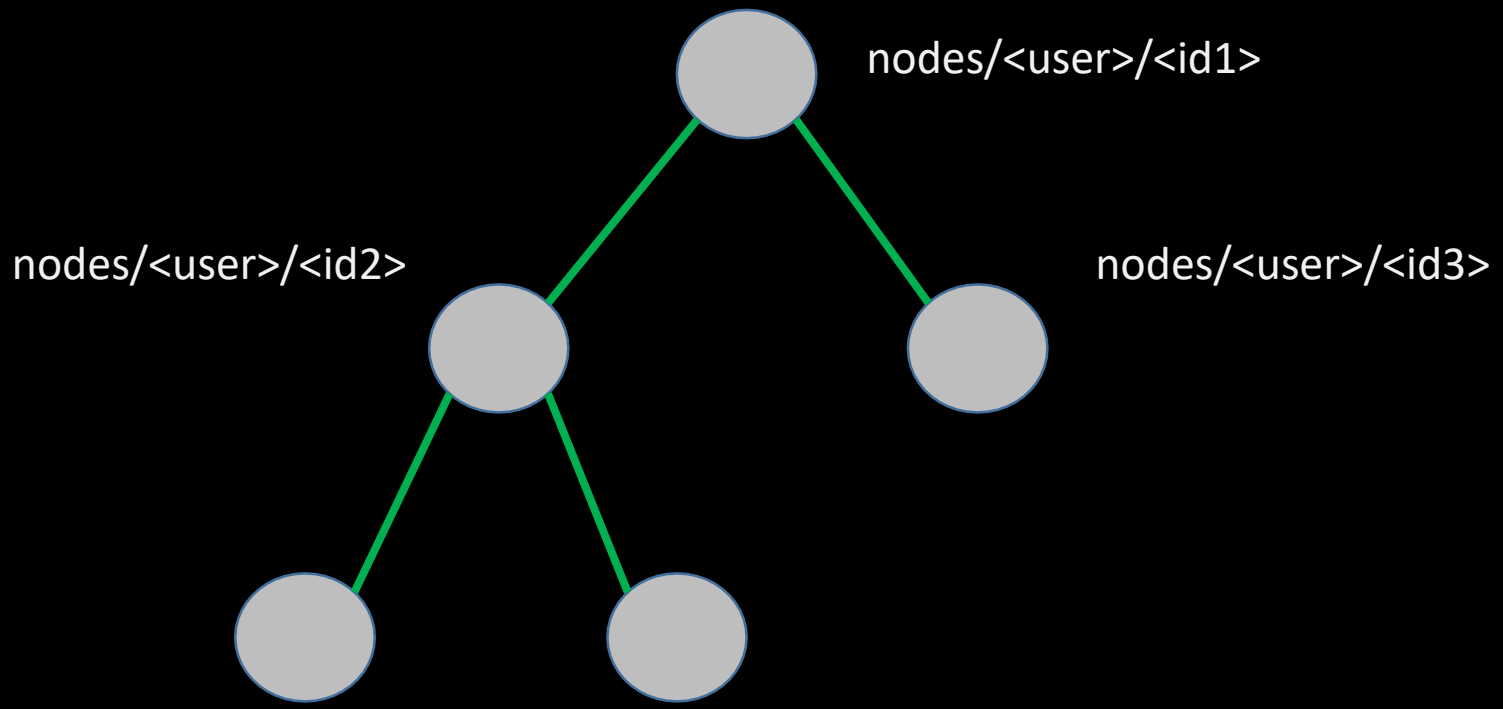
- This is an item
  - And a child that can be shared with other
    - Foo
    - Bar
    - Baz
    - Quix
  - Another child
- This is another item
- And one more!

Data as a tree



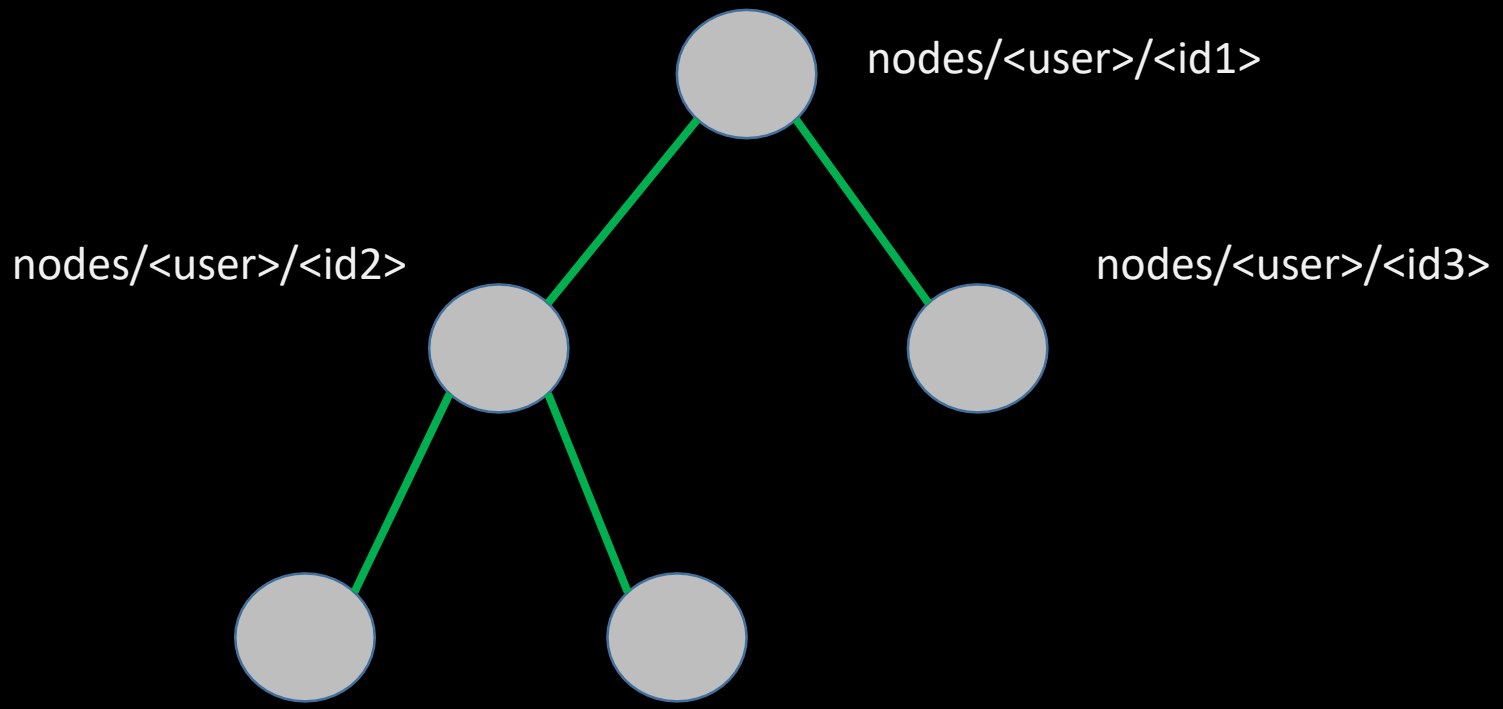
How do we structure data in Firebase

```
{
  nodes: {
    “u12345”: {
      “a3f0”: {
        parent_id: “b02c”,
        order: 18
      },
      “b02c”: {
        parent_id: “”,
        order: 1
      }
    }
}
```



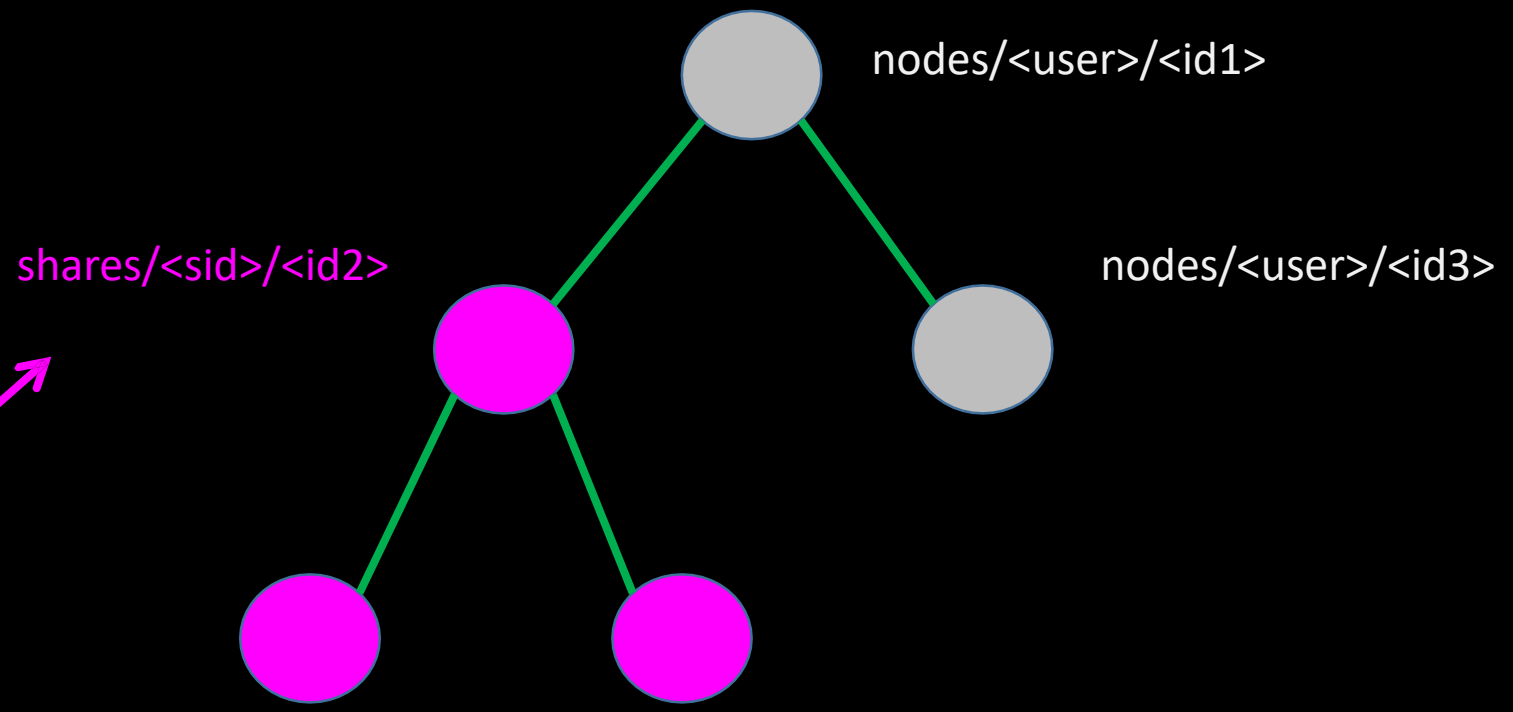
How do we structure data in Firebase

```
{
  users: {
    “u12345”: { … },
  },
  nodes: {
    “u12345”: {
       “a3f0”: … ,
       “b41c”: …
    }
  },
  shares: {
    “s98765”: { “2abf”: … }
  }
}
```



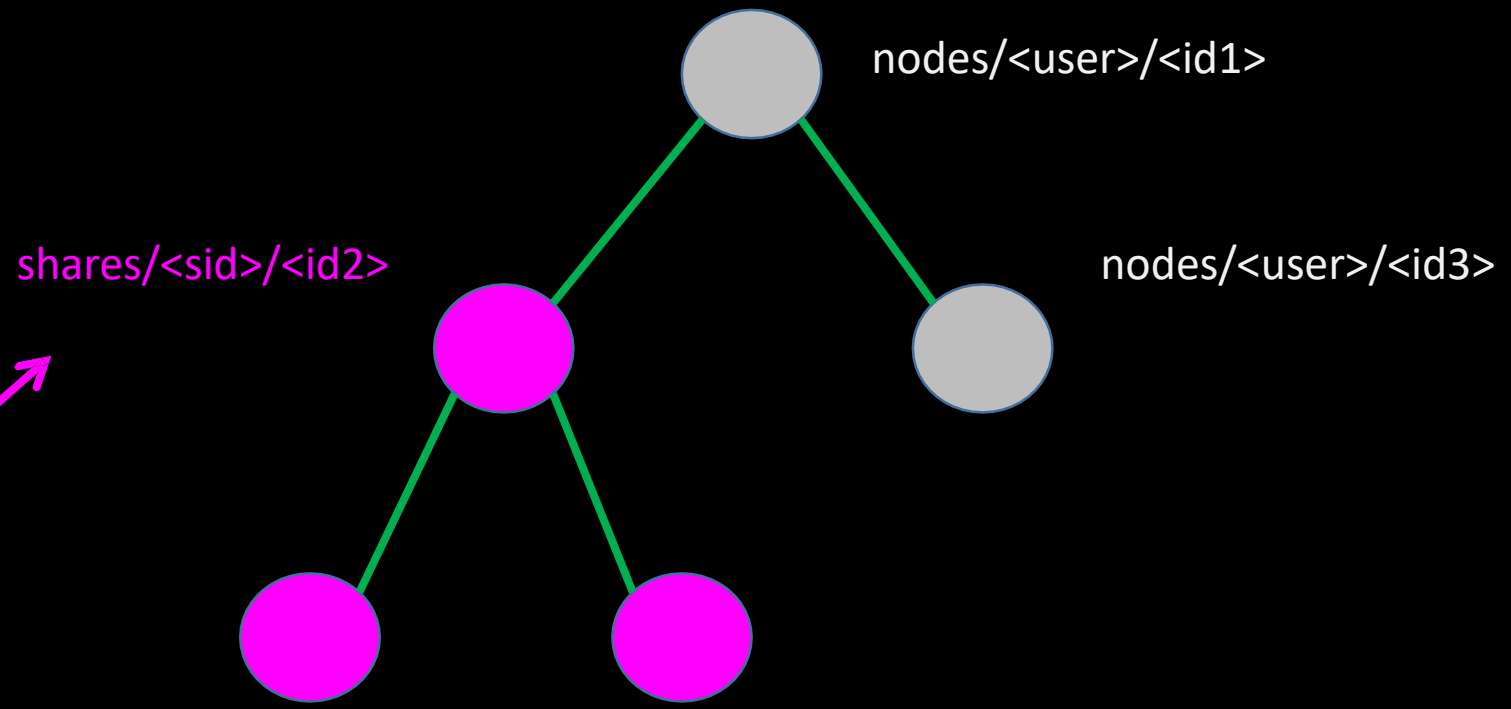
When we want to share a sub-tree

```
{
  users: {
    “u12345”: { … },
  },
  nodes: {
    “u12345”: {
       “a3f0”: … ,
       “b41c”: …
    }
  },
  shares: {
    “s98765”: { “2abf”: … }
  }
}
```



When we want to share a sub-tree

```
{
  users: {
    “u12345”: { … },
  },
  nodes: {
    “u12345”: {
        “a3f0”: … ,
        “b41c”: …
    }
  },
  shares: {
    “s98765”: { “2abf”: … }
  }
}
```



- Firebase does not support “move” action
- Permission is defined on sub-tree
- We have to copy data from “nodes” to “shares”

Why not Firebase?

1. Firebase does not support “move” action
2. Permission is defined on sub-tree
   - When sharing some nodes, we have to copy them from “nodes” to “shares”
   - This defeats synchronization!
3. Does not resolve conflict
   - Latest update win.
   - Workaround solutions like hack!

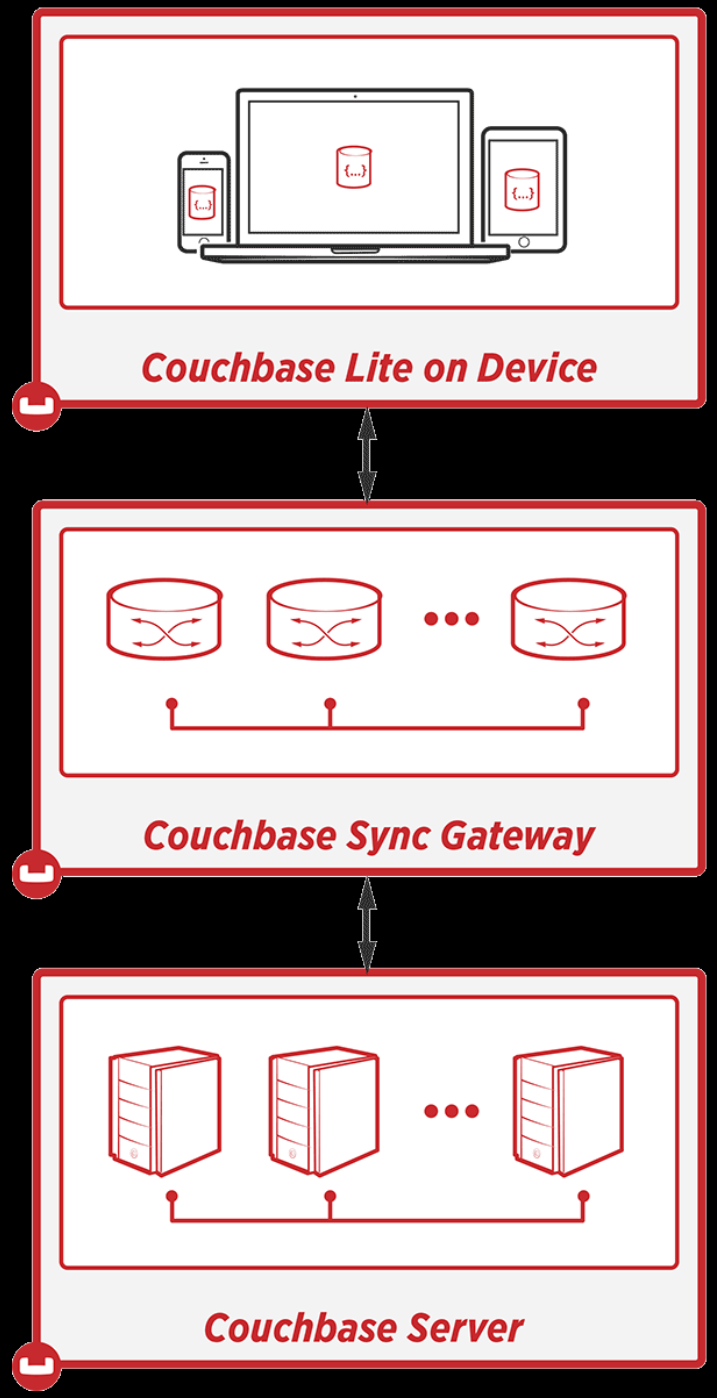
Second version

Couchbase Mobile

Couchbase Mobile?

1. Couchbase Server
2. Couchbase Sync Gateway
3. Couchbase Lite on Device

- Use CouchDB protocol
- PouchDB for offline JS database



Sample code in PouchBD

```
var db = new PouchDB('dbname');

db.put({
  _id: 'dave@gmail.com',
  name: 'David',
  age: 69
});

db.changes().on('change', function() {
  console.log('Ch-Ch-Changes');
});

db.replicate.to('http://example.com/mydb');
```

Why Couchbase Mobile?

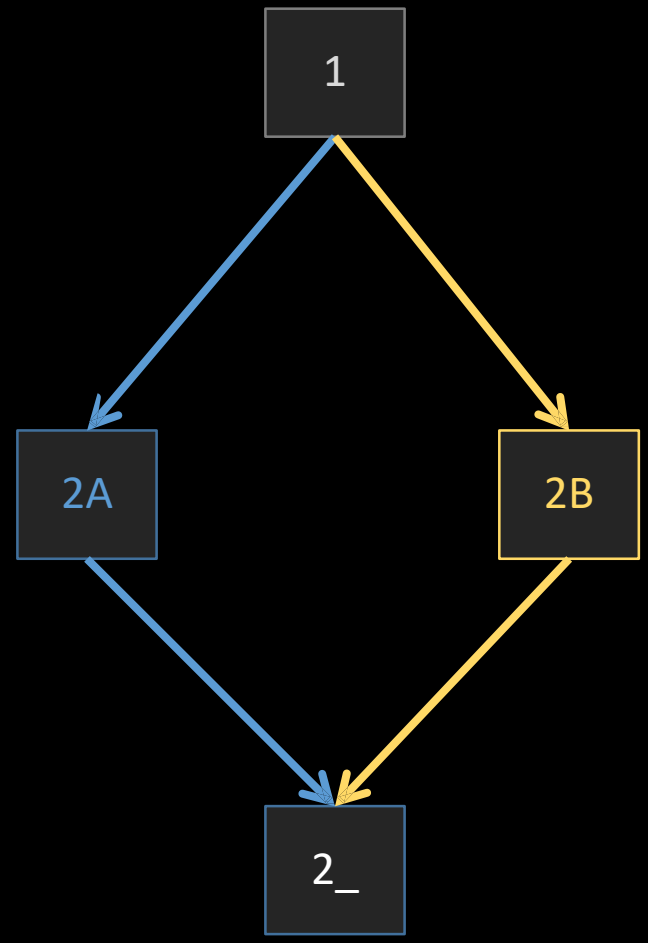
1. Real-time database (self-deployment)
2. Work on web and mobile (Android, iOS)
3. Store value as a JSON documents
4. Permission using channels
5. It does handle conflict!

Why Couchbase Mobile?

1. Real-time database (self-deployment)
2. Work on web and mobile (Android, iOS)
3. Store value as a JSON documents
4. Permission using channels
5. It does handle conflict!

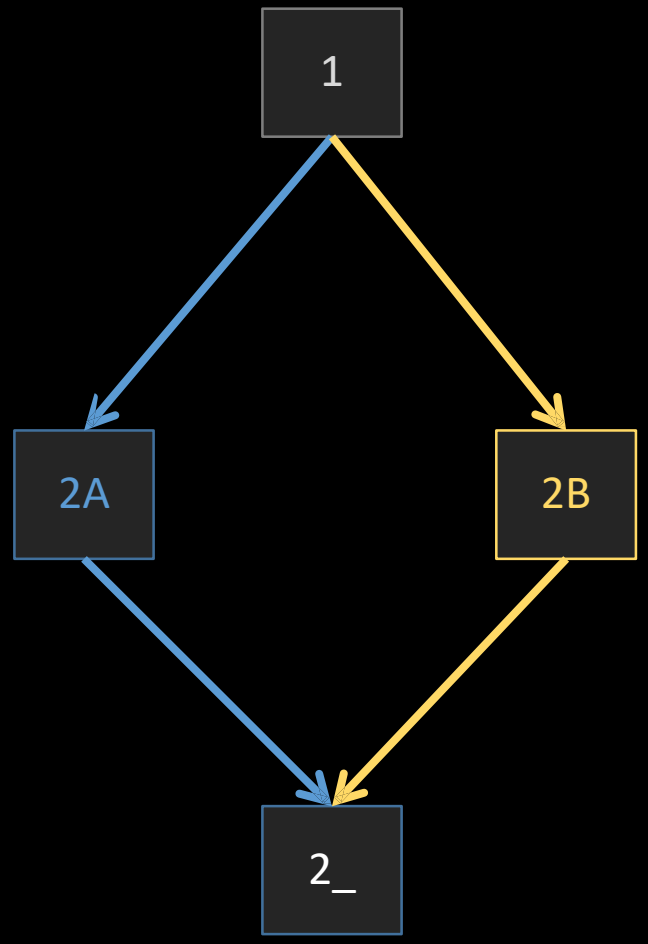
How Couchbase handle conflict?

```
db.put({
  _id: 'dave@gmail.com',
  _rev: '1-af2345c1',
  name:  'David',
  age: 69,
});

db.replicate.to(“http://...”)
```

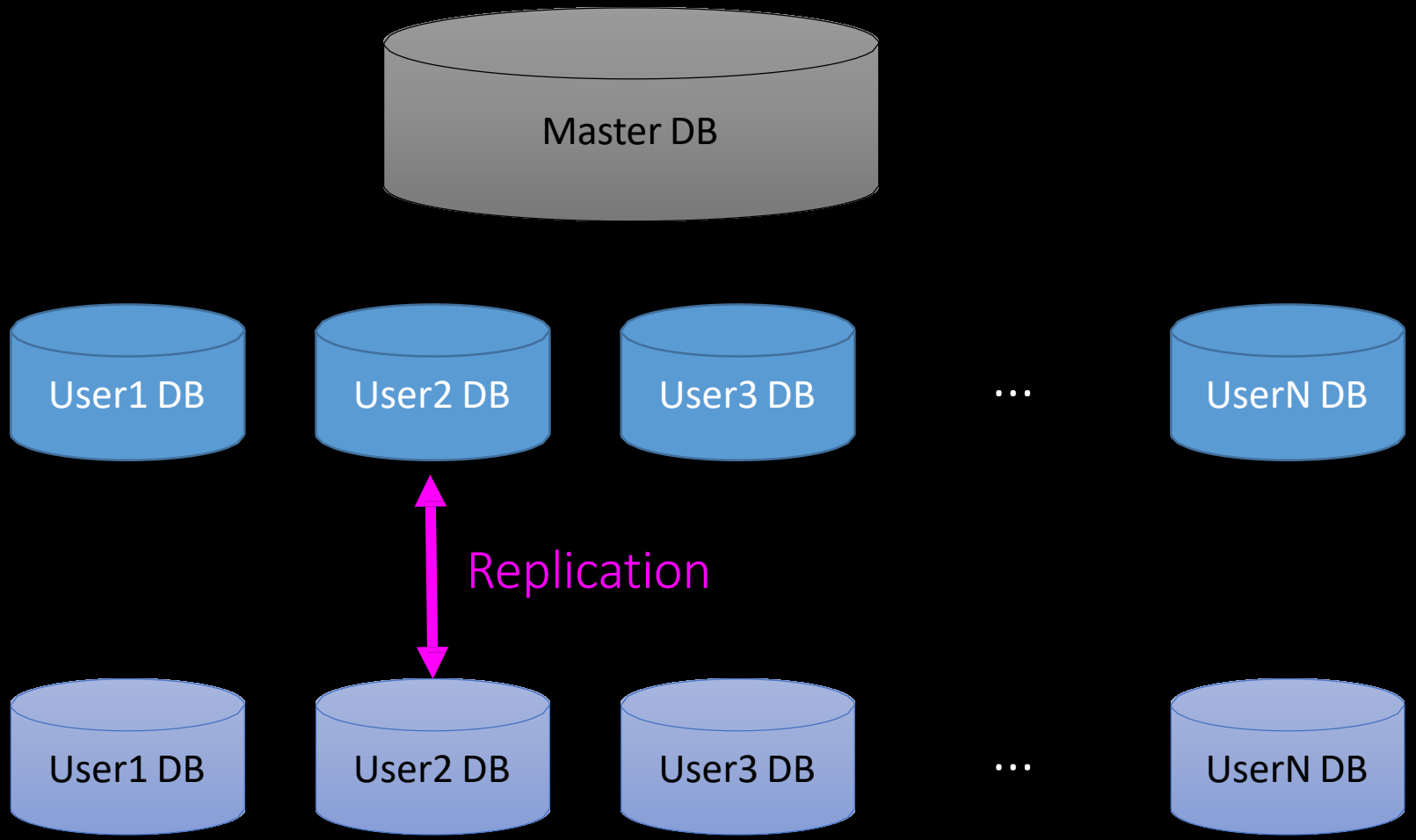


How Couchbase handle conflict?

1. Each change must include _rev
2. When two changes conflict
    1. Choose arbitrary winner
       based on a “deterministic algorithm”
    2. The document has two _rev numbers
3. Client know that a document is in conflicting status and can decide to resolve.
4. Deleted documents are kept in db
   { _id: “”, _rev: “”, _deleted: true }



Couchbase Mobile Permission



How Couchbase handle changes?

```
GET /<db>/_changes?since=900

{
  {"results":[
    {"seq":909,"id":"_user/1NEzeQ","changes":[]},
    {"seq":1317,"id":"mIM0Uuscg0MC","deleted":true,
      "changes":[{"rev":"1-96307afa"}]},
    {"seq":1318,"id":"mQQ0NwKUM0Dy","changes":[{"rev":"1-5fdcaba"}]},
    {"seq":1319,"id":"sLX0SKTpM41X","changes":[{"rev":"1-d15ee59"}]}
  }
}
```

How Couchbase handle changes?


```
{
  "_deleted": true,
  "_sync": {
    "rev": "1-91fe6048079942fb279c6c51cfd039f0",
    "sequence": 10392,
    "history": {
      "revs": [
        "1-91fe6048079942fb279c6c51cfd039f0"
      ],
    }
}
```

How Couchbase handle changes?

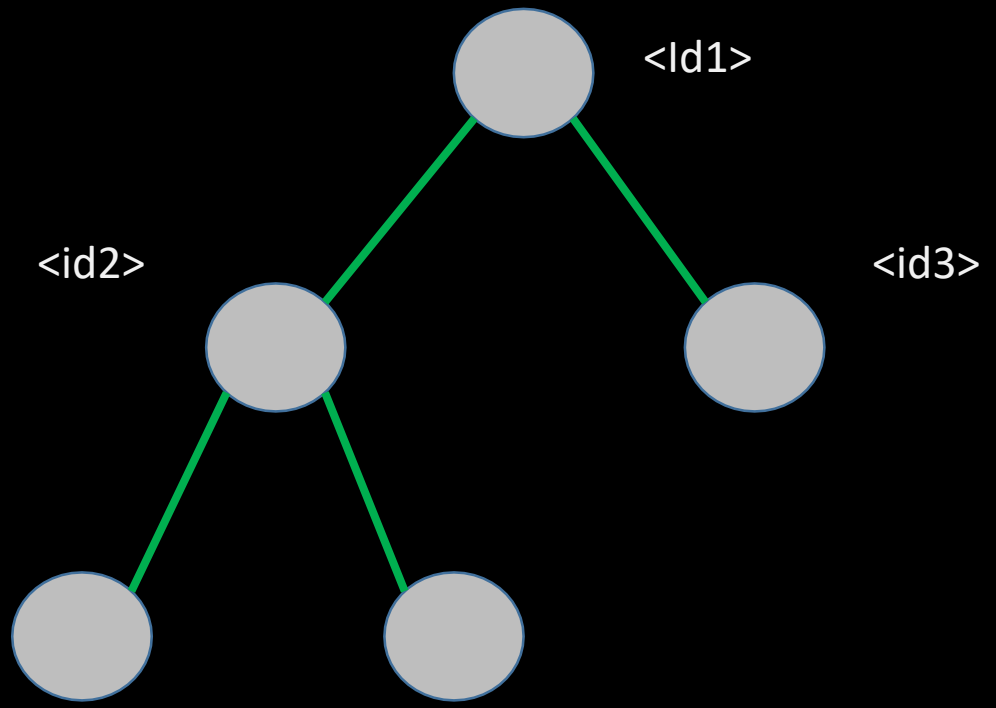
1. Each update include a sequence number
2. Client can request changes by ?since=seq
   - Only latest document revisions are returned
3. Client can request by long-polling or websocket

Why Couchbase Mobile?

1. Real-time database (self-deployment)
2. Work on web and mobile (Android, iOS)
3. Store value as a JSON documents
4. Permission using channels
5. It does handle conflict!

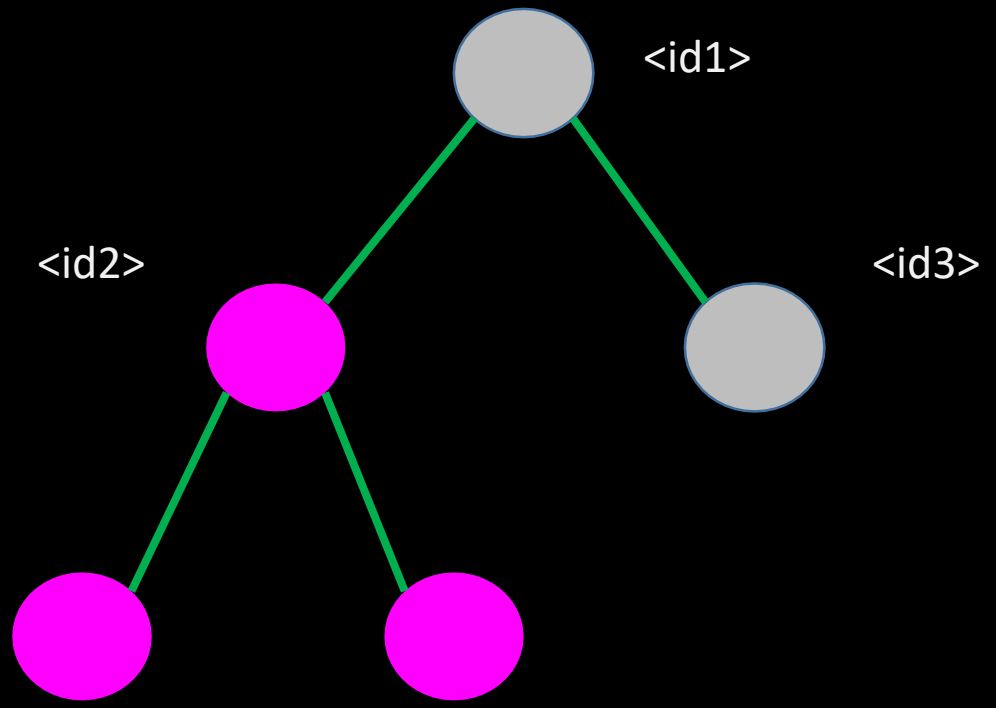
How do we structure data in Couchbase Mobile

```
{
  _id: “a3f0”,
  _rev: “…”
  parent_id: “b02c”,
  order: 18
  shares: {
    “u231b”: …
  }
}
```



How do we structure data in Couchbase Mobile

```
{
  _id: “a3f0”,
  _rev: “…”
  parent_id: “b02c”,
  order: 18
  shares: {
    “u231b”: …
  }
}
```



- Couchbase Mobile does not understand parent-child relationship
- We have to update “shares” property on all child nodes
  => All child nodes will be updated and synchronized to all clients

Why not Couchbase Mobile?

1. Permission based on each document
   - It does not understand parent-child relationship
2. We can not easily share a sub-tree
   - Have to update all children nodes
   - This defeats synchronization purpose!

Third version

Build our own solution with Couchbase

What did we learn?

1. We need our own data structure and API
   - Pre-built solutions are nice. But they do not fit to our app.
2. We learned how Couchbase Sync Gateway implement synchronization and conflict resolving
3. We learned from Couchbase Sync Gateway
   - It is written in Go!

What do we have to do?

1. Our own data structure and API
2. Sync service
3. SDK for JavaScript and mobiles (Android and iOS)
4. Synchronization and conflict solution
5. Sharing solution

Why still Couchbase?

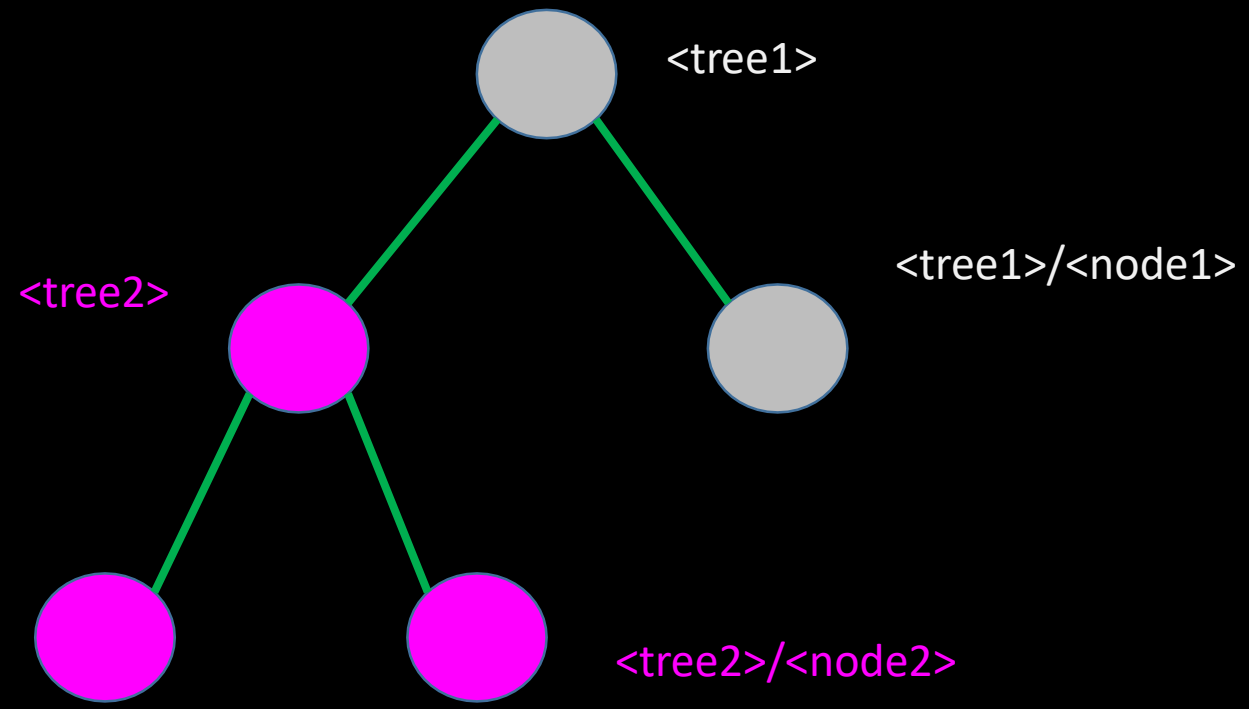
1. Both database and memcached
2. Data Change Protocol for replication (DCP)
   - Can resume at a specific time
   - Can be used to replicate whole database content

Sample DCP receiver

```
func (r *Receiver) OnError(err error)
func (r *Receiver) DataUpdate(vbucketId uint16, key []byte, seq uint64,
       req *gomemcached.MCRequest) error
func (r *Receiver) DataDelete(vbucketId uint16, key []byte, seq uint64,
       req *gomemcached.MCRequest) error
func (r *Receiver) SetMetaData(vbucketId uint16, value []byte) error
func (r *Receiver) GetMetaData(vbucketId uint16)
func (r *Receiver) Rollback(vbucketId uint16, rollbackSeq uint64) error
```

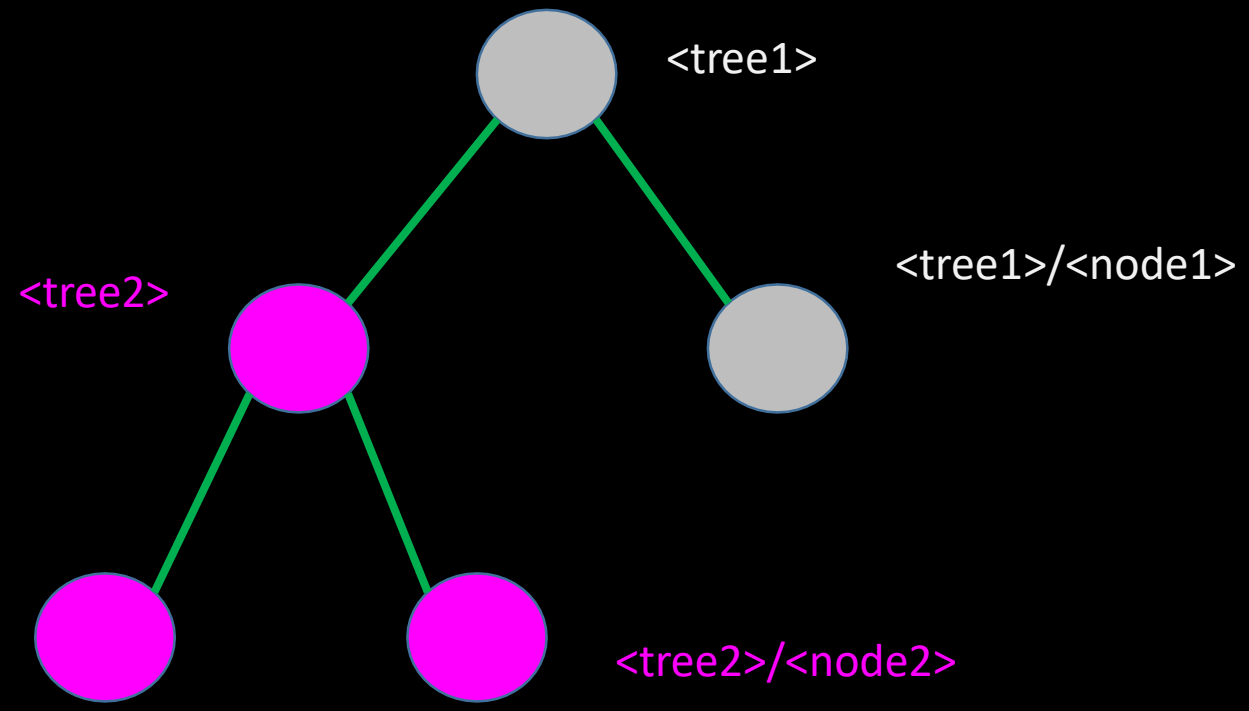
How do we structure data in Couchbase

```
// Tree
{
  _id: “…”,
  _rev: “…”,
  shares: { … },
  items: []
}

// Node
{
  _id: “…”,
  _rev: “…”,
  content: []
}
```



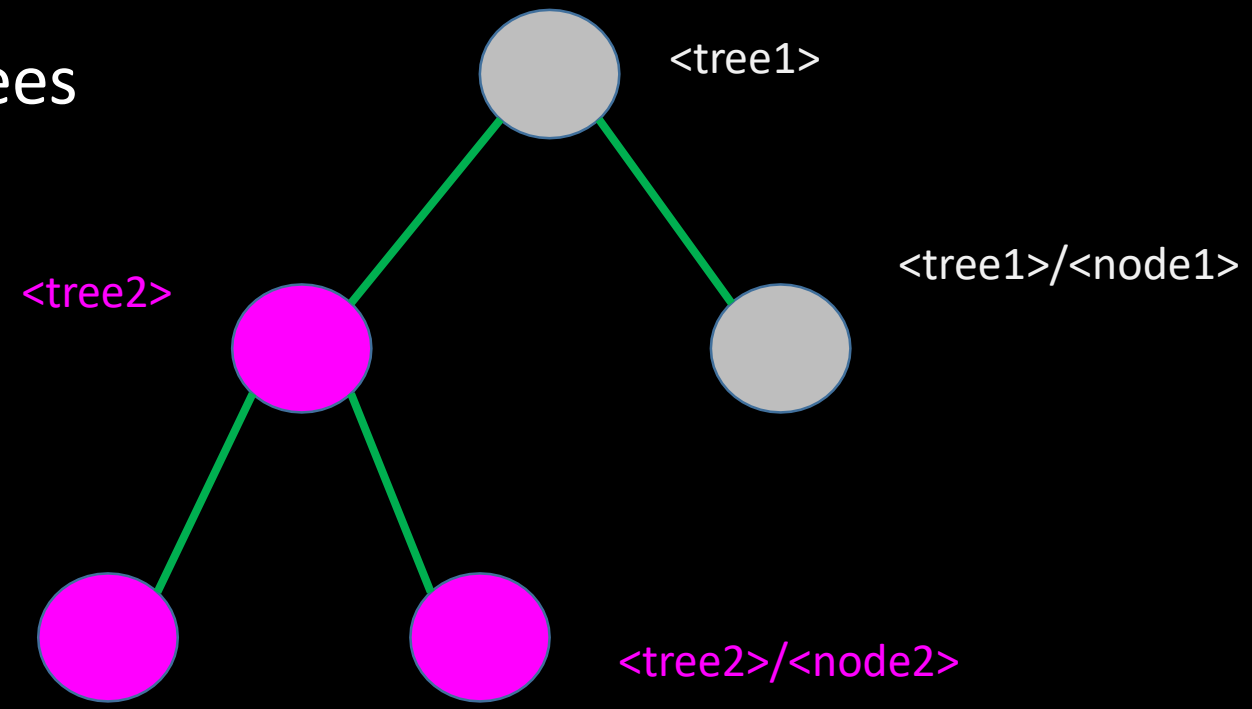
How do we handle changes?

```
GET /change?since=524
{
  trees: [
    { _id: “…”, _rev: “…” }
  ],
  nodes: [
    { _id: “…”, _rev: “…” }
  ],
  current_seq: “…”
}
```



How do we share?

1. Only define permission on trees
2. Split tree when needed
3. Tree has it own revisions
4. Nodes are used to store data (without explicit permission)



Remaining problems

1. Change set & history
2. Moving & tree conflict
3. Optimize loading: load important data first
4. Improve caching
5. More tests
6. …

Grokking #9

THANK YOU

Oliver N.

Software Engineer

Grokking #9

Building an offline & real-time editing service with Couchbase

Oliver N.

Software Engineer