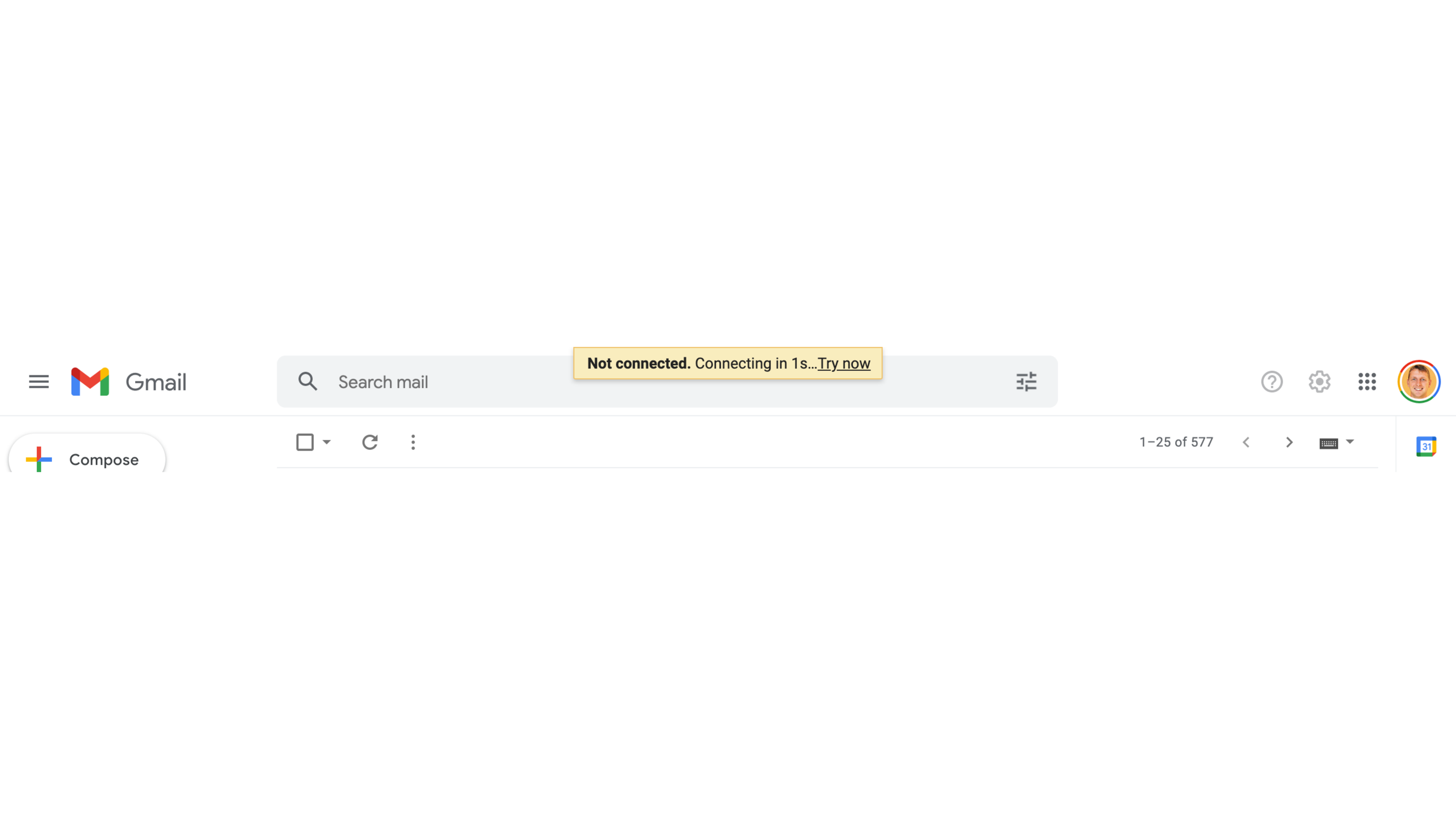
Offline-first data sharing with peer-to-peer databases

Robbie Mackay

Can we share data and collaborate when offline?


Gmail
Search mail
Not connected. Connecting in 1s... Try now
Compose
1–25 of 577

Offline-first

Can you view, edit and search when offline?

Offline data sharing

Can you share data while offline?

Offline data sharing

Can you share data while offline?

… with only a browser?

Peer-to-peer

Sending data direct from A-to-B
Doesn’t rely on a central server

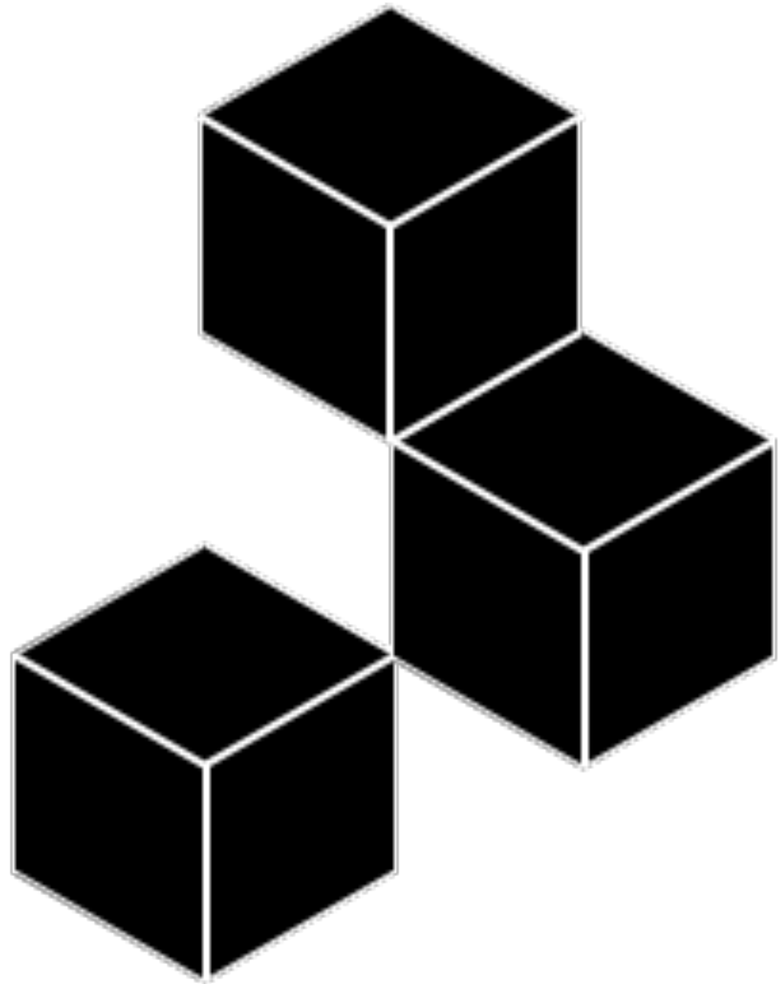
Goals

- Can we view and edit data offline?
- Can we share data without internet?
- Can we share data without a central server?
- Can we share data with only a browser?

P2P / decentralised building blocks

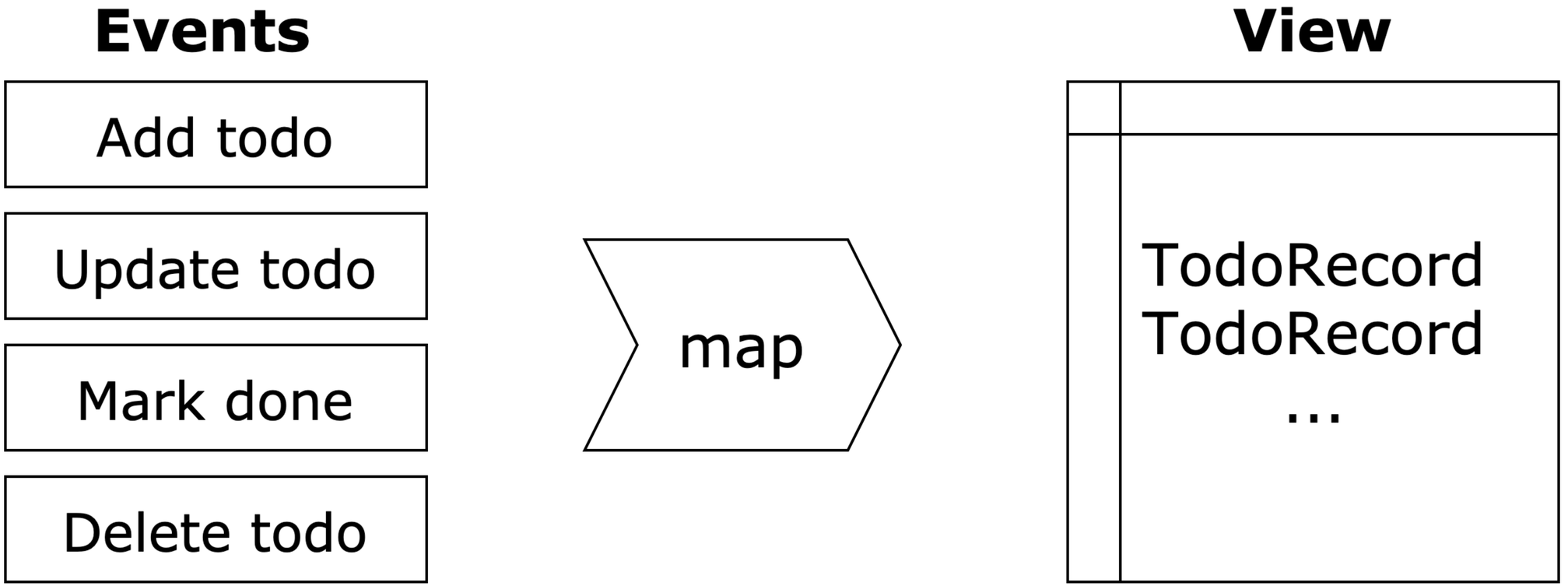
Hypercore

- Hypercore
- Hyperswarm
- Hyperdrive



Append-only log

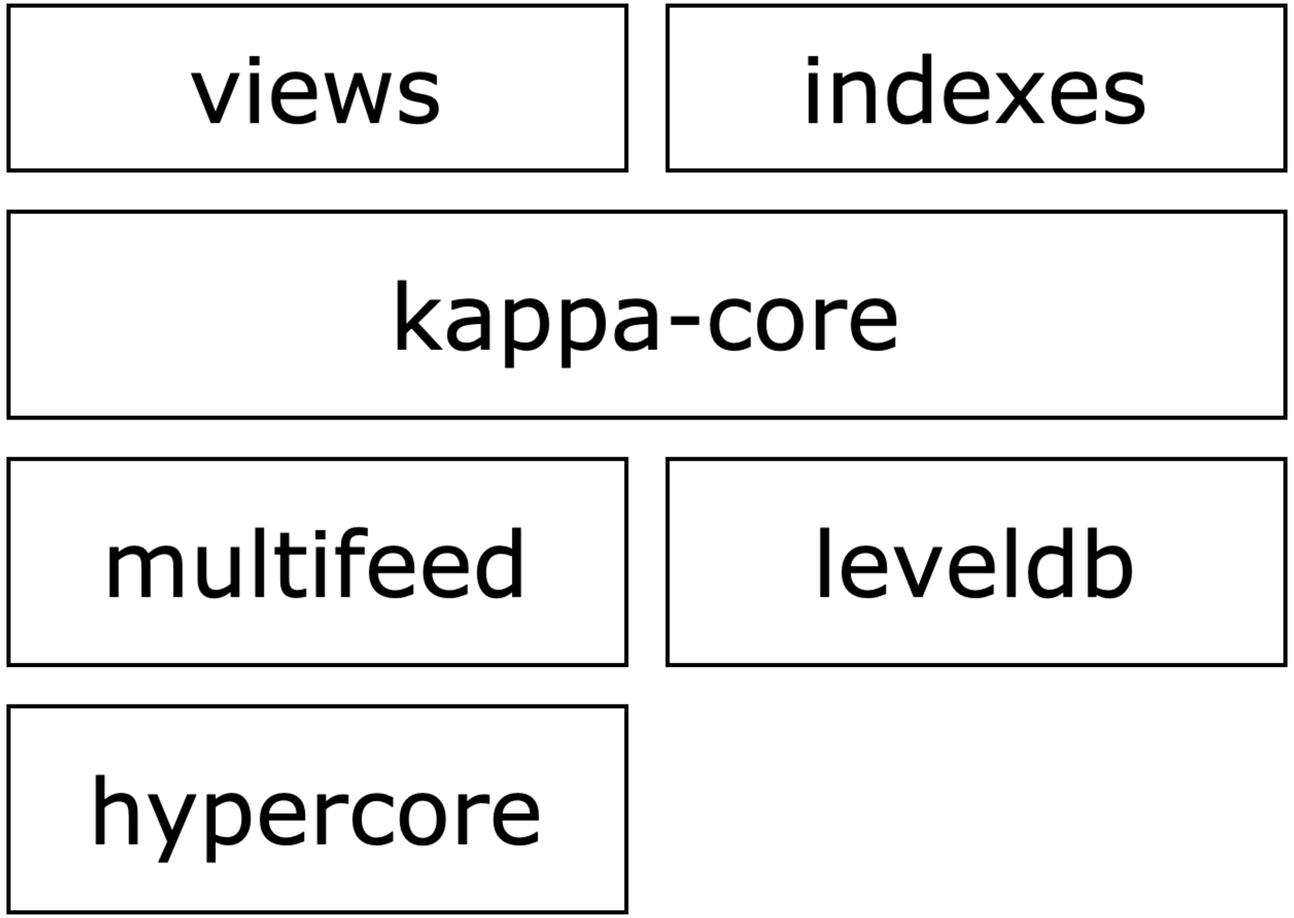
kappa-db



Event sourcing

```
{
    type: "put",
    id: 1,
    text: "Do the thing!",
    done: false,
}
```

```
{
    type: "del",
    id: 1,
    text: "Do the thing!",
    done: false,
}
```


views
indexes
kappa-core
multifeed
leveldb
hypercore

```
npm i kappa-core
```

Setting up

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });
```

Setting up

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });
```

random-access-storage

Add some data

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });

// Get a writable hypercore
core.writer("local", async function (err, feed) {
    // Add a todo to our log
    const todo = { id: 1, text: "Do the thing!", done: false };
    feed.append({
        type: 'put',
        ...todo
    });
});
```

Add some data

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });

// Get a writable hypercore
core.writer("local", async function (err, feed) {
    // Add a todo to our log
    const todo = { id: 1, text: "Do the thing!", done: false };
    feed.append({
        type: 'put',
        ...todo
    });
});
```

Add some data

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });

// Get a writable hypercore
core.writer("local", async function (err, feed) {
    // Add a todo to our log
    const todo = { id: 1, text: "Do the thing!", done: false };
    feed.append({
        type: 'put',
        ...todo
    });
});
```

Using a view

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');
const recordView = require('./record-view');
const level = require('level-mem');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });
core.use('records', recordView(level({ valueEncoding: "json" })))

core.ready([], function () {
    core.api.records.all(function (data) {
        console.log(data);
    })
})
```

Using a view

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');
const recordView = require('./record-view');
const level = require('level-mem');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });
core.use('records', recordView(level({ valueEncoding: "json" })))

core.ready([], function () {
    core.api.records.all(function (data) {
        console.log(data);
    })
})
```

Using a view

```
const kappa = require('kappa-core');
const ram = require('random-access-memory');
const recordView = require('./record-view');
const level = require('level-mem');

// Create a kappa instance.
const core = kappa(ram, { valueEncoding: "json" });
core.use('records', recordView(level({ valueEncoding: "json" })))

core.ready([], function () {
    core.api.records.all(function (data) {
        console.log(data);
    })
})
```

More data...

```
core.writer("local", async function (err, feed) {
    // Add a todo to our log
    const todo1 = { id: 1, text: "Do the thing!", done: false };
    const todo2 = { id: 2, text: "Do more!", done: false };
    feed.append({
        type: 'put',
        ...todo1
    });
    feed.append({
        type: 'put',
        ...todo2
    });
    feed.append({
        type: 'del',
        ...todo1
    });
    feed.append({
        type: 'put',
        ...todo2,
        done: true
    });
});
```

The view

```
const makeView = require("kappa-view");

module.exports = (storage) => {
    return makeView(storage, { valueEncoding: "json" }, function (db) {
        return {
            map: function (entries, next) {
                const batch = entries.map(function (entry) {
                    const { key, type, ...value } = entry.value;
                    return {
                        type: type === "del" ? "del" : "put",
                        key,
                        value,
                    };
                });
                db.batch(batch, next);
            },
        };
    });
};
```

The view

```
const makeView = require("kappa-view");

module.exports = (storage) => {
    return makeView(storage, { valueEncoding: "json" }, function (db) {
        return {
            map: function (entries, next) {
                const batch = entries.map(function (entry) {
                    const { key, type, ...value } = entry.value;
                    return {
                        type: type === "del" ? "del" : "put",
                        key,
                        value,
                    };
                });
                db.batch(batch, next);
            },
        };
    });
};
```

The view

```
const makeView = require("kappa-view");

module.exports = (storage) => {
    return makeView(storage, { valueEncoding: "json" }, function (db) {
        return {
            map: function (entries, next) {
                const batch = entries.map(function (entry) {
                    const { key, type, ...value } = entry.value;
                    return {
                        type: type === "del" ? "del" : "put",
                        key,
                        value,
                    };
                });
                db.batch(batch, next);
            },
        };
    });
};
```

The view

```
const makeView = require("kappa-view");

module.exports = (storage) => {
    return makeView(storage, { valueEncoding: "json" }, function (db) {
        return {
            map: function (entries, next) {
                const batch = entries.map(function (entry) {
                    const { key, type, ...value } = entry.value;
                    return {
                        type: type === "del" ? "del" : "put",
                        key,
                        value,
                    };
                });
                db.batch(batch, next);
            },
        };
    });
};
```

The view

```
const makeView = require("kappa-view");

module.exports = (storage) => {
    return makeView(storage, { valueEncoding: "json" }, function (db) {
        return {
            map: function (entries, next) {
                const batch = entries.map(function (entry) {
                    const { key, type, ...value } = entry.value;
                    return {
                        type: type === "del" ? "del" : "put",
                        key,
                        value,
                    };
                });
                db.batch(batch, next);
            },
        };
    });
};
```

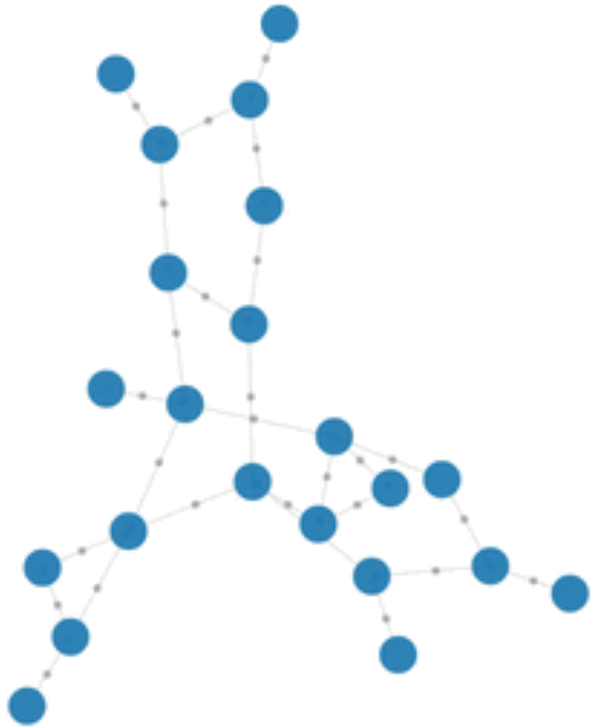
The view

```
api: {
    get: function (core, key, cb) {
        core.ready(function () {
            db.get(key, cb);
        });
    },
    all: function (core, cb) {
        core.ready(() => {
            const data = [];
            db.createReadStream()
                .on("data", (entry) => {
                    data.push(entry);
                })
                .on("end", () => {
                    cb(data);
                });
        });
    },
},
```

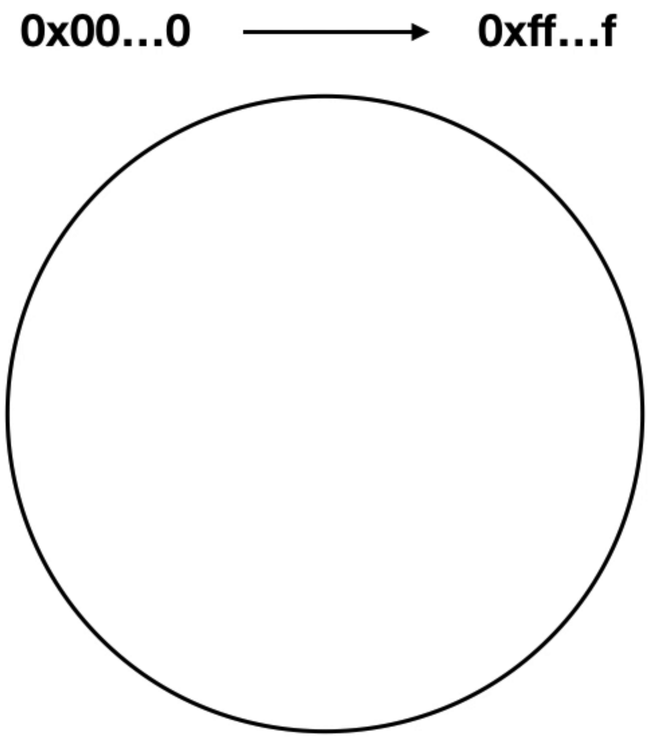
How do we share data?

Hyperswarm

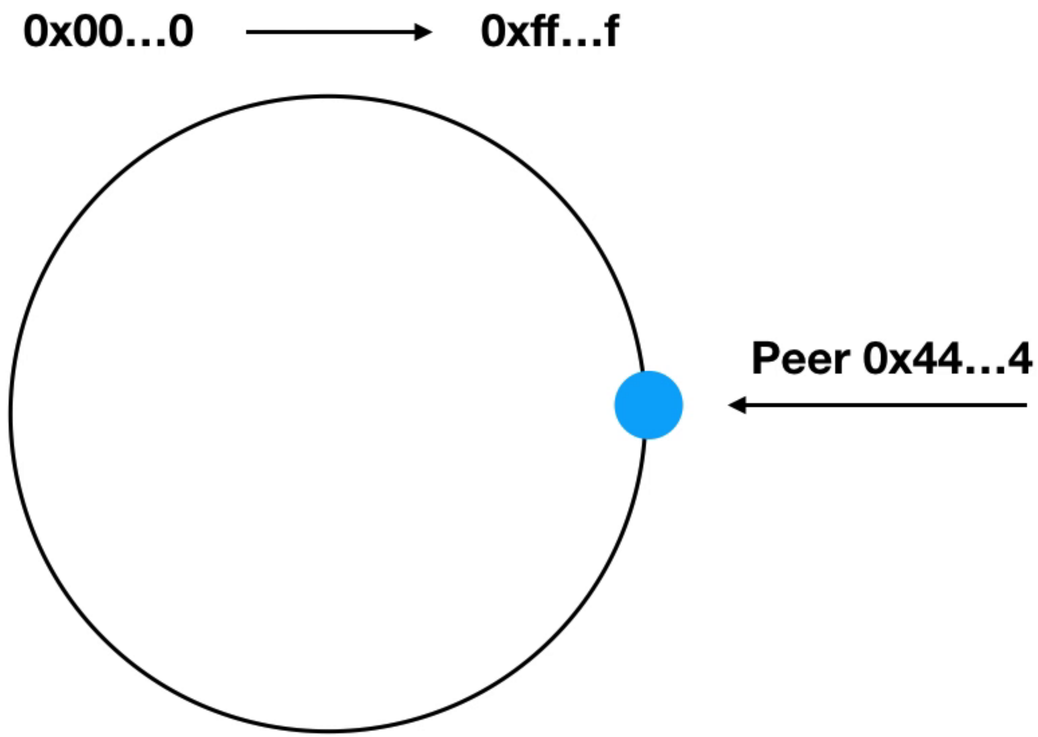
- Distributed Hash Table (DHT) for global peer discovery
- mDNS for local network peer discovery



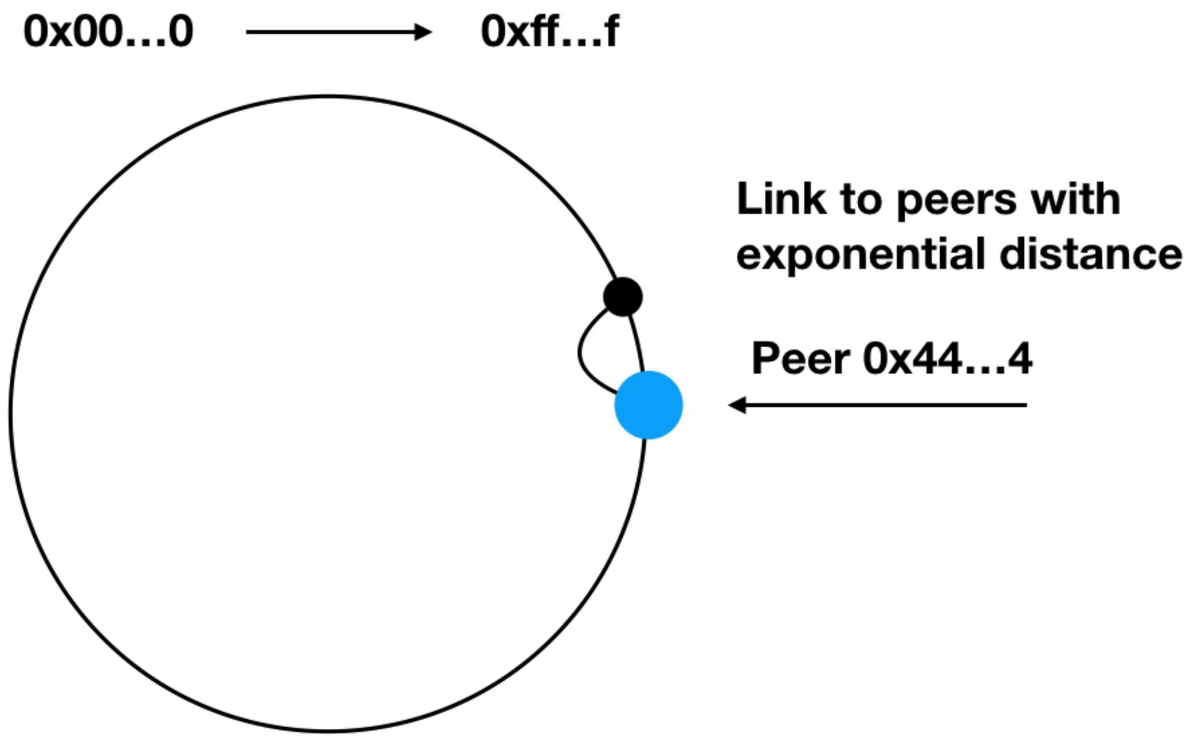
Kademlia DHT



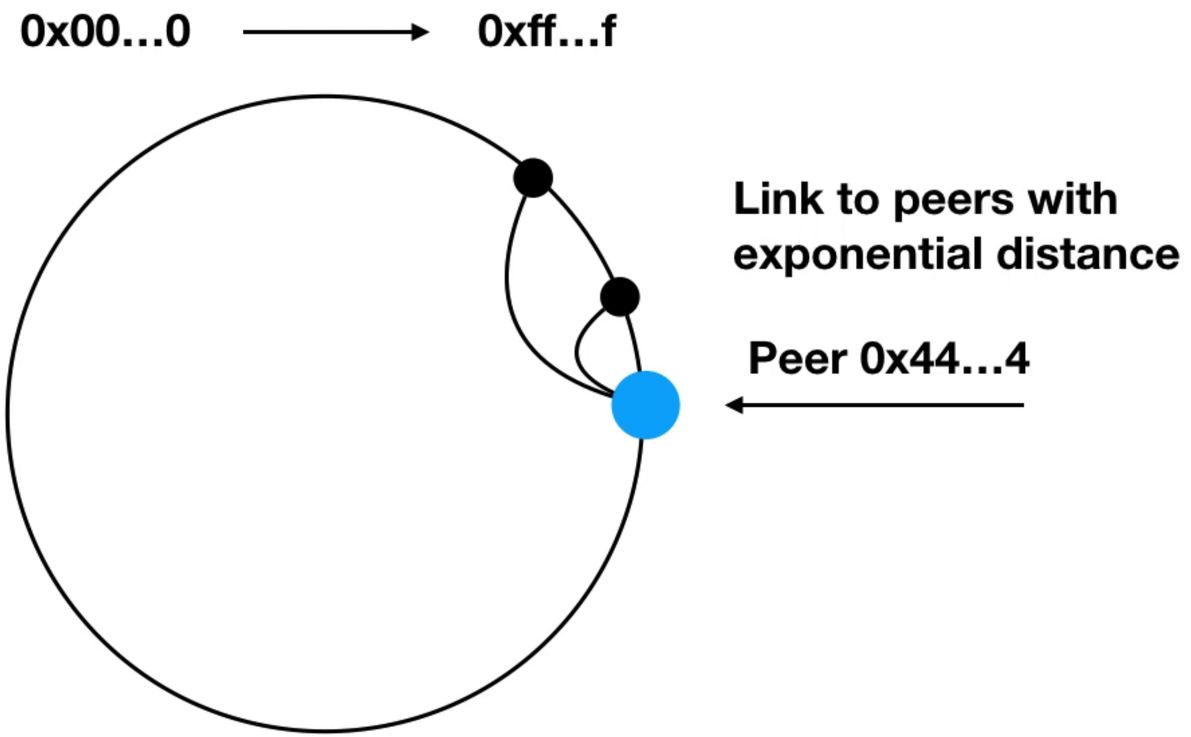
Kademlia DHT



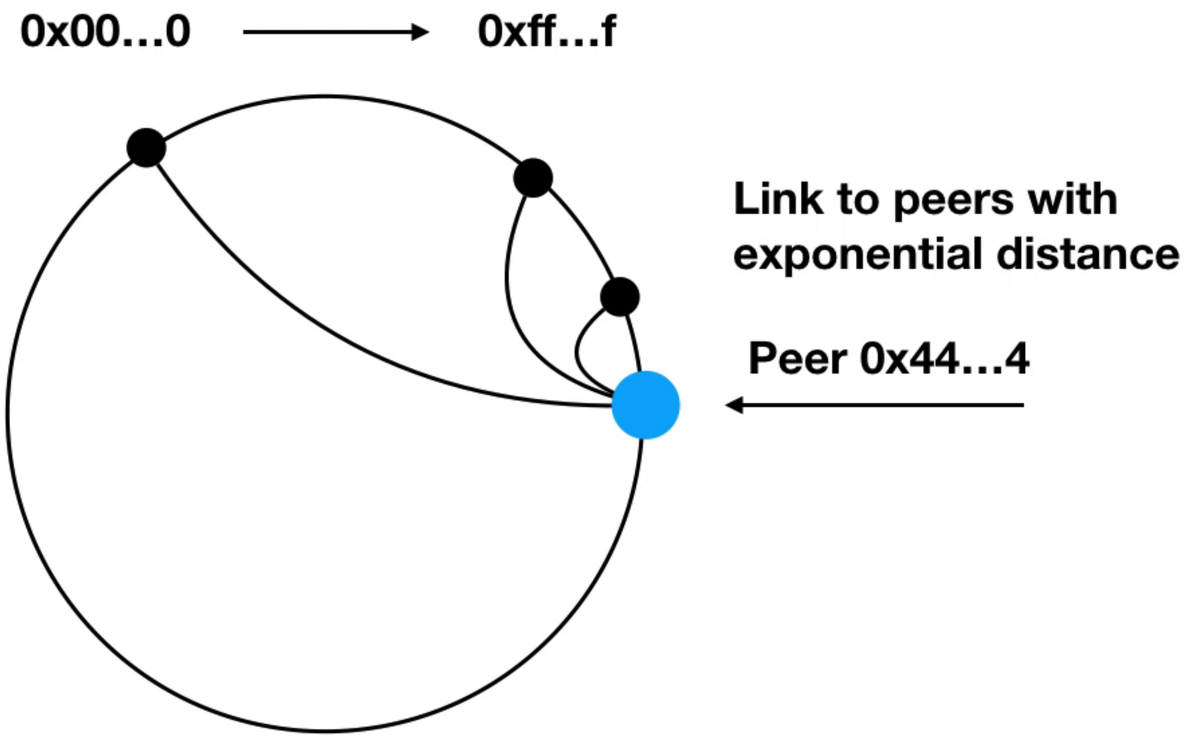
Kademlia DHT



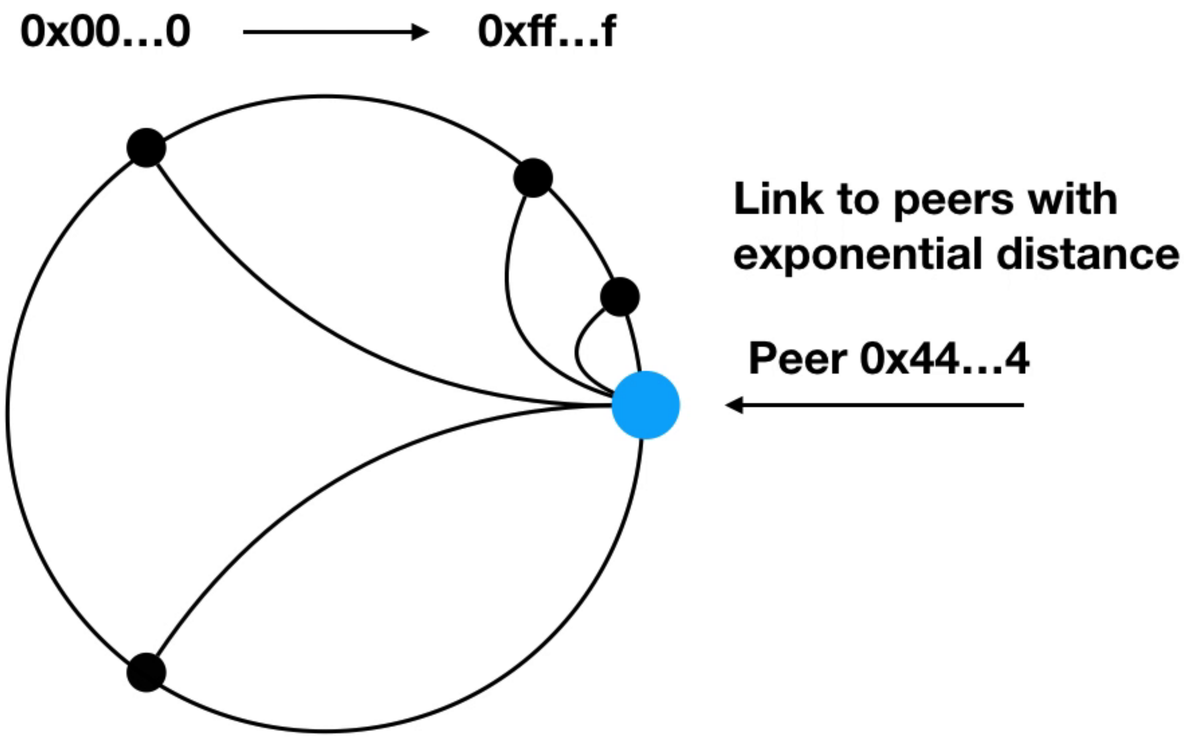
Kademlia DHT



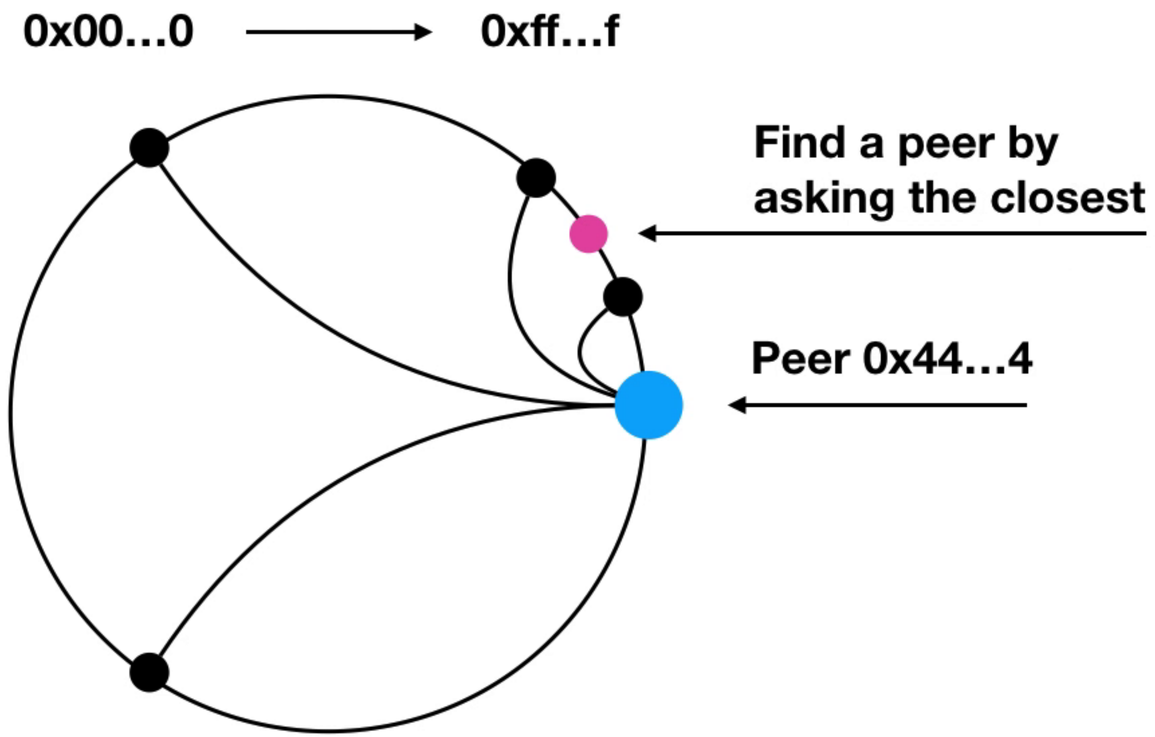
Kademlia DHT



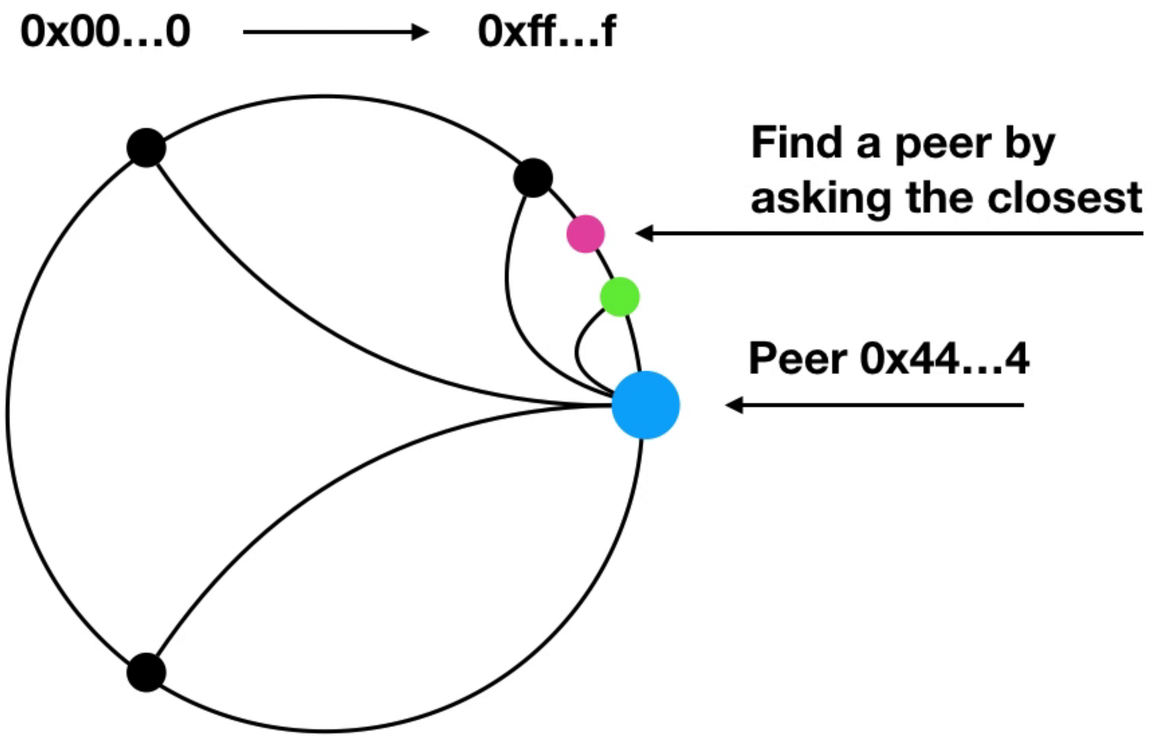
Kademlia DHT



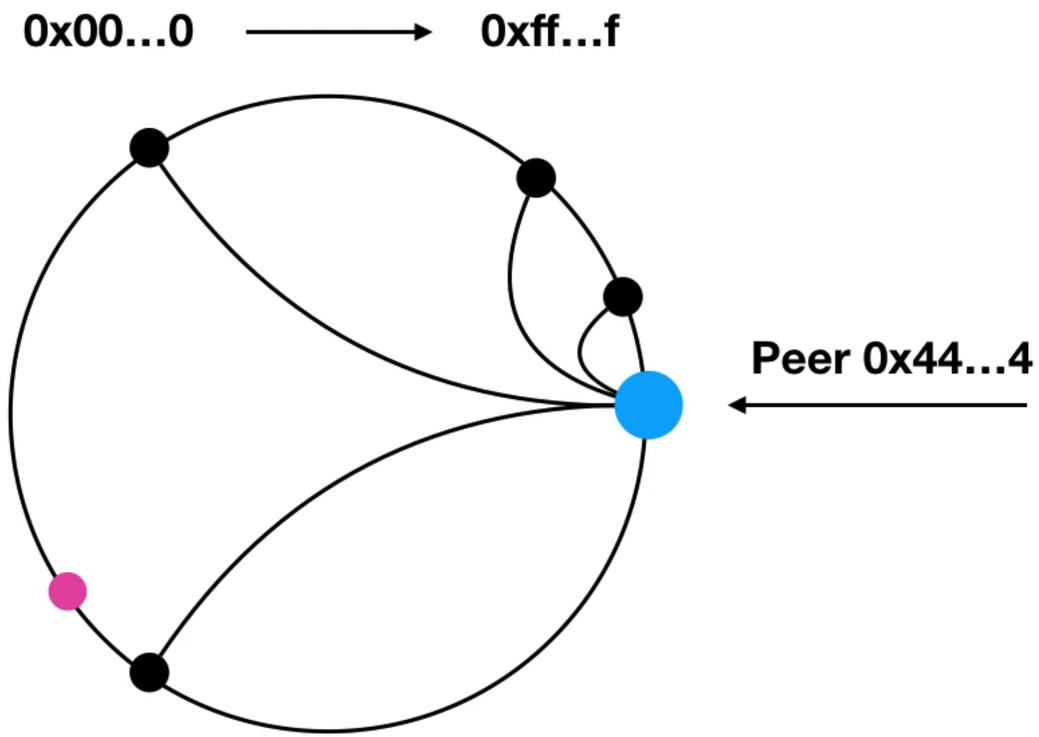
Kademlia DHT



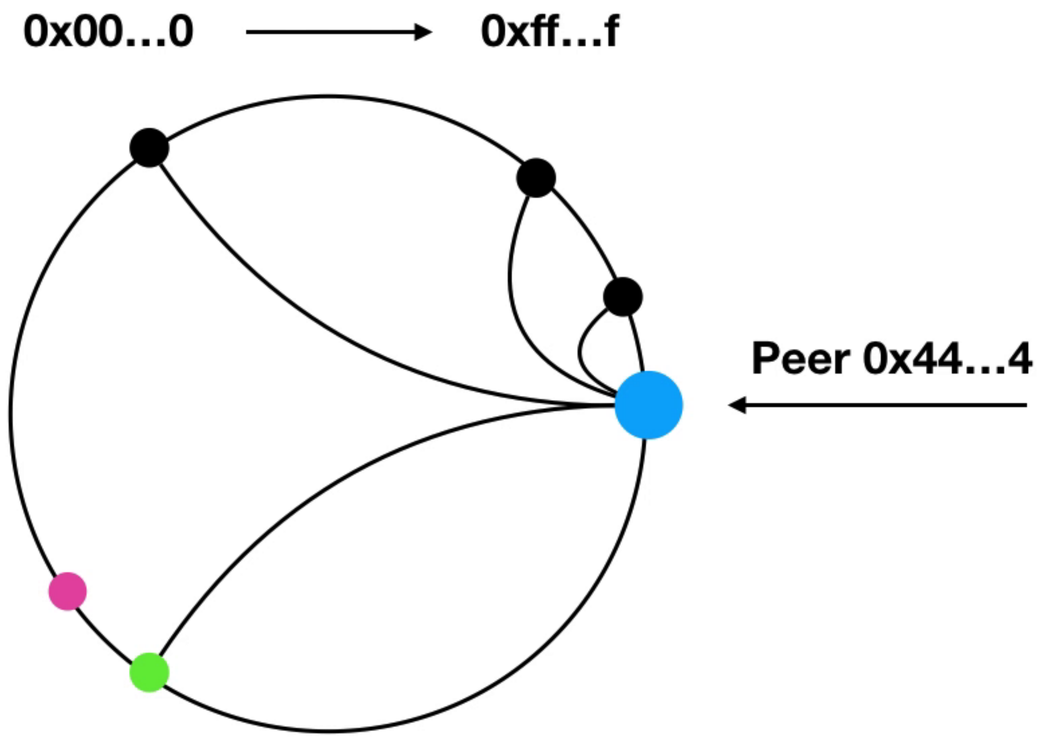
Kademlia DHT



Kademlia DHT



Kademlia DHT



replicate()

```
const topic = crypto.createHash('sha256').update('jsconf-todos').digest()
const swarm = hyperswarm()

const core = kappa(ram, { valueEncoding: 'json' })

core.writer('local', function (err, feed) {
    swarm.join(topic, { lookup: true, announce: true })
    swarm.on('connection', function (connection, info) {
        console.log("New peer!")
        pump(connection, core.replicate(info.client, { live: true }), connection)
    });
})
```

replicate()

```
const topic = crypto.createHash('sha256').update('jsconf-todos').digest()
const swarm = hyperswarm()

const core = kappa(ram, { valueEncoding: 'json' })

core.writer('local', function (err, feed) {
    swarm.join(topic, { lookup: true, announce: true })
    swarm.on('connection', function (connection, info) {
        console.log("New peer!")
        pump(connection, core.replicate(info.client, { live: true }), connection)
    });
})
```

replicate()

```
const topic = crypto.createHash('sha256').update('jsconf-todos').digest()
const swarm = hyperswarm()

const core = kappa(ram, { valueEncoding: 'json' })

core.writer('local', function (err, feed) {
    swarm.join(topic, { lookup: true, announce: true })
    swarm.on('connection', function (connection, info) {
        console.log("New peer!")
        pump(connection, core.replicate(info.client, { live: true }), connection)
    });
})
```

replicate()

```
const topic = crypto.createHash('sha256').update('jsconf-todos').digest()
const swarm = hyperswarm()

const core = kappa(ram, { valueEncoding: 'json' })

core.writer('local', function (err, feed) {
    swarm.join(topic, { lookup: true, announce: true })
    swarm.on('connection', function (connection, info) {
        console.log("New peer!")
        pump(connection, core.replicate(info.client, { live: true }), connection)
    });
})
```

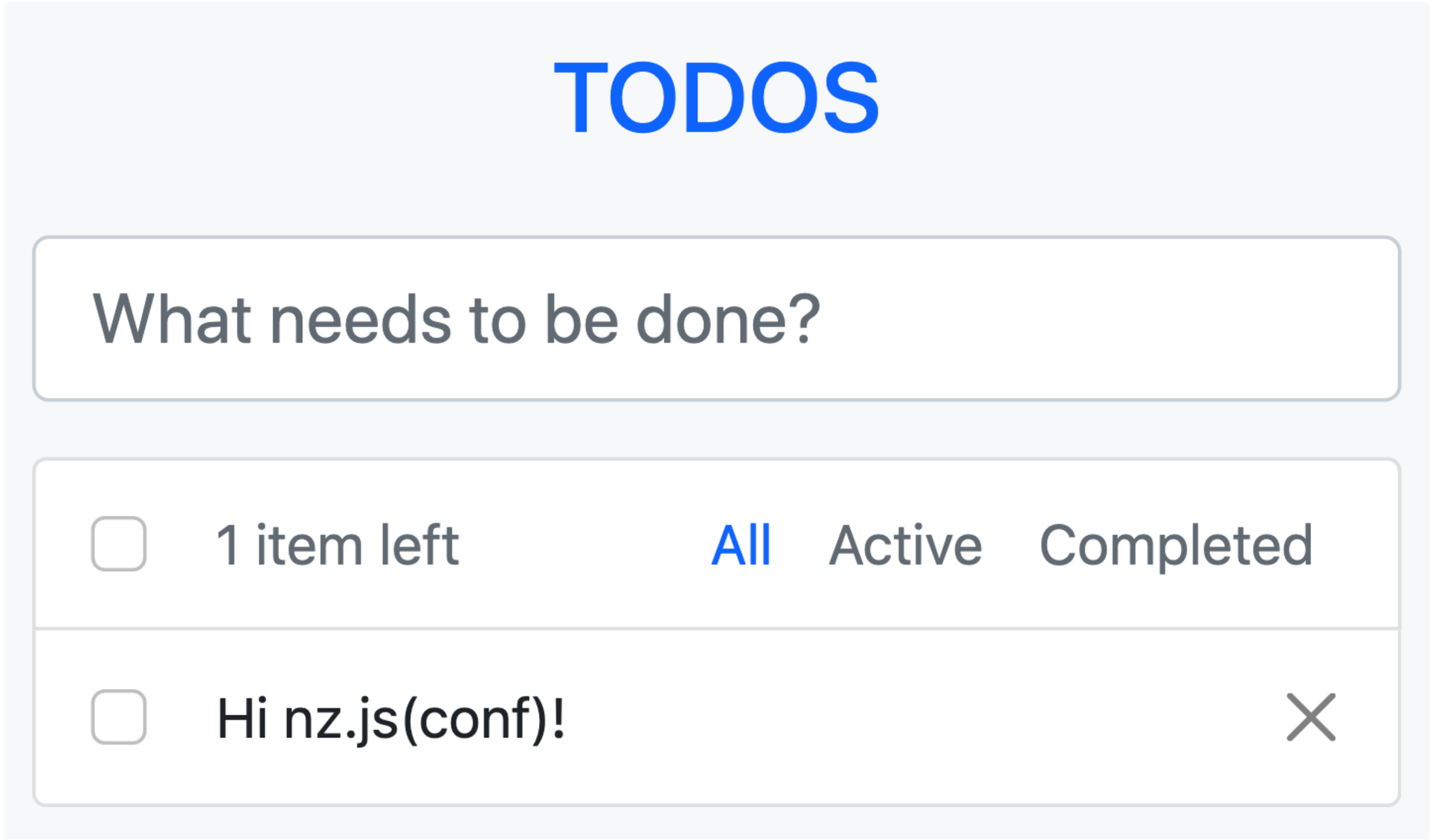
Listen for changes

```
// Listen for latest message.
core.api.records.on('batch', function (data) {
    for (let msg of data) {
        console.log(msg)
    }
});
```

In a browser?

```
npm i hyperswarm-web
```

- DHT over WebRTC
- Proxy Hyperswarm over websockets.



Let’s sync a todo list

store.js

```
import { createStore } from "vuex";
const state = { todos: [], };
const mutations = { ... };

const actions = {
  addTodo({ commit }, text) {
    commit("addTodo", {
      todo: {
        key: Date.now().toString(),
        text,
        done: false,
      },
    });
  },

  removeTodo({ commit }, todo) {
    commit("removeTodo", { todo });
  },

  toggleTodo({ commit }, todo) {
    commit("editTodo", { todo, done: !todo.done });
  },

  editTodo({ commit }, { todo, value }) {
    commit("editTodo", { todo, text: value });
  },
};

export default createStore({ state, mutations, actions, plugins, });
```

Connect kappa to Vuex

```
const plugins = [kappaPlugin];

export default createStore({ state, mutations, actions, plugins, });
```

Kappa Vuex Plugin

```
store.subscribe(({ type, payload: { todo } }, state) => {
  const todoKey = todo.key;
  todo = state.todos.find((t) => t.key === todoKey) || todo;
  feed.append({
    ...todo,
    type: type === "removeTodo" ? "del" : "put",
  });
});
```

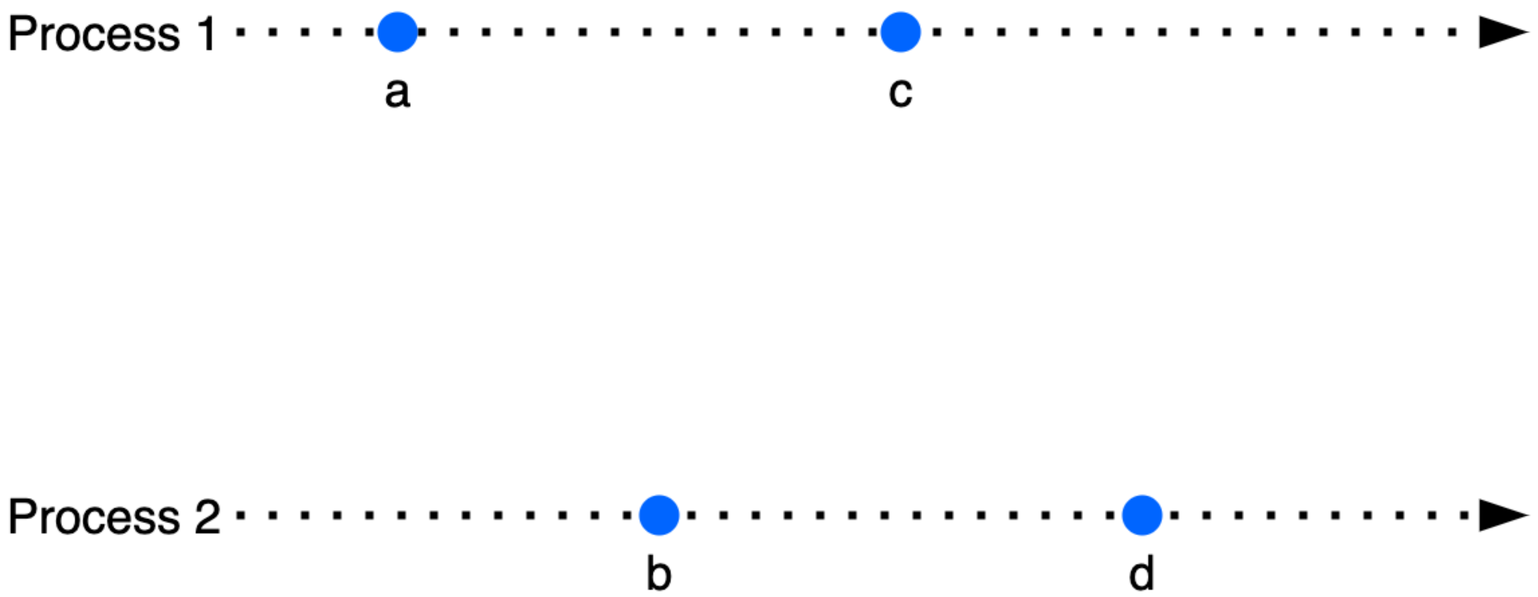
Kappa Vuex Plugin

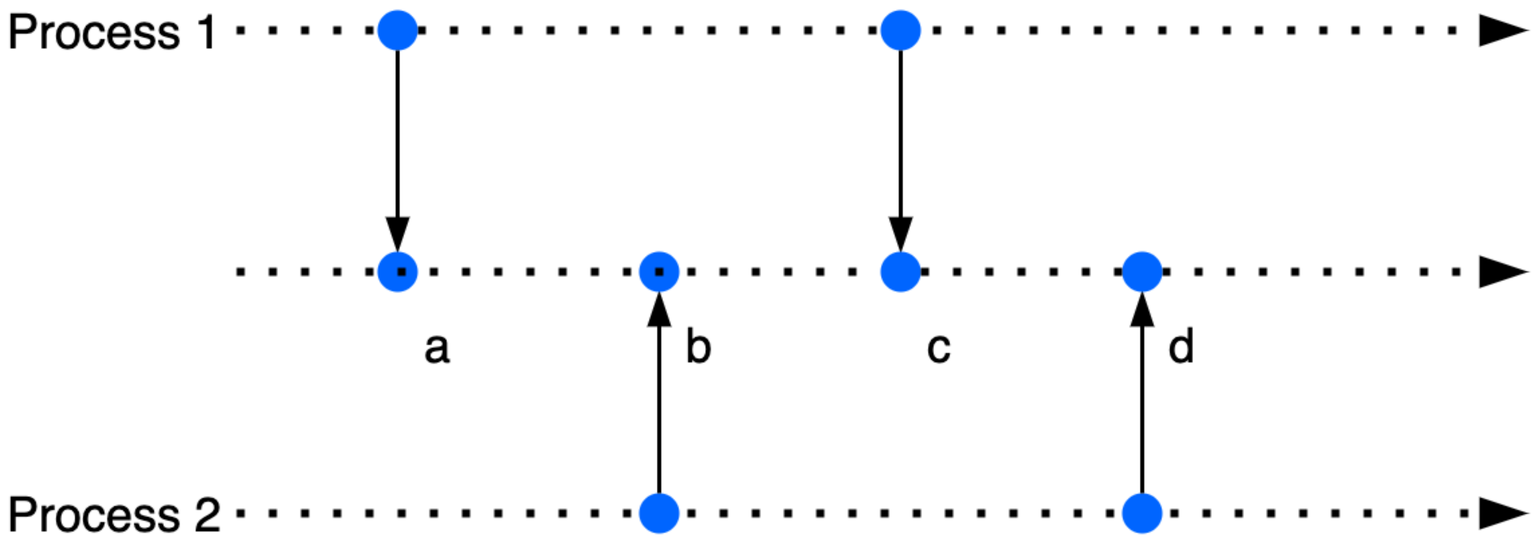
```
core.api.kv.on("batch", (ops) => {
  core.api.kv.all((data) => {
    store.commit(
      "receiveData",
      data.map(({ key, value }) => {
        return {
          key,
          ...value,
        };
      })
    );
  });
});
```

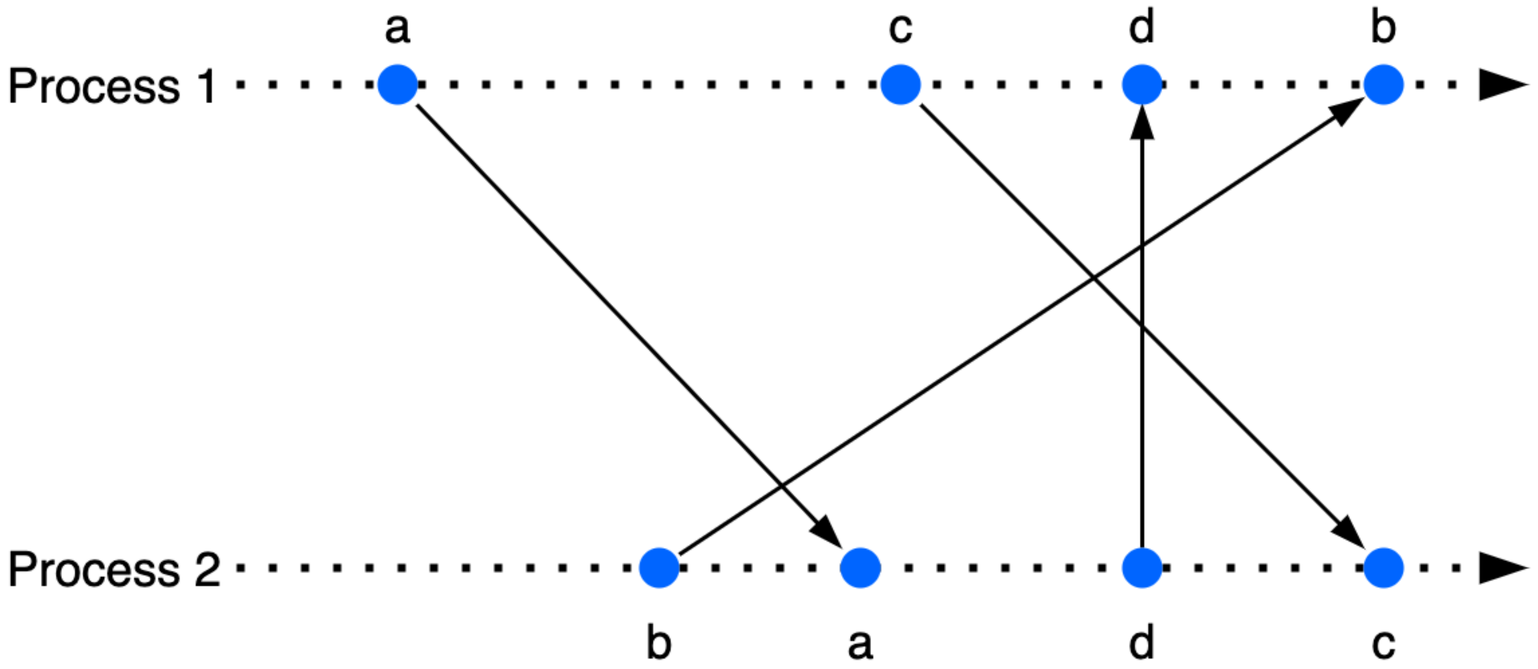
Local apps are a distributed system

Unreliable Ordering

Conflicts


Process 1
a
c
Process 2
b
d


Process 1
a
b
c
d
Process 2


a
c
d
b
Process 1
Process 2
b
a
d
c

Add timestamps

```
const clock = new HLC();

store.subscribe(({ type, payload: { todo } }, state) => {
  const todoKey = todo.key;
  todo = state.todos.find((t) => t.key === todoKey) || todo;
  feed.append({
    ...todo,
    type: type === "removeTodo" ? "del" : "put",
    ts: clock.now().toJSON(),
  });
});
```

Add timestamps

```
const clock = new HLC();

store.subscribe(({ type, payload: { todo } }, state) => {
  const todoKey = todo.key;
  todo = state.todos.find((t) => t.key === todoKey) || todo;
  feed.append({
    ...todo,
    type: type === "removeTodo" ? "del" : "put",
    ts: clock.now().toJSON(),
  });
});
```

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }

{ type: “put", text: “Do the thing!”, done: true, ts: 2 }

{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
```

```
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }
```

```
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

State

```
{ text: “Do the thing!”, done: false }
```

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
```

```
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }
```

```
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

State

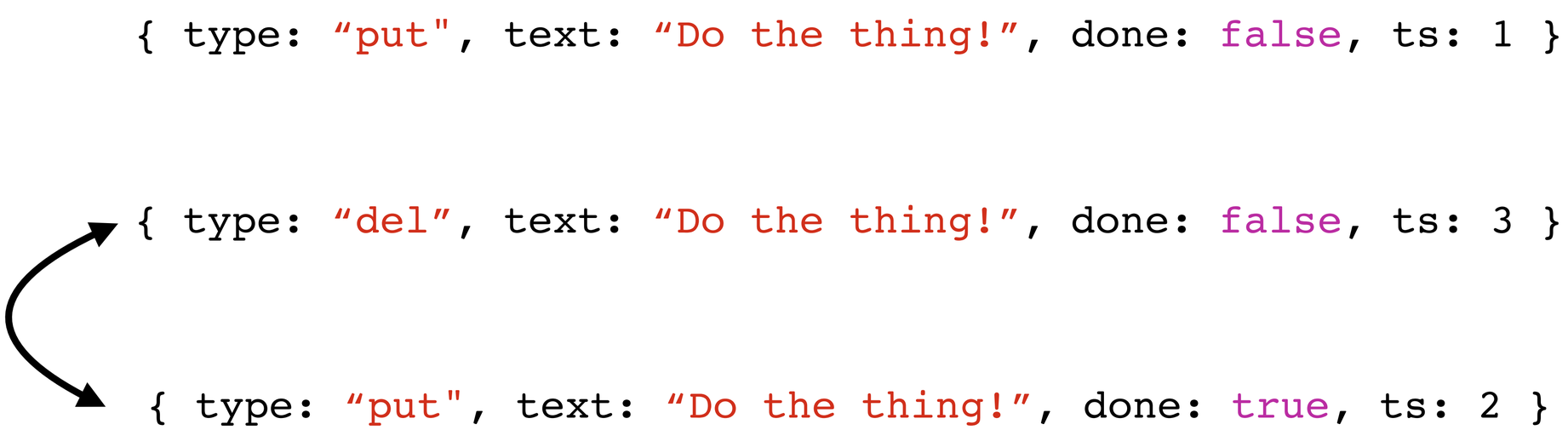
```
{ text: “Do the thing!”, done: true }
```

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
```

```
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }
```

```
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

State
Deleted


{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
```

```
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

```
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }
```

State

```
{ text: “Do the thing!”, done: false }
```

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
```

```
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

```
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }
```

State
Deleted

```
{ type: “put", text: “Do the thing!”, done: false, ts: 1 }
```

```
{ type: “del”, text: “Do the thing!”, done: false, ts: 3 }
```

```
{ type: “put", text: “Do the thing!”, done: true, ts: 2 }
```

State
Deleted

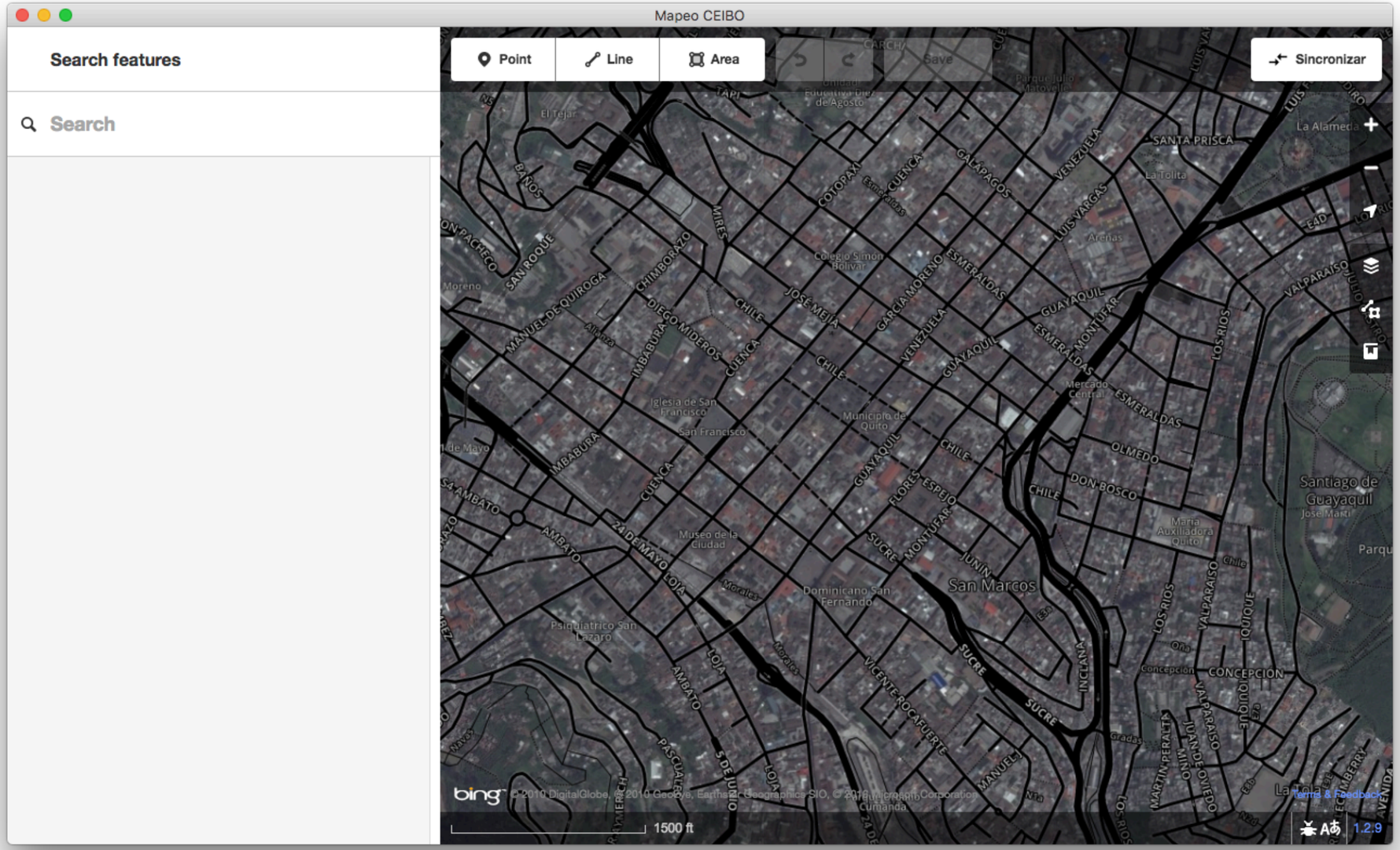
LWW (Last-Write-Wins) Map

Limitations?

Limitations

- We still need a signalling server or hyperswarm proxy
- Hyperswarm can’t always traverse NATs
- View doesn’t handle out-of-order messages
- Can’t sync if no-one else is on the network

Use-cases


Mapeo CEIBO
Search features
Search
Point
Line
Area
Sincronizar
Save
Santiago de Guayaquil
San Marcos
1500 ft
1:2.9

Mapeo


cooperative cloud

Summary

Can we share data and collaborate when offline?

Goals

- Can we view and edit data offline?
- Can we share data without internet?
- Can we share data without a central server?
- Can we share data with only a browser?

Goals

- Can we view and edit data offline? 
- Can we share data without internet?
- Can we share data without a central server?
- Can we share data with only a browser?

Goals

- Can we view and edit data offline? ✓
- Can we share data without internet? ✓
- Can we share data without a central server?
- Can we share data with only a browser?

Goals

- Can we view and edit data offline? ✓
- Can we share data without internet? ✓
- Can we share data without a central server? ✓
- Can we share data with only a browser?

Goals

- Can we view and edit data offline? ✓
- Can we share data without internet? ✓
- Can we share data without a central server? ✓
- Can we share data with only a browser? ✗

Thanks!

Links

- KappaDB Workshop https://kappa-db.github.io/workshop/build/01.html
- Hypercore https://hypercore-protocol.org/
- Kademlia DHT viz https://kelseyc18.github.io/kademlia_vis/basics/1/