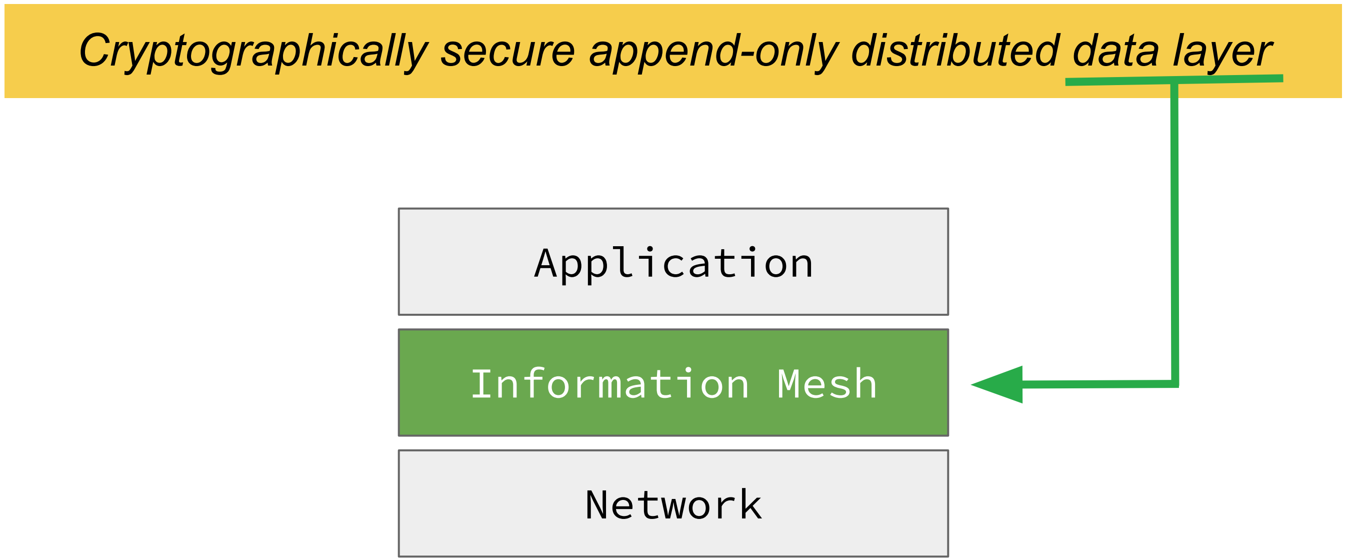
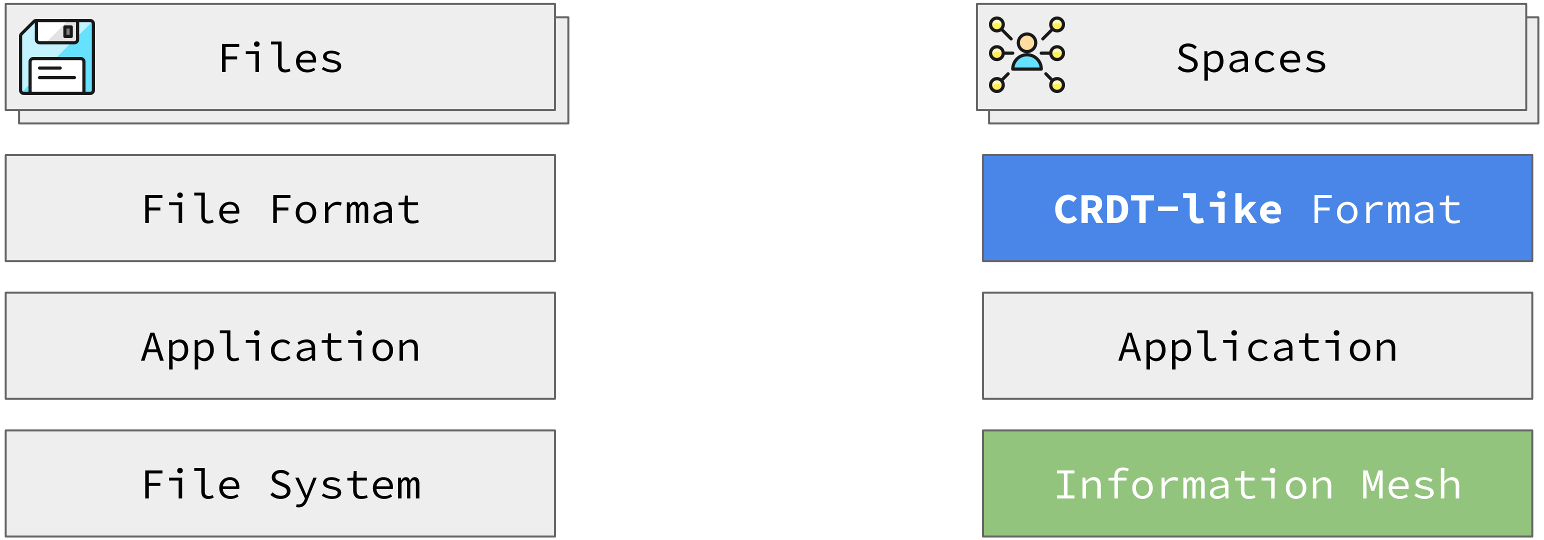
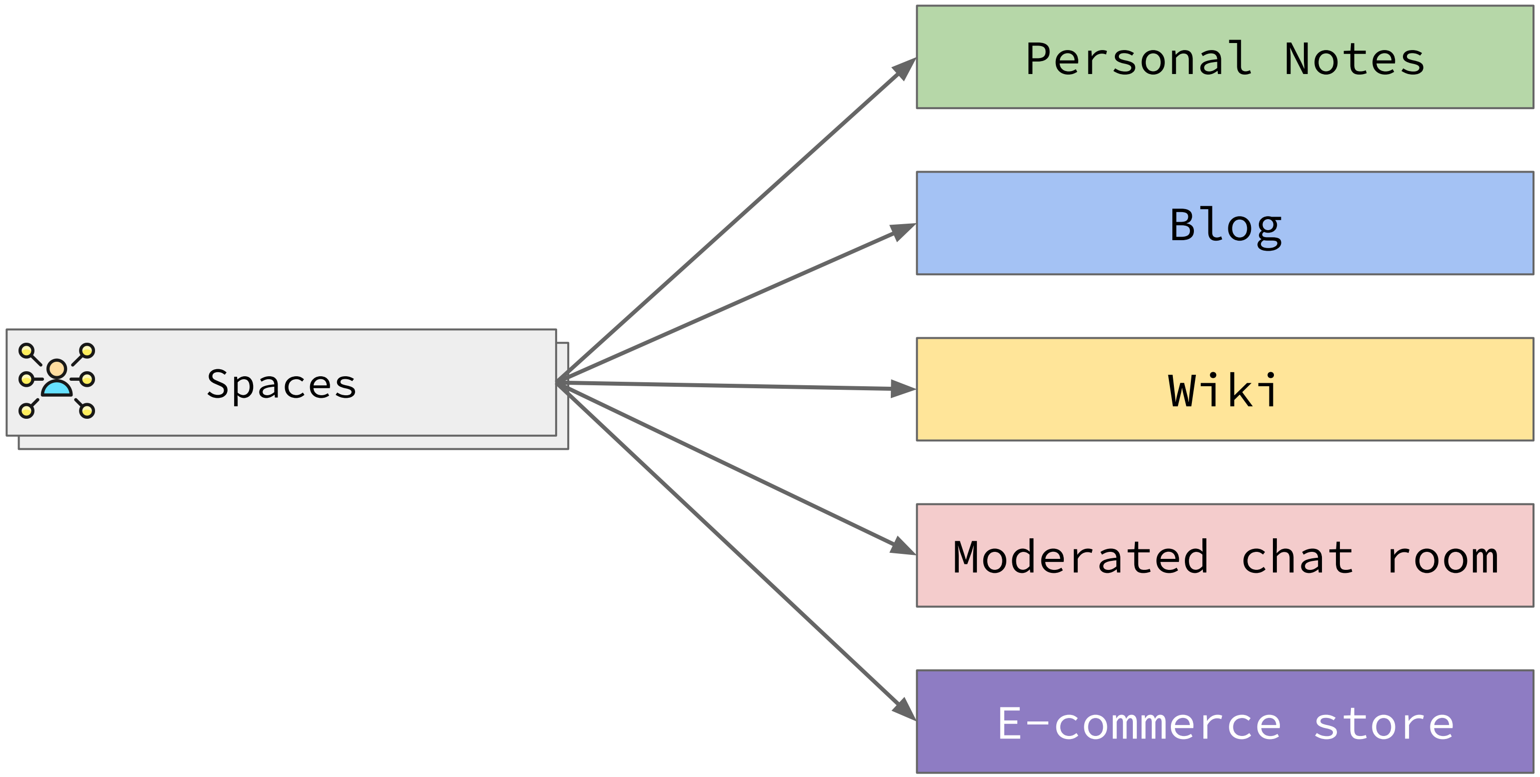
HYPERHYPER SPACE


Cryptographically secure append-only distributed data layer

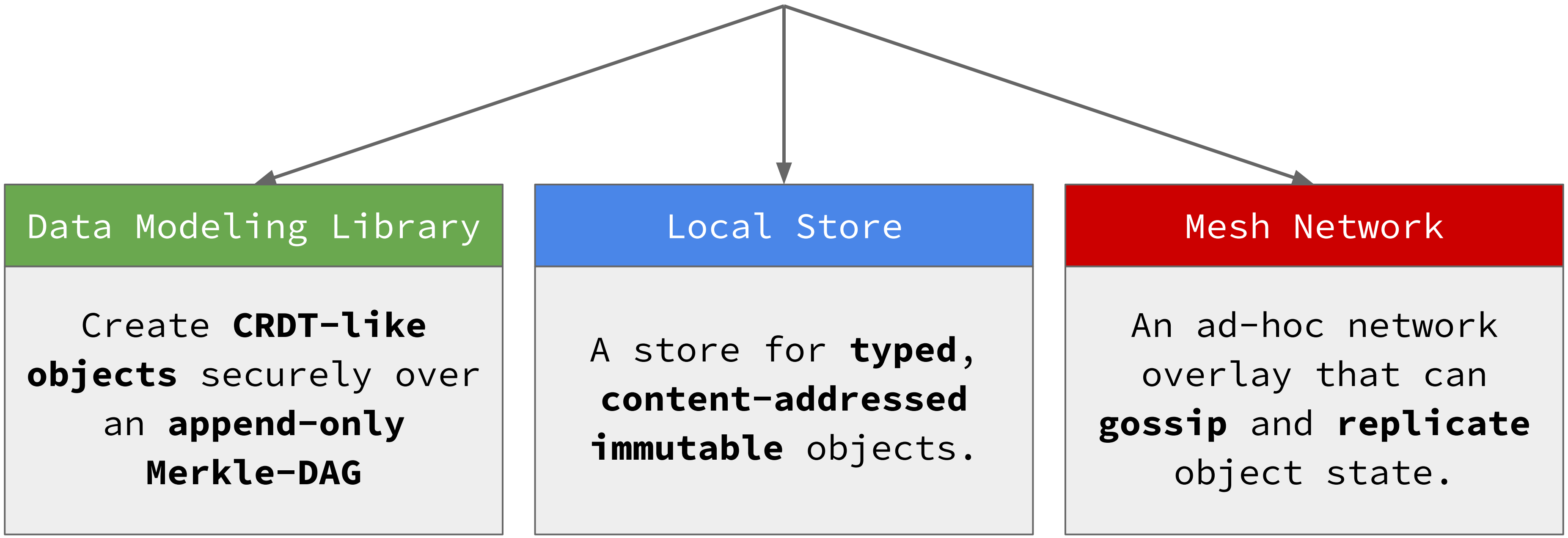
HYPERHYPER SPACE

We help people create **distributed collaborative apps**




Spaces
Personal Notes
Blog
Wiki
Moderated chat room
E-commerce store

HYPERHYPER
SPACE



Data Modeling Library

Create **CRDT-like objects** securely over an **append-only Merkle-DAG**

Local Store

A store for **typed, content-addressed immutable** objects.

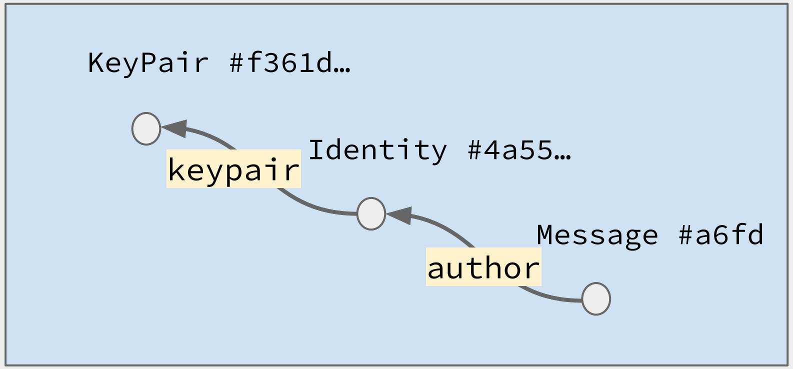
Mesh Network

An ad-hoc network overlay that can **gossip** and **replicate** object state.

Local Store

Typed, immutable objects.

Hash	Type	Value
#f361d…	KeyPair	{ public: '--BEGIN…
#4a455…	Identity	{ name: 'Santi', …
#a6fd2…	Message	{ text: 'hi', author: #4a455… }



Object's **content hash** used as id

JavaScript literal + hash-based references

Hash-based references between objects in the store form a DAG.

Data Modeling Library (i)

All immutable objects that will be **stored** extend **HashedObject**:

Example:

```
class Message
 extends HashedObject {
    author: Identity;
    content: string;
    timestamp: number;


    validate(): boolean {
 …
```

Like an ‘assert’, but paranoid 😱

Provides:

- Consistent **hashing**
- **Literalization**
- Replacing **object references** for hash-based ones
- Authorship / **signatures**

Requires:

- A **validate**() function, to be used by sync.

Data Modeling Library (ii)

All mutable objects are implemented as **op-based CRDTs**.

Sets, arrays, references, etc. are provided, other types may be implemented by extending **MutableObject** and **MutationOp**.

```
let s = new MutableSet();
store.save(s);
```

Hash	Type	Value
#66ad3…	Mutable Set	{ seed: ‘a53af…’ }

```
s.add(‘apple’);
s.add(‘orange’);
store.save(s);
```

#bb8c3…	AddOp	{ element: ‘apple’, target: ref<#66ad3…, **prevOps**: {} }
#39d46…	AddOp	{ element: ‘orange’, target: ref<#66ad3…, **prevOps**: {#bb8c3…} }

Data Modeling Library (iii)

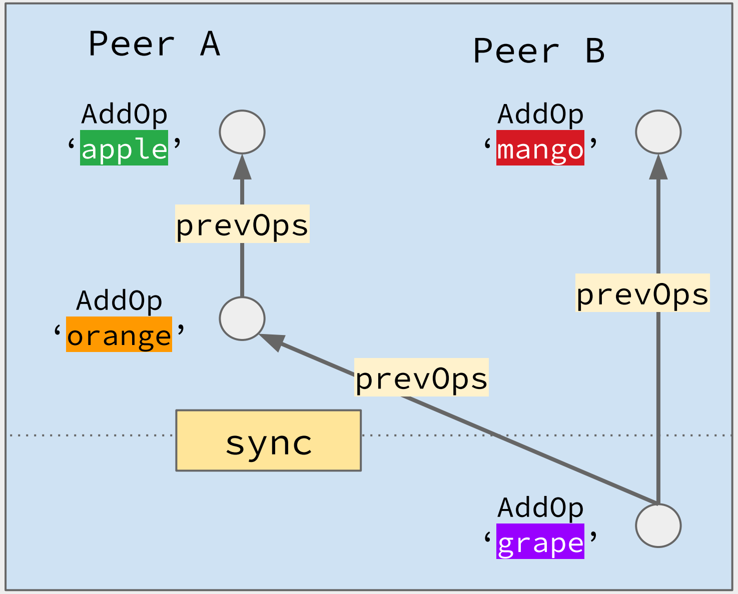
PrevOps defines **a partial (Merkle-ized) order** on the set of ops for a mutable object. The set of maximal elements univocally defines its state.

```
s.add('apple');
s.add('orange');
store.save(s);
```

```
s.add('mango');
store.save(s);
```

sync

```
s.add('grape');
store.save(s);
```



Data Modeling Library (iv)

Challenge: need a way to express more complex types / invariants!

A **moderated** chat group type

```
class ChatGroup
 extends HashedObject {
  owner        : Identity;

  moderators : Set<Identity>;
  members      : Set<Identity>;

  messages     : Set<Message>;
  …
```

Rules:

* Only members can post messages.

* Moderators are designated by the owner.

* Members can delete their own messages, while moderators can delete other’s. …

Data Modeling Library (v)

Challenge: need a way to express more complex types / invariants!

* Members can delete their own messages, while moderators can delete other’s. …

Alice is removed from the set of moderators

Alice deletes a message from **Bob**, using her moderator rights.

Those two operations need to commute!

Data Modeling Library (vi)

Increase the **expressive power** by adding **explicit causal relationships** and **cascaded operation invalidation**.

Example: CausalSet

Causal sets have three operations:

- **Add** an element
- **Delete** an element (*)
- **Attest** that element is in the set

(*) and record where this happens in the local copy of op. history

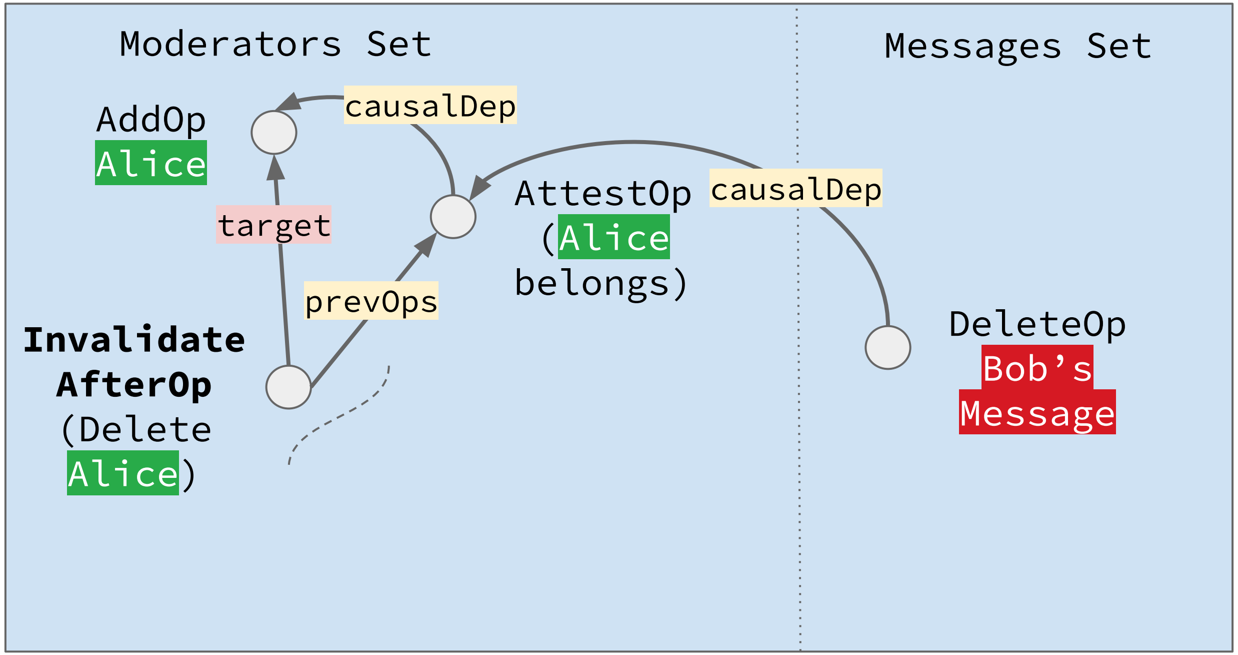
Make 'moderators' and 'messages' causal sets!

Now alice needs to **attest** that she belongs to the moderators set in order to delete Bob's message.

Her deletion of Bob's message will be causally dependent on that attestation.

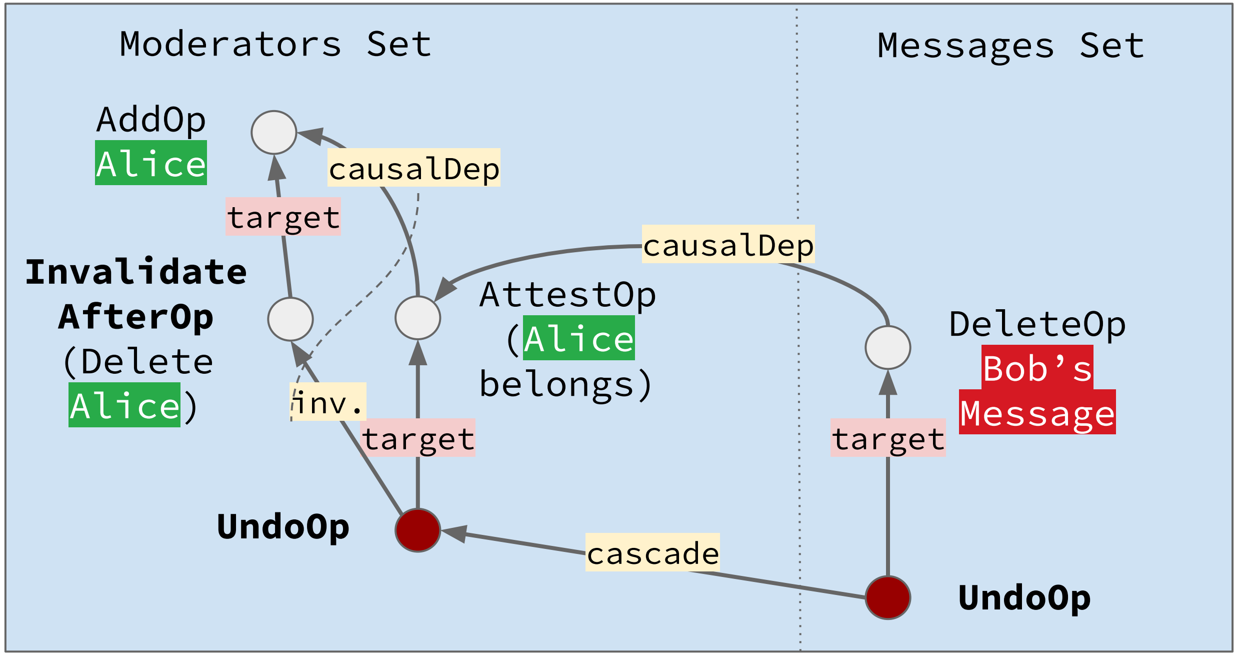
Data Modeling Library (vii)

The **prevOps** field in InvalidateAfterOp indicates the **attestation was present** when it was generated, **hence it is valid**.



Data Modeling Library (viii)

The **attestation was not present** when InvalidateAfterOp was generated, **hence it is undone**, alongside all its causal deps.

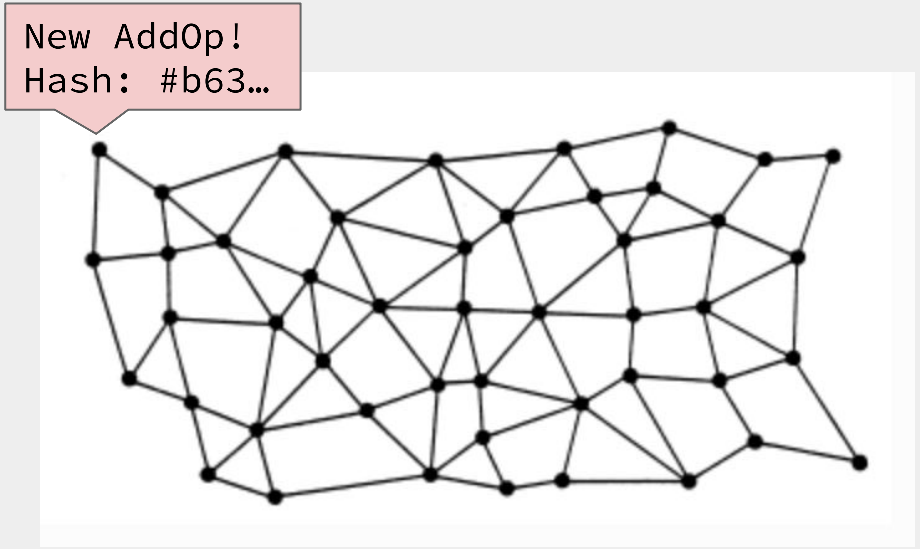


Data Modeling Library (ix)

Summary

- Represent data as content-addressed immutable typed objects, that cross-reference each other using their hashes (**DAG**).
- Provide **validators** for all objects.
- Use op-based **CRDTs** for mutability, use local history to **partially order operations**.
- Use explicit causal dependencies and cascaded invalidation to enable **composition of datatypes**.

Mesh Network



The mesh is organized in **Peer Groups** that want to sync (approx) the same set of **MutableObjects**.

Peer **sourcing** is application-defined, could be almost anything (a torrent-like file, dynamic discovery, a set inside H.H.S.)

Gossip: the state of each MutableSet can be expressed as the hash of the set of 'maximal' ops (as per the defined partial order). **This hash is gossiped.**

Sync: operation headers are requested (when gossip so indicates) to allow a performant and resilient **streaming replication algorithm.**

The end

www: https://www.hyperhyperspace.org

white paper: https://www.hyperhyperspace.org/whitepaper

demo: https://hyperhyper.space

sources:

https://github.com/hyperhyperspace/hyperhyperspace-core

https://github.com/hyperhyperspace/chat-group

Thanks !