Workspace
Local-first Collaborative Editing over NDN
Tianyuan Yu, Xinyu Ma, Varun Patil
Local-First Software

- Presented by Martin at last DINRG meeting
- Local-first software provides
  - Availability when offline
  - Direct user communications
  - End-to-end security and encryption
- CRDT merges concurrently changes from peers
- Workspace is a local-first collaboration application
- This talk shares development experience and lessons learned
Scenario

- Suppose Tianyuan and Xinyu are writing a paper together
- We use collaborative apps to resolve conflicts
Scenario

- However, when we are both on an airplane and do not have access to the Internet
- What can we do?
Scenario

- However, when we are both on an airplane and do not have access to the Internet
- We need a new collaborative editing app that
  - Allows access data offline
  - Users directly exchange secured data
In an open environment, how can a device trust received data?

*Direct user-to-user security*
User Identity

- Requirements:
  - Unique: one identifier one user
  - Verifiable: verify ownership in the internet identification space
  - Semantic meaningful
    - Users need to know who they are talking to, beyond key hashes

- Semantic identifiers come from today's Internet
  - Semantically identifying an entity: DNS name owner, organization, etc
  - Email addresses: xinyu.ma@cs.ucla.edu
  - Platform usernames: tianyuan129 in github.com
Direct User Data Security

- Identifiers cannot stand alone, need binding with key pairs
- The workspace owner authenticate users by identifiers
  - Then issues a certificate to the key-identifier binding
    - /workspace/xinyu.ma@cs.ucla.edu/KEY/v1
    - Signature: by /workspace/KEY/v1

The Trust Anchor
/workspace/KEY/1

An Author's Cert
/workspace/xinyu/KEY/1

A change
/workspace/xinyu/1

An Author's Cert
/workspace/tianyuan/KEY/1

A change
/workspace/tianyuan/1
How does the Workspace work?

Demonstrate *data-centric communication* by an example
Data Model: Event Sourcing

- **Mutable**
- **Multiple authors**
- **Viewed in the frontend**

---

**Changes**

- **Securely** tracking who did what
- **Sync**: when getting same set of changes => same version

---

**The Document**

```latex
\documentclass[12pt]{article}
\usepackage{lingmacros}
\usepackage{tree-dvips}
\begin{document}
\section*{Section 1}
\end{document}
```

<table>
<thead>
<tr>
<th>Change</th>
<th>Author</th>
<th>Insert</th>
</tr>
</thead>
<tbody>
<tr>
<td>xinyu/1</td>
<td>Xinyu</td>
<td>&quot;docu\ldots&quot;</td>
</tr>
<tr>
<td>tianyuan/1</td>
<td>Tianyuan</td>
<td>&quot;sect\ldots&quot;</td>
</tr>
<tr>
<td>xinyu/2</td>
<td>Xinyu</td>
<td>&quot;end\ldots&quot;</td>
</tr>
</tbody>
</table>
Eventual Consistency: CRDT

- **Sequentialize** changes to solve conflicts
- Mature algorithms and implementations

The Document

<table>
<thead>
<tr>
<th>Change</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xinyu/1</td>
<td>Xinyu</td>
<td>Insert: “\docu...”</td>
</tr>
<tr>
<td>tianyuan/1</td>
<td>Tianyuan</td>
<td>Insert: “\sect...”</td>
</tr>
<tr>
<td>xinyu/2</td>
<td>Xinyu</td>
<td>Insert: “\end{...}”</td>
</tr>
</tbody>
</table>
When I make a change …

CRDT captures this edit and outputs a new **change**

Insert: “Add some text.”
Pos: 118
(Internal data…)
Exchanging **Semantically Named, Secured Data**

Insert: “Add some text.”  
Pos: 118  
Author: Xinyu  
(Internal data…)

Sign by  
/workspace/xinyu/KEY/1

/workspace/xinyu/3  
Insert: “Add some text.”  
Pos: 118  
Author: Xinyu  
(Internal data…)

Publish  
Sync

Save  
Browser’s storage

OPFS
Apply the change

Sync

Subscribe

Save

CRDT

CRDT computes the new document based on the new change

Tianyuan’s side

Browser’s storage

/workspace/xinyu/3

Insert: “Add some text.”
Pos: 118
Author: Xinyu
(Internal data...)

Frontend Update

```latex
\documentclass[12pt]{article}
\usepackage{lingmacros}
\usepackage{tree-dvips}
\begin{document}
\section*{Section 1}

Add some text.

\end{document}
```
How do users exchange data?

NDN Testbed for rendezvous
Rendezvous in the air

Centralized Server

NDN
NDN Testbed to support named data communication

- Infra for exchanging named data efficiently
  - Does not (need to) see or understand data
  - Not specific to application

- Multi-continent network **ready for use** today
  - Run by volunteer organizations
  - Anybody (you) can join!
  - Using the testbed is (very) easy

https://101.named-data.net/testbed/
Getting notified for new data

Alice:
I've new data!
/alice/new

Bob:
I need it!
(sends interest)

Cathy:
Don't need that one

Dan:
Can't hear...
Sync – NDN Transport

- “Distributed Dataset Synchronization”
- Synchronize the set of produced data names
  - Producers serve their own data
  - Network lets consumers fetch data directly
- State Vector Sync Protocol

https://named-data.github.io/StateVectorSync/
https://github.com/named-data/ndn-svs
CRDT over NDN Sync

- Versioning: Sync state
- Updates as patches
  - CRDT deltas wrapped as Sync data
- Subscriptions: joining a Sync group
- Merge Types: CRDT
But users may not be online simultaneously

Asynchronous communication with Repo, a generic in-network storage
In-network Data Storage

- Generality as network service
- Agnostic to application security
- Does not see or understand the stored data
Summary
Basic Components of Workspace

- Semantic user identifiers originating from today’s Internet
  - Unique, verifiable, and semantic meaningful
- Data signature and end-to-end encryption
- A decentralized network that exchange secured data

The high-level design is generic and applicable to other apps (e.g., chat)
Low Entry Bar for Developers

- An undergrad from Shanghai, Xinchen, felt Workspace is interesting
  - With little knowledge about NDN, security, or the NDN testnet.
  - Spent a few days of learning CRDT libraries

- Developed a chat app for Workspace very easily
  - One week: coding, pull request, code merged
  - Yjs + a bit front end work

- Workspace provides API to build local-first, decentralized app with security built-in
  - Write app just as writing plain CRDT apps
    - Make changes on shared data structures
  - Workspace backend exchanges secured CRDT updates
Takeaways

- Fundamental building blocks of decentralized app
  - Security, Sync, Storage
    - Users need to have semantic identifiers
    - Data-centric security enables secure communication independent of the channel
    - Semantically named and secured data objects allow generic network services
      - Synchronization and data storage

Workspace URL: https://ndn-workspace.web.app/

Contributions are welcome