NDN-RTC and Experimental Library Functionality

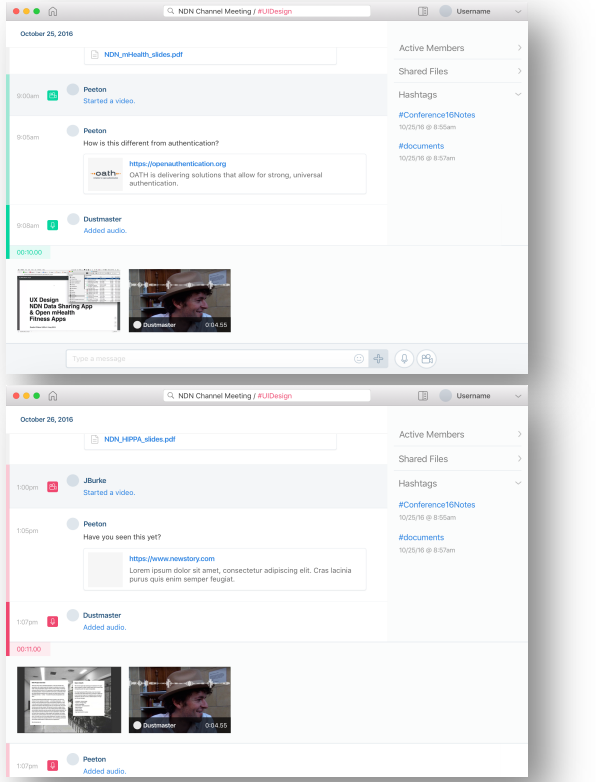
Peter Gusev

NDNComm, March 2017

Flume

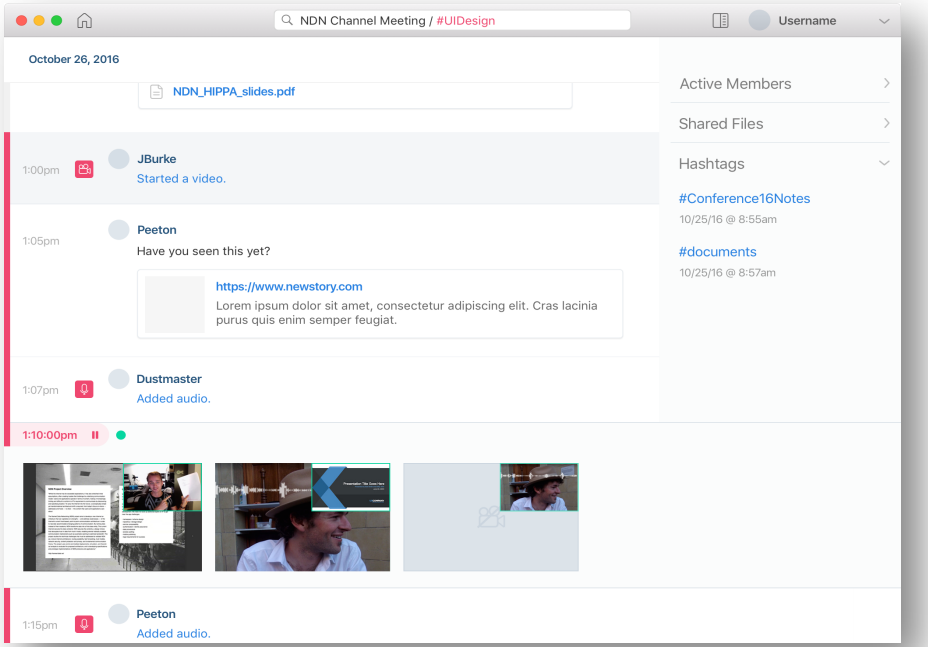
Slack channels + Skype with DVR

- Conversational group channels
- Publish audio/video within the text chat context
- Seamless DVR-like playback and access to historical data

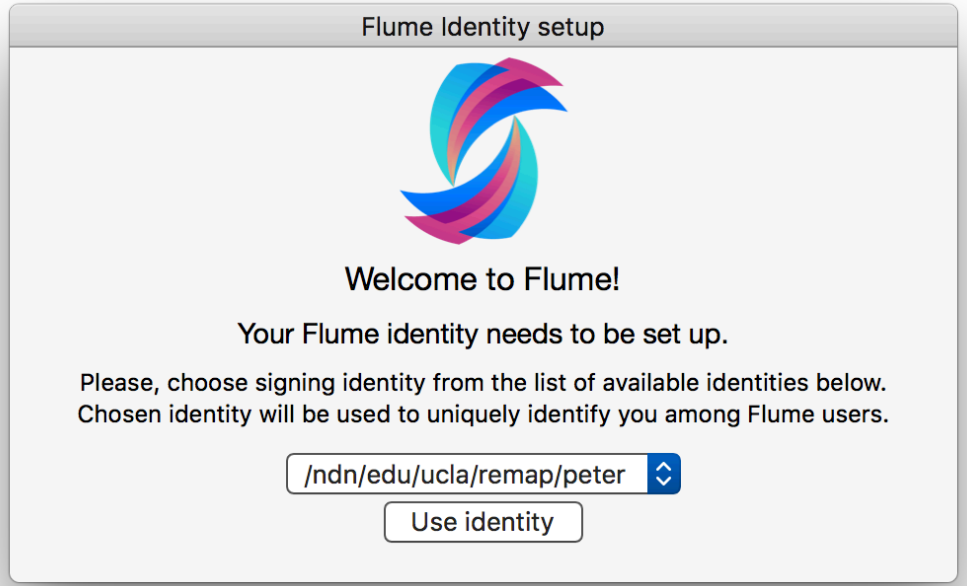


Another app?

- Continuation of NDN-RTC/*ndncon* work
- Useful tool
- Focusing on:
  - peer-to-peer scenarios
  - mundane resilience
  - data mules
  - mobile producers



Application identity setup

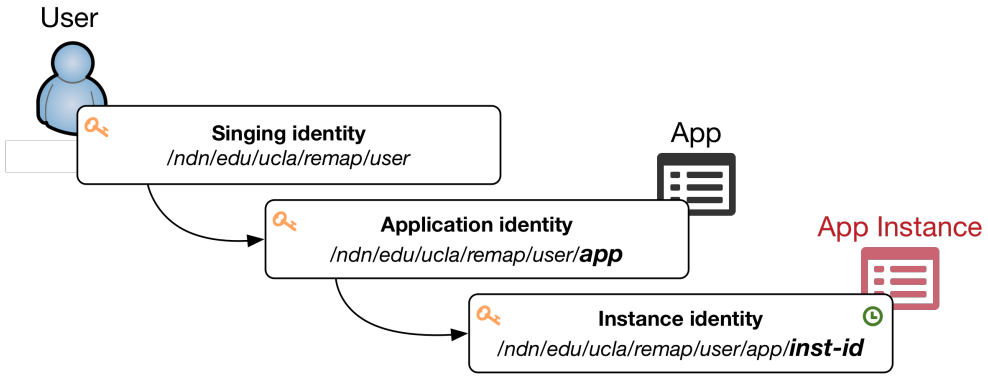


Incorporating schematized trust

- Namespace update
- Hierarchical verification
- Using short-lived (1h) keys for actual data

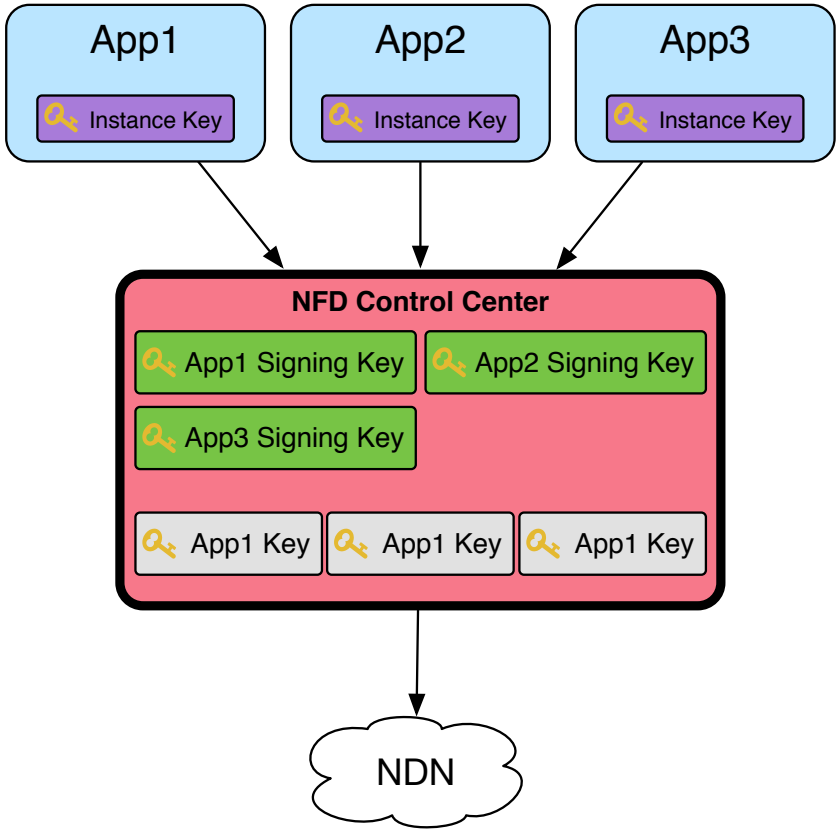
/ndn/edu/ucla/remap/ **app** / **user**

/ndn/edu/ucla/remap/ **user** / **app**



NDN applications ecosystem

- NFD Control Center
  - required platform for NDN applications
  - stores and serves application certificates
  - provides API for applications:
    - to request signing identity
    - to store application identity
    - connectivity check



Local connectivity

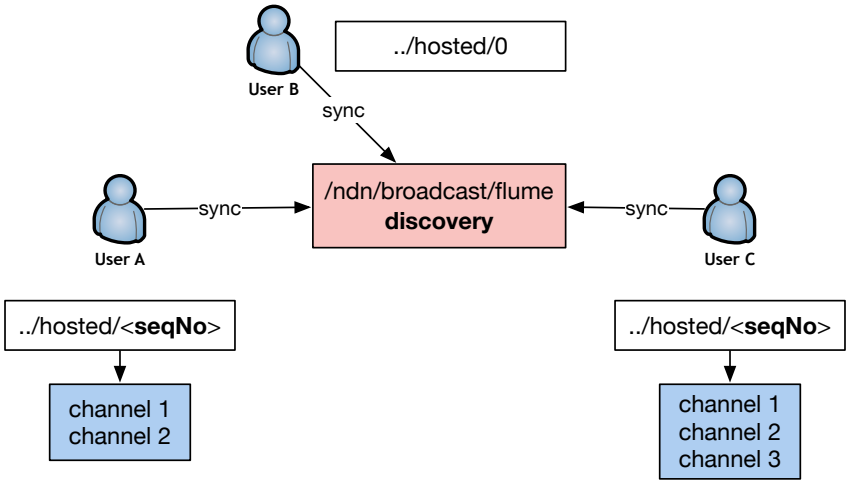
- Goal: *enable NDN apps to operate locally, without requiring testbed connection*
- Establishing local peer-to-peer connectivity in common adhoc scenarios: Local WiFi, LAN
- Two modules:
  - **Discover:** discovering nearby peers (WiFi direct, Bonjour, Bluetooth,…)
  - **Routes management:** establish NDN routes between discovered peers

Experimental Library Functionality

- Shim between NDN-CPP and application (Flume)
- Experimental concepts and ideas, potentially can be generalized for common use in NDN-CPP or NDN-CNL

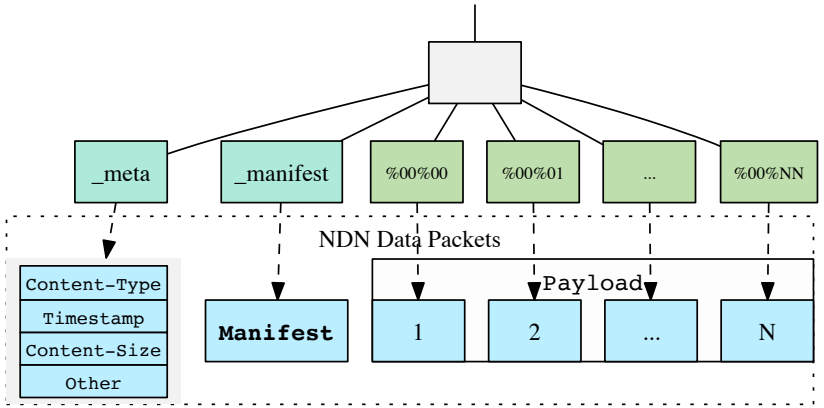
Channel discovery

- Global “discovery” sync object
- User publishes list of channel IDs
- Channel ID list numbered sequentially (ChronoChat message)



Generalized object fetching

- **_meta** describes the object:
  - MIME content type
  - timestamp
  - content size
  - other
- **payload** is segmented
- **_manifest** for verification



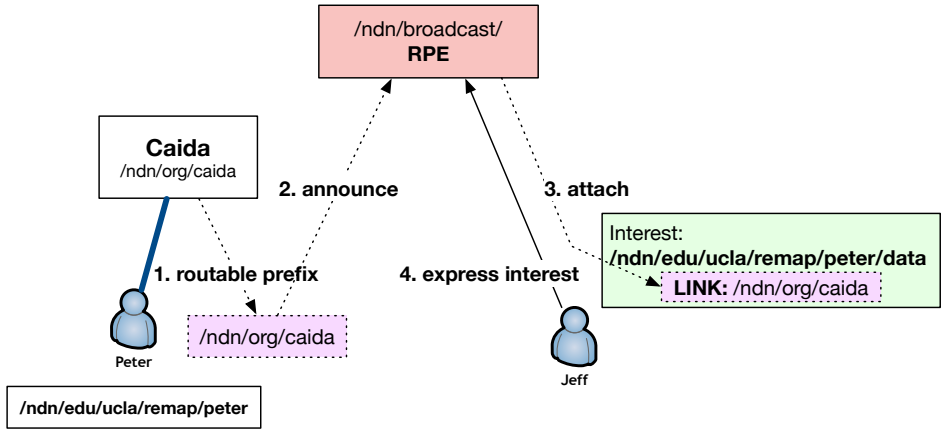
Abstracting Interest/Data exchange

- Asynchronous callback-based API
- **Sync** object into a namespace:
  - **publish**, if namespace is local
  - **fetch,** if namespace is remote
- Namespace scoping
- Namespace subtrees

```
LocalNamespace n(appPrefix, face, keyChain);
Namespace profile = n.scope("profile");
NamespaceObject o("data");

o.setPayload("{ username: Peter; bio: 'UCLA REMAP'; }");
profile.sync(o,
             [](NamespaceObject& o){
                 // published "<appPrefix>/profile/data"
              },
              [](NamespaceObject& o, std::string errMsg){
                 // handle error
              });

...

RemoteNamespace n(appPrefix, face, keyChain);
Namespace profile = n.scope("profile");
NamespaceObject o("data");

profile.sync(o,
             [](NamespaceObject& o){
                // fetched "<appPrefix>/profile/data"
             },
             [](NamespaceObject& o, std::string errMsg){
               // handle error
             });
```

Routable prefix exchange

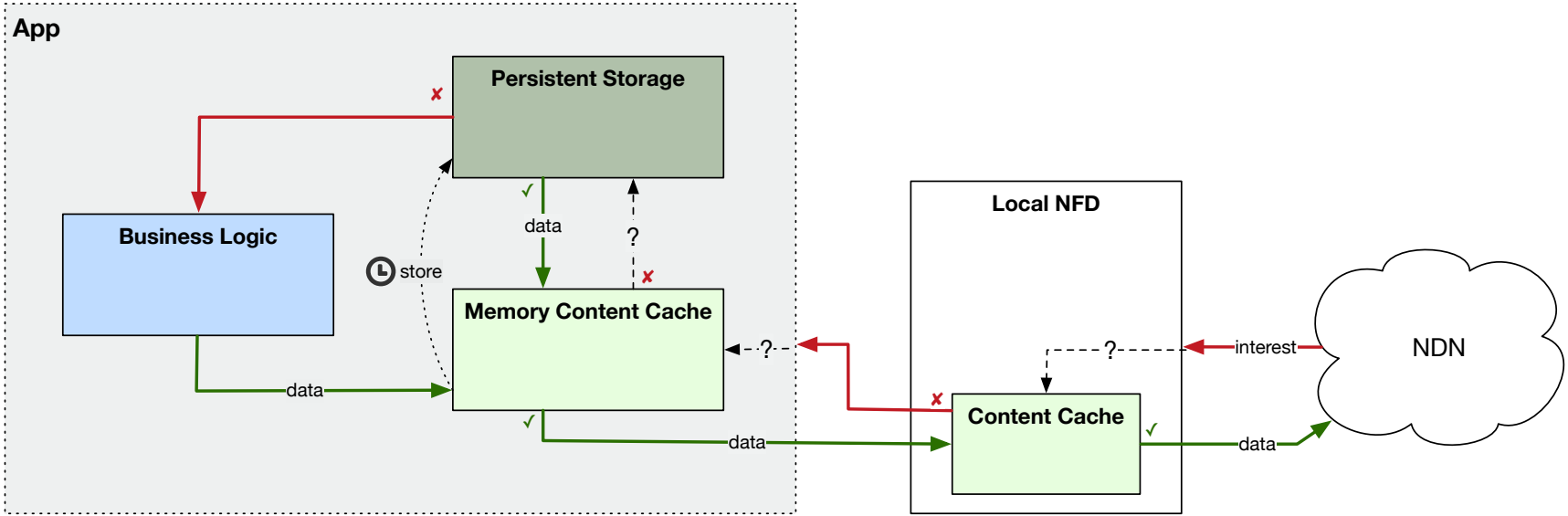
- Using ChronoSync for prefix exchange
- Happens in background
- Discovered prefixes attached as LINK objects to Interests



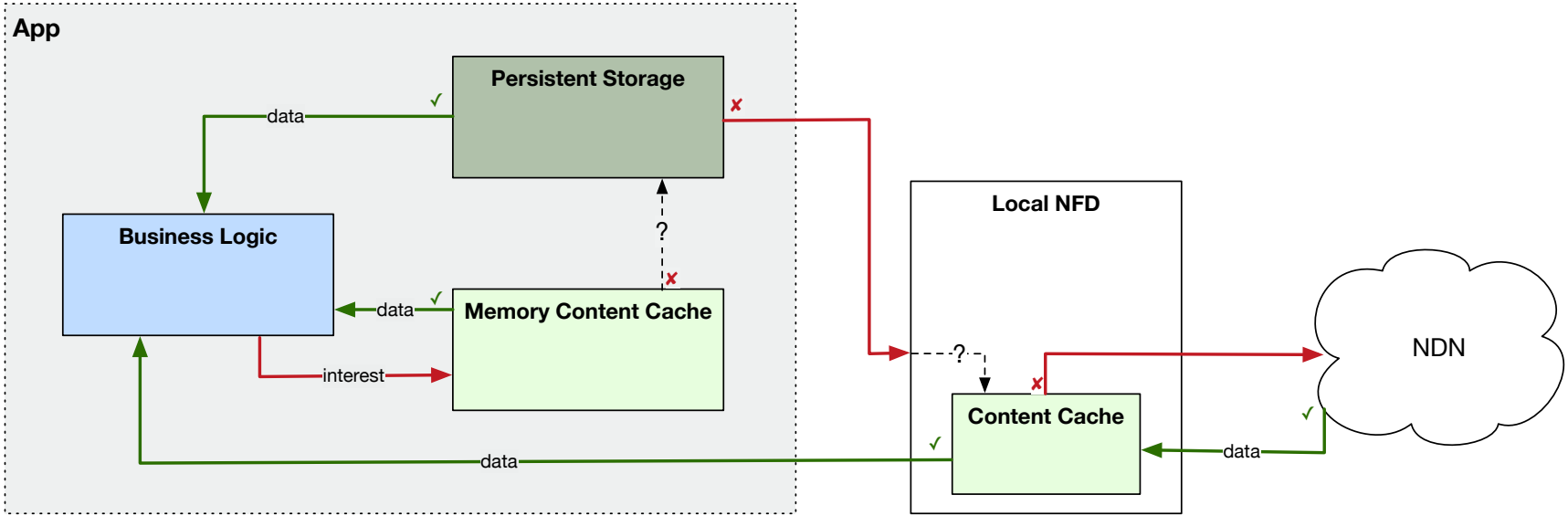
Storage

- Persistence across multiple app launches
- Storage rotation scheme
- Handles high frequency interests (real-time requirement)
- Store other peer's data

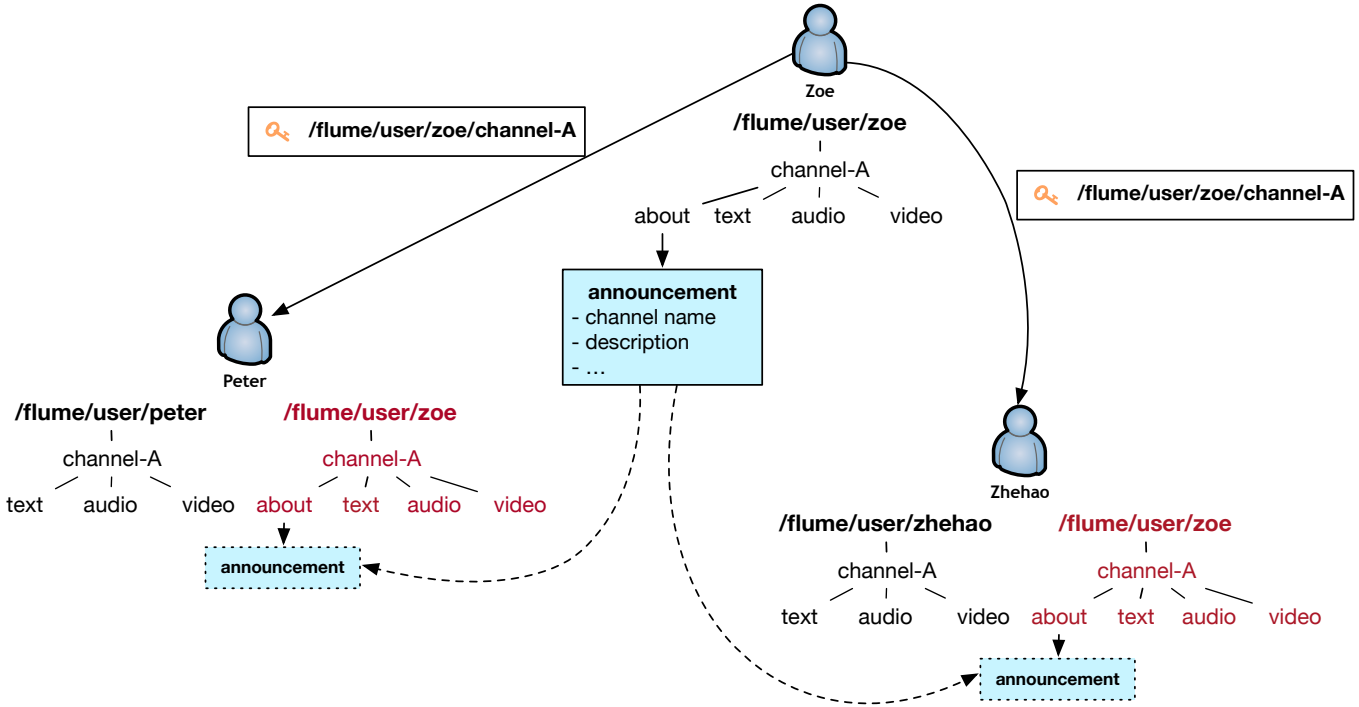
Storage. Serving data



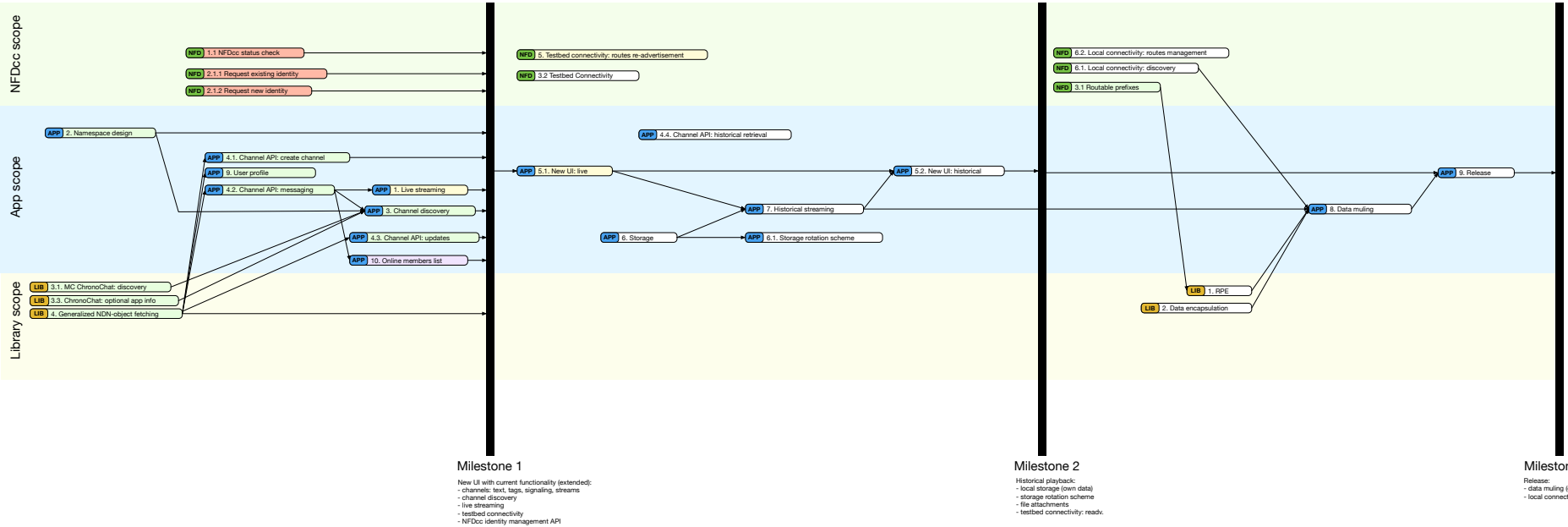
Storage. Requesting data



Data mules


Zoe
/flume/user/zoe/channel-A
/flume/user/zoe
channel-A
about text audio video
/flume/user/zoe/channel-A
announcement
- channel name
- description
- ...
Peter
/flume/user/peter
channel-A
text audio video
/flume/user/zoe
channel-A
about text audio video
announcement
Zehao
/flume/user/zhehao
channel-A
text audio video
/flume/user/zoe
channel-A
about text audio video
announcement

Roadmap


NFDcc scope
App scope
Library scope
NFD 1.1 NFDcc status check
NFD 2.1.1 Request existing identity
NFD 2.1.2 Request new identity
APP 2. Namespace design
APP 4.1. Channel API: create channel
APP 9. User profile
APP 4.2. Channel API: messaging
APP 1. Live streaming
APP 3. Channel discovery
APP 4.3. Channel API: updates
APP 10. Online members list
LIB 3.1. MC ChronoChat: discovery
LIB 3.3. ChronoChat: optional app info
LIB 4. Generalized NDN-object fetching
NFD 5. Testbed connectivity: routes re-advertisement
NFD 3.2 Testbed Connectivity
APP 4.4. Channel API: historical retrieval
APP 5.1. New UI: live
APP 5.2. New UI: historical
APP 7. Historical streaming
APP 6. Storage
APP 6.1. Storage rotation scheme
NFD 6.2. Local connectivity: routes management
NFD 6.1. Local connectivity: discovery
NFD 3.1 Routable prefixes
APP 9. Release
APP 8. Data muling
LIB 1. RPE
LIB 2. Data encapsulation
Milestone 1
New UI with current functionality (extended):
- channels: text, tags, signaling, streams
- channel discovery
- live streaming
- testbed connectivity
- NFDcc identity management API
Milestone 2
Historical playback:
- local storage (own data)
- storage rotation scheme
- file attachments
- testbed connectivity: readv.
Milestone 3
Release:
- data muling (other peer's data)
- local connectivity

TBD

- **NFD scope**
  - Identity management API − ***To Be Implemented***
  - Testbed connectivity status − ***To Be Implemented***
  - Certificate bundle/NFD RIB direct cert fetch − ***To Be Implemented***
  - Routes re-advertisement − ***Feedback (#3818)***
  - Local connectivity − ***To Be Implemented***
- **Library scope**
  - Routable prefix exchange − ***To Be Implemented***
  - Persistent storage − ***To Be Implemented***
- **App scope (*To Be Implemented*)**
  - Storage
  - Channel API: historical playback
  - Data mules (multi-producer)
  - UI

THANK YOU