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

Welcome to Flume!
Your Flume identity needs to be set up.

Please, choose signing identity from the list of available identities below. Chosen identity will be used to uniquely identify you among Flume users.

/ndn/edu/ucla/remap/peter

Use identity
Incorporating schematized trust

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

Diagram:
- NFD Control Center
  - App1
    - Instance Key
    - App1 Signing Key
    - App1 Key
  - App2
    - Instance Key
    - App2 Signing Key
  - App3
    - Instance Key
    - App3 Signing Key
  - App1 Key
- NDN
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

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

o.setPayload("{ username: Peter; bio: 'UCLA REMAP'; }" until namespace is local
profile.sync(o,
   [](NamespaceObject& o){
       // published "/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
Roadmap

Milestone 1
- New UI with current functionality (plaintext)
  - 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
- Historical playback
  - local storage (other peer's data)
  - local connectivity

NearDoc scope
- NFDcc status checks
- NFDcc stack debug
- NFDcc routing

App scope
- Data encapsulation
- Data muling
- Live streaming
- Storage
- New UI: live
- New UI: historical
- Testbed connectivity: routes re-advertisement

Library scope
- MC ChronoChat: discovery
- MC ChronoChat: optional app info
- Storage rotation scheme
- File attachments
- Testbed connectivity: readv.
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