CATCHING UP WITH NDN-RTC

Peter, Jeff
UCLA REMAP
7th NDN retreat @ CSU
Multi-user tests in Spring ‘16

- NDN seminar WebEx bridging
- Focusing on QoE metrics:
  - interruptions
  - latency
- 4-5 clients connected to different hubs
  - UCLA, REMAP, CAIDA, WU, CSU
Issues (#3485)

• Choppy audio:
  – consumers affect each other?
    • additional tests didn’t reproduce the problem (#3526) – requires further testing

• Performance (CPU usage):
  – chat discovery library bug (#3508) – fixed
  – interests forwarded to all producers connected to the same access hub
    • problem: ineffective AccessStrategy measurements (#3219 – in progress)
    • can be avoided with auto prefix propagation
      – required changes in NDN-RTC namespace and additional refactoring – completed

• Weird traffic patterns:
  – BestRouteStrategy: initial retx suppression is too low (#3230) – re-open?
    • waiting on NDNLPv2 link reliability improvements (#3230#note-8)
    • need to reproduce the problem (#3551)
Refactoring and Schematized Trust

• Namespace changes

• Key management
Supporting schematized trust

• Hierarchical verification
  – [testbed root]--signs→ [testbed node]--signs→ [user]--signs→ [app]--signs→ [app instance]--signs→ [data]

• NDN app “eco-system”:
  – who’s responsible for serving app certificate?

• Key rollover for delay-sensitive apps:
  – pre-fetch new certificate before data arrives
  – explicit vs. implicit rollover mechanism
Namespace refactoring

```
/root/ndnrtc/user/<producer-id>/
    |       |
    v       v
  streams  session info
    |
    v
cam
    |
    v
screen
     |
     v
pmcu low mid hi
     |
     v
frames
      |
      v
delta
      |
      v
key
     |
     v
0 1 ...
N

/data
    |
    v
data_type
    |
    v
segment
    |
    v
data
    |
    v
parity
      |
      v
00 01 ...

<stream name>

<sequence no>

Encoding Metadata

<segment #>

Frame Payload

<segment #>

FEC Payload

<segment #>

Manifest Payload

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>

<segment #>

<segment #>

<segment #>

<sequence no>
Mundane resilience
Making the ‘out-of-the-box’ NDN platform live up to the hype

A key advantage of NDN apps should be the resilient communication that is supposed to be inherited from the stack and work with or without global connectivity.

• If a face goes down, automatically use another face (#3521) - **in progress**
• Automatic use of multiple interfaces (#3566) – **completed** *(how do we use?)*
• Prefix propagation for non-home hubs (#3521) - **in progress**
• Autoconfig-related features (#3465, 2719, 3261, 3800)
• Remote prefix registration: fetch certificates from requester without any other connectivity requirements (#2237) – **open for 2 years**
• Traceroute and other testing tools, where are they? *(nfd-dev Sept 9)*
• **How can we shift the design philosophy to emphasize this type of resilience from the start of building new features, rather than make them fixes?**
Fetching latest data - Data Stream

- Sample period – $T$
- Sample sequence counter – $S$
- Freshness period – $F = \alpha T$
Establishing Low Latency

- Ask for Rightmost child, MustBeFresh in **Delta** namespace
- Upon receipt, initialize $\lambda$, $S$ and $Q$ (query counter)
- Fill the pipeline with outstanding interests
- $\lambda$ may not be precise, but the value is within these boundaries:
  - $[\text{DRD}/T; \text{DRD}/T+\alpha]$
ndncon, new incarnation

Demo
Flume Namespace (draft v1)
Use Cases: Local connectivity

1. NDN hackathon, WiFi AP or adhoc;
2. 5 users join Flume channel and able to exchange text, files, streams (screenshare, video/audio) b/w each other

How to automatically bootstrap local communication between NFDs connected to the same WiFi AP/ad hoc?
Use Cases: Local connectivity (cont.)

- Similar to using Ethernet multicast face
- But usually WiFi APs (commercial) configured to disable multicast
- **Workaround #1**: use allowed multicast protocols for discovery, e.g. Bonjour (#3465 – open for 9 months)
- **Workaround #2**: use other mediums for discovery – Bluetooth
  - announce peer IP address and its’ connected peers via Bluetooth
- *Any other ways?*
Use Cases: Local connectivity (cont. 2)

• After peer is discovered, establish 2-way routes. How?
• Workaround #1:
  – leverage info from Bluetooth announcements to establish routes (over UDP)
• Workaround #2:
  – peers issue command Interests for creating backroutes
• *Any other ways?*
Use Cases: Local+Global connectivity

1. NDN meeting, WiFi AP with Internet connection;
2. 5 local and 3 remote users join Flume channel for text/file/stream exchange
3. Remote user(s) publish live stream
Use Cases: Local+Global connectivity (cont.)

- Local peers can have different home hubs
- When they fetch remote stream from remote peer, interests go towards these different hubs
- This results in duplicated traffic
- *Can just one of the peers fetch remote stream and other local peers from him instead?*
Use Cases: Peer as a Data Mule

1. Peter, Zhehao and JeffT join Flume channel “REMAP” and exchange text, file and JeffT shares screen for few minutes
2. Peter goes to the IRL meeting, there is WiFi, but no Internet connection
3. JeffB arrives to the IRL meeting, connects to WiFi and joins Flume channel “REMAP”
4. JeffB is able to fetch new data that was generated by Zhehao, Peter and JeffT while he was travelling: text messages, a file and a screen share
Use Cases: Peer as a Data Mule (cont.)

• Peer carries data known to her
  – it’s not only data generated by peer:
    • data, generated by other users in the channel
    • other users’ data, that transited through peers’ NFD
  – data is a subject to peer’s retention policy

• **What does this storage/repo need to be?**
  – able to watch (any) prefixes?
  – able to receive passing Interests and answer with data if available?

• Such a storage can be generalized:
  – services that provide users with persistent storage
  – “catches” data while user was offline
Handling Identities

• Establish root application namespace:
  – /com/flume
    • /com/flume/user
    • /com/flume/multicast
• User identity: /com/flume/user/peter
• With Internet connection – how does peer know which hub to connect to?
  – testbed identities unambiguously define a “home hub”
• Peer mobility: need to announce/propagate peer’s prefix to other hub. How/When?
Recap

• Local connectivity
  – bootstrapping
  – prefix registration
• Local+Global connectivity
  – avoiding duplicate traffic
• Peer as a data mule
  – what storage do we need
• Handling identities
  – hub selection
  – peer mobility and prefix announcement/propagation
so... when do we ship?