Network Measurement Framework and Path Tracing Utility for NDN

{ Lotfi.Benmohamed, Davide.Pesavento }@nist.gov
Network Measurement

• Network measurement is fundamental to network operation, application performance, and policy (not just research)

• Stakeholder communities involved:
  – Network user community
    • What the network is doing to my packets
    • Measure QoS/QoE/SLA
    • Academic community: networking research
  – Network operators
    • Anomaly detection/capacity planning
    • Monitor network health/network forensics
  – Network policy makers
    • Define baseline for policies/pricing
    • Measure adoption of technologies
Goal

• Empower NDN with a built-in measurement framework that
  – can support multiple use cases,
  – can be used by different applications that need to produce and/or consume network measurements

• As new applications are developed, they
  – can be instrumented with measurement probes, and
  – make use of the framework to meet their measurement needs
**Design**

- **Client-Agent interaction:**
  - 4-way handshake
  - Similar to RICE approach
    - Measurement request
    - Reply containing token
    - Interest with token
    - Data with results
New Effort

• DNMP: Distributed Network Measurement Protocol
  – Pollere’s project funded under a NIST SBIR Phase1, started August 1.

• Uses a brokerless publish-subscribe communications model for both the measurement requests and the resulting measurement data
  – with goal to automate process of adding measurement probes using trust schema templates

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Duration</th>
<th>Funding level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase I Feasibility</td>
<td>6 months</td>
<td>Up to $100K</td>
</tr>
<tr>
<td>Phase II R&amp;D with commercialization plan</td>
<td>2 years</td>
<td>Up to $300K</td>
</tr>
</tbody>
</table>
Path Tracing

• Think of path tracing as just one of the probes within the NDN Measurement Framework
• Challenges for tracing in ICN vs. in IP
  – In IP: one next-hop, one path
  – In ICN, due to Caching/Multiple paths/Forwarding strategy: path(s) followed by one Interest can change over time even if routing (i.e. FIB entry) doesn’t change
• Send a “trace” Interest to discover how a “regular” Interest would be routed
  – Returns list of traversed nodes
  – Associated RTTs
  – Potentially other information as well
• Parameters in trace request include:
  – Exploring multiple paths
  – Discover all available paths while bypassing strategy
  – Ignore cached content to discover producers
NDN-Trace Design

- $t_1$: time Interest arrives at trace daemon from NFD
- $t_2$: time Interest (for upstream) is sent to NFD
- $d_i^U = t_2 - t_1$
- $t_3$: time Data arrives at trace daemon from NFD
- $t_4$: time Data (for downstream) is sent to NFD
- $d_i^D = t_4 - t_3$

Diagram:

- Node 0 to Node 1
- Node i to Node j1
- Node j1 to Node j2
- Node j2 to Node k1

Time intervals:
- $T_{0,1}$
- $T_{i,j1}$
- $T_{i,j2}$
- $d_{i,j2}$

Connections:
- 0 → 1
- i → j1
- j1 → j2
- j2 → k1
- k2 → i

Time points:
- $t_1$
- $t_2$
- $t_3$
- $t_4$
Other Traceroute Proposals

  - For tracing multiple paths:
    - requires PIT entries (created by trace requests) to be kept alive for some timeout period
    - represents a significant departure from the “flow balance” principle
  - Does not provide RTT measurements between the client node and intermediate nodes on the path
    - only provides RTT between client and the node where the content is found (same as ping)
    - also provides one-way delay, but requires synchronized clocks
  - Trace built in the request packet as it travels upstream

  - Resembles the mechanism used by IP traceroute tool
  - Iteratively discovers the path by issuing Interests with a progressively increasing HopLimit field
  - Uses new PathSteering header:
    - constructed hop-by-hop while the reply travels back to the client
    - included in subsequent trace requests that must be forwarded along the same path
NMF: NDN Network Measurement Framework

NDN-Trace is just one of the many “probes” of the **NDN Measurement Framework**.

Poster at ACM ICN’17

NMF: Status

Completed implementation of first version, including:
- Node agent with a few measurement probes
  - https://github.com/usnistgov/nmf-agent
- C++ client library and interactive command-line tool
  - https://github.com/usnistgov/nmf-client

Design of new features ongoing
- “Long-term” storage of measurements data (repo?)
- Access control (NAC?)
- Capability discovery/negotiation (pub-sub?)

Two main use cases
- Monitoring of operator networks, POV of the operator
- Measure/probe the network from the edge (RIPE Atlas, CAIDA Ark)
NMF: Basic Interaction

Client

Send measurement request with name:
\(<\text{domain}\><\text{node}\>/\text{NMF}/\text{measure}\/<\text{metric}\>/\text{MParams}\>/\text{CommandInterestComponents}\>

Agent

- perform measurement
- construct Data packet

Probe

alt

[instant measurement]

Reply with Data packet containing status and results

[long-term measurement]

Reply with Data packet containing status and token

Send Interest with token

Reply with Data packet containing status and results

Client

Agent

Probe
NMF: Eventual Goals

Extensibility
- Anyone can add a custom probe
- We will define an internal API for probes and provide a small library to help writing new probes
- App developers can integrate the framework in their app

Integration with access control protocols
- Measurement data can be sensitive
- Different clients have different access privileges

Long-term storage and querying of historical data
NMF: Protocol Details

Verbs
- measure
- query
- retrieve
- interrupt

We have defined:
- Packet format, including TLV encoding
- Client <-> agent protocol
NMF: Protocol Details

/\<domain\>/\<node\>/NMF/\<verb\>/\<MParams\>/\<CIComps\>
(still based on v0.2 packet format)

\[
\begin{align*}
\text{MParams} & :\ = \text{MPARAMS-TYPE TLV-LENGTH} \\
\text{Token} & :\ = \text{TOKEN-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string} \\
\text{Label} & :\ = \text{LABEL-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string} \\
\text{When} & :\ = \text{WHEN-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string} \\
\text{(repeatable) Parameter} & :\ = \text{PARAMETER-TYPE TLV-LENGTH} \\
\text{Name} & :\ = \text{NAME-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string ("face.id" => \texttt{\'f\', \texttt{\'a\', \texttt{\'c\', \texttt{\'e\', \texttt{\'.\', \texttt{\'i\', \texttt{\'d\'))}}}} \\
\text{Value} & :\ = \text{VALUE-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string ("257" => \texttt{\'2\', \texttt{\'5\', \texttt{\'7\'))}} \\
\text{(repeatable) Result} & :\ = \text{RESULT-VALUE-TYPE TLV-LENGTH} \\
\text{Name} & :\ = \text{NAME-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string} \\
\text{Value} & :\ = \text{VALUE-TYPE TLV-LENGTH} \\
& \quad \text{non-terminated unsigned char string}
\end{align*}
\]
Old NDN-Trace Design
Old NDN-Trace Design

Current v1 implementation is considered “done”
- Paper at ACM ICN’17
  - [https://conferences.sigcomm.org/acm-icn/2017/proceedings/icn17-86.pdf](https://conferences.sigcomm.org/acm-icn/2017/proceedings/icn17-86.pdf)
- Tested by John DeHart on ONL

However:
- It requires non-trivial changes to every strategy
- Code is hard to maintain
- We’d like to have more power over the tracing process (e.g., trace only a subset of next hops)
NDN-Trace: New “In-Forwarder” Design

Forwarding plane dispatches trace requests and responses to a traceroute module.

Strategy’s Interest forwarding API adds a dry-run mode.

- In dry-run mode, the chosen strategy offers its forwarding decisions, but does not transmit Interests or alter any state or data structures.

The traceroute module invokes strategy dry-run mode, transmits trace requests to next hops as needed, and collects and returns responses.
NDN-Trace: Router Behavior

1. Let the strategy decide which next hops it would use for an Interest, assuming the Interest was retransmitted $T$ times ($T=0$ for new Interests).

2. Send trace requests to the first $K$ next hops chosen by the strategy.

3. Collect the first $R$ trace responses that come back before a timeout $TO$ expires, discard the rest.

4. Combine the responses into one (how?) and send it back.

![Diagram showing the trace module, forwarding strategy, and FIB with downstream and upstream arrows.]
NDN-Trace: Request
NDN-Trace: Reply
NDN-Trace: Router Behavior

Reply forwarding/aggregation: two approaches
- Hop-by-hop wrapping and signing, maintains existing signatures
- Discard signature, re-sign, send downstream

Trace process termination
- Producer
- Cache
- Either producer or cache, whichever is found first
- Abnormal termination
  - No route
  - Administratively prohibited
  - ...
Bypassing the Forwarding Strategy

It can be argued that a path (next hop) that appears in FIB but would never be considered by a strategy is not a meaningful path.

- Some strategies do not use the FIB, e.g.:
  - Flooding strategy always floods.
  - Self-learning strategy may flood. FIB may be empty.
  - Hyperbolic FIB (in eventual implementation) maps prefixes to coordinates instead of next hops. FIB does not contain next hops.

Therefore the traceroute module should never look at the raw FIB entries without going through a strategy.

But what if we need to explore alternative paths that the current strategy would not consider?

- Do we need this functionality, e.g. for strategy debugging?
NDN-Trace: Future Work

Currently exploring how to adapt NDN-Trace to take advantage of new features in NDN Packet Format v0.3

- Use **typed name components** to recognize trace Interests?
- Carry parameters in Interest **Parameters** TLV

Explore deployment/testing on the NDN Testbed

Lots more...