NDN Architectural Development and Routing Design

Lan Wang
University of Memphis
Nov. 21, 2014
www.named-data.net
Outline

• Introduce basic concepts of Named Data Networking (NDN)

• Present an overview on NDN development

• Show an example of name-based routing

• Summarize potential solutions in scaling NDN routing
NDN Project

NDN: Named Data Networking

- NDN project started on 9/1/2010
- Part of NSF Future Internet Architecture Program

NDN-NP (NDN-Next Phase)

- Started in May 2014
- 8 collaborating institutions

http://www.named-data.net/

<table>
<thead>
<tr>
<th>NDN-NP PROJECT PIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCLA: Van Jacobson, Jeff Burke, Lixia Zhang</td>
</tr>
<tr>
<td>University of Arizona: Beichuan Zhang</td>
</tr>
<tr>
<td>UCSD: Kim Claffy</td>
</tr>
<tr>
<td>Colorado State University: Christos Papadopoulos</td>
</tr>
<tr>
<td>UIUC: Tarek Abdelzaher</td>
</tr>
<tr>
<td>University of Memphis: Lan Wang</td>
</tr>
<tr>
<td>University of Michigan: J. Alex Halderman</td>
</tr>
<tr>
<td>Washington University: Patrick Crowley</td>
</tr>
</tbody>
</table>
Today’s Internet Traffic

- Communication is “Content” driven.
Internet Protocol (IP)

- Underlying communication is “destination” driven.
Data distribution via TCP pipes
Data distribution via TCP pipes
Data distribution via TCP pipes
Data distribution via TCP pipes
Data distribution via TCP pipes
Challenges Caused By Point-to-Point Communication

**Difficult to**

- *disseminate* data to a large group of users.
- *handle* mobile users whose addresses change.
- *secure* data as it moves from device to device.
A Simpler Way

Suppose you could ask for what you wanted?

/memphis.edu/lanwang/talks/UALR14.pdf

/named-data.net/video/van-ccnx

/room/thermostat/1/status
NDN: Retrieving Named Data

**Interest packet**
- Content Name
- Selector
  - (order preference, publisher filter, scope, ...)
- Nonce

**Data packet**
- Content Name
- Signature
  - (digest algorithm, witness, ...)
- Signed Info
  - (publisher ID, key locator, stale time, ...)
- Data

User/App supplied name:
- /parc.com/videos/WidgetA.mpg/
- _v <timestamp>_/s3

Globally-routable name
Organizational name
Conventional/automatic

Versioning & Segmentation
Named Data Networking (NDN)
Named Data Networking (NDN)

www.youtube.com/slamdunk

Internet

www.youtube.com/slamdunk

www.youtube.com/slamdunk
Named Data Networking (NDN)

www.youtube.com/slamdunk

www.youtube.com/slamdunk

www.youtube.com/slamdunk

www.youtube.com/slamdunk/s1

Internet
Named Data Networking (NDN)
Named Data Networking (NDN)

www.youtube.com/slamdunk

www.youtube.com/slamdunk/s1

www.youtube.com/slamdunk

www.youtube.com/slamdunk/s1

www.youtube.com/slamdunk/s1

www.youtube.com/slamdunk
Named Data Networking (NDN)
Named Data Networking (NDN)
Named Data Networking (NDN)
Named Data Networking (NDN)
Named Data Networking (NDN)
Named Data Networking (NDN)
Named Data Networking (NDN)

NDN uses "Interest" and "Data" packets to get "content"
Data distribution via NDN
Packet forwarding in NDN

Downstream
- Interest → Content Store
- Content Store → Pending Interest Table (PIT)
- PIT → FIB
- FIB → forward
- Data
- cache
- Content Store
- PIT
- discard Data
- lookup hit
- lookup miss

Upstream
- forward
- Data
- Pending Interest Table (PIT)
- PIT
- Content Store
- discard Data
- lookup hit
- lookup miss
Why network security is difficult today?
Why network security is difficult today?

“the ‘trust’ the user gets in the content they depend upon is tied inextricably to the connection over which it was retrieved, that trust is *transient* – leaving no reliable traces on the content after the connection ends”
Why network security is difficult today?

“the ‘trust’ the user gets in the content they depend upon is tied inextricably to the connection over which it was retrieved, that trust is *transient* – leaving no reliable traces on the content after the connection ends”

Old way of thinking: add boundary for protection

- Securing my IP network $\rightarrow$ securing its perimeter
- But people punch holes on the barrier to get work done
- Strong security $\rightarrow$ burden/barrier to communication
Think differently

• The ultimate question from applications: did I get the right data I need?
  - and did others peek into my data (if secrecy required)?

• NDN secures data directly: every Data packet is signed by the producer
  $\rightarrow \textit{reduces}$ the reliance on trusting network intermediaries

$\rightarrow$ This leaves one problem to solve: verify that the data is indeed signed by the producer’s key

$\rightarrow$ Each NDN application has its trust model to verify producer’s key.
NDN ARCHITECTURAL DEVELOPMENT
Preserving the Hourglass Shape

- The only layer that requires universal agreement is layer 3, the network layer.
- Much of IP’s success is due to the simplicity of its network layer and the weak demands it makes on layer 2.
NDN Architecture Development [1]

- application-driven
- test and deploy on operational testbed
- conduct real-world demos

Progress

All code is open source at https://github.com/named-data/.

- **Multimedia applications**
  - NDNVideo [2]
  - ChronoChat [3]
  - NDN-RTC
  - ChronoShare [4]
- **IoT applications**
  - Building automation and management [5, 6]
  - Vehicular net [7]
- **Libraries**: NDN-CCL, NDN-CXX, pyNDN2, ndn-js, ChronoSync [9], NDN repo
- **NDN Forwarding Daemon (NFD)**
- **Routing protocol** [8]
ChronoChat \[^3\]\n
- **server-less** chat application based on ChronoSync
- chat messages are synchronized among participants
- leverage multicast nature of NDN

---

NDNVideo [2]

- Live and pre-recorded streaming to multiple consumers.
- No session semantics => scalability. Tested for ~1000 clients from 1 src
- First Interest sent can randomly access a keyframe at any timecode
- Leverages caching.

**ndnrtc**

- Real-time audio/video/text chat application enabling multi-peer conferencing over NDN.
- Version 0.1.0: Oct 2014.

- Based on WebRTC codebase, using ChronoSync for conference discoveries.

- Explore how to handle packet losses and delays while maintaining a session-less approach.
ChronoShare: Distributed File Sharing and Editing [4]

- Think Google Drive, but no centralized server
  - Different users can share folders.
  - Each user can sync folders on different devices.
- How?
  - Each user’s actions (on file) form a stream of data.
  - Use ChronoSync to distribute knowledge of user actions

Building Automation and Management \[5,6\]

- Improve application development process, management, interoperability and security.
- Practical work so far: NDN interfaces to BacNET and Modbus sensing, authenticated lighting control.
- Partner: UCLA Facilities Management.

NDN Platform Release

- **http://named-data.net/codebase/platform/**
  - a new release every few months

  - The NDN Forwarding Daemon (NFD), version 0.2.0
  - The ndn-cxx library, version 0.2.0
  - The NDN Common Client libraries suite (NDN-CCL), version 0.3
    - C++ - NDN-CPP, Python - PyNDN2, JavaScript - NDN-JS, Java - jNDN
  - The Named Data Link State Routing Protocol (NLSR), version 0.1.0
  - The next generation of NDN repository (repo-ng), version 0.1.0
  - A ping application For NDN (ndn-tlv-ping), version 0.2.0
  - A traffic generator For NDN (ndn-traffic-generator), version 0.2.0
  - A packet capture and analysis tool for NDN (ndndump), version 0.5
Scalable Forwarding Engine Design

• Requirements
  - data structures to store millions to billions of names
  - fast table lookup of variable-length names
  - fast packet processing

• NDN project’s progress
  - multi-million entry FIBs stored in less than 10MB [10]
  - FIB lookup speeds on the order of microseconds [10]
  - PIT: 37 to 245 MiB memory for 100 Gbps throughput (small enough to fit in fast memory chips) [11]

NDN Testbed

More info at [http://named-data.net/ndn-testbed/](http://named-data.net/ndn-testbed/)
Contact us if interested in joining the testbed.
## Annual Demonstrations

<table>
<thead>
<tr>
<th>Demo Feature</th>
<th>2012 Demo</th>
<th>2013 Demo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large-scale, <strong>wide-area operation</strong></td>
<td>All 4 US time zones, ~300 machines</td>
<td>5 continents, ~1000 machines</td>
</tr>
<tr>
<td>Mix of content distribution and <strong>interactive apps</strong></td>
<td>4 distinct services</td>
<td>Multiple services</td>
</tr>
<tr>
<td><strong>Visualization</strong> of both app-level and net-level activity</td>
<td>NDN map</td>
<td>NDN map</td>
</tr>
<tr>
<td>Demonstrate both steady-state and <strong>react-to-change</strong> modes</td>
<td>Drop links during app sessions</td>
<td>Forwarding strategy</td>
</tr>
<tr>
<td><strong>Something IP+HTTP cannot do</strong></td>
<td>Scalable video streaming*, multi-path routing</td>
<td>Scalable video streaming*, multi-path routing</td>
</tr>
<tr>
<td><strong>Integrated PKI, better security</strong></td>
<td></td>
<td>Show key auth</td>
</tr>
<tr>
<td><strong>NDN-based device monitoring</strong></td>
<td></td>
<td>Stage lighting ctrl</td>
</tr>
</tbody>
</table>
Live bluegrass band performance, NDN-based control of stage lights

- Delivery of live audio and video from performance studio at UCLA
  - Jeff Burke’s Center for Research in Engineering, Media and Performance (REMAP)
- Lighting control application is NDN-based
- Server at studio homed off REMAP gateway
- Laptop on-site homed off Tokyo gateway
NDN Next Phase

• NSF FIA-NP (2014-2016)

• Applications
  - Open M-Health
  - E-BAMS
  - Mobile Media Application Cluster

• Forwarding and routing: Interdomain routing, forwarding strategy

• Security: privacy, trust management

• NDN Consortium (Sept. 2014): industry and academic members
A NAME-BASED ROUTING PROTOCOL
Routing in NDN

- **Requirement: Routing based on “name”**
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- **Non-requirement: Fast routing convergence**
  - Stateful forwarding plane can adapt to changes

A only has some content under /nytimes/art

C wants to read /nytimes/art
Routing in NDN

- Requirement: Routing based on “name”
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- Non-requirement: Fast routing convergence
  - Stateful forwarding plane can adapt to changes
Routing in NDN

- **Requirement: Routing based on “name”**
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- **Non-requirement: Fast routing convergence**
  - Stateful forwarding plane can adapt to changes
Routing in NDN

- **Requirement**: Routing based on “name”
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- **Non-requirement**: Fast routing convergence
  - Stateful forwarding plane can adapt to changes

---

![Diagram](https://via.placeholder.com/150)

- C wants to read `/nytimes/art`
- A only has some content under `/nytimes/art`
- D’s FIB: `/nytimes/art` - A, B, E
Routing in NDN

- **Requirement: Routing based on “name”**
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- **Non-requirement: Fast routing convergence**
  - Stateful forwarding plane can adapt to changes
Routing in NDN

- Requirement: Routing based on “name”
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- Non-requirement: Fast routing convergence
  - Stateful forwarding plane can adapt to changes
Routing in NDN

- **Requirement: Routing based on “name”**
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- **Non-requirement: Fast routing convergence**
  - Stateful forwarding plane can adapt to changes

---

[Diagram showing routing in NDN with nodes A, B, C, D, E and paths for /nytimes/art, /nytimes/art1, /nytimes/art/img3.]

- Node A only has some content under /nytimes/art
- Node D's FIB: /nytimes/art [A, B, E]
Routing in NDN

• Requirement: Routing based on “name”
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

• Non-requirement: Fast routing convergence
  - Stateful forwarding plane can adapt to changes
Routing in NDN

- Requirement: Routing based on “name”
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- Non-requirement: Fast routing convergence
  - Stateful forwarding plane can adapt to changes
Routing in NDN

- **Requirement: Routing based on “name”**
  - Guides each “interest” packet to all potential providers (all paths)
  - Some providers may not have all content in a “name”

- **Non-requirement: Fast routing convergence**
  - Stateful forwarding plane can adapt to changes

D’s FIB:

- /nytimes/art
- A, B, E

A only has some content under /nytimes/art

C wants to read /nytimes/art

Interest for /nytimes/art/img3

DATA

Interest for /nytimes/art1
Routing Mechanism in NDN

Any routing algorithm that works for IP (e.g., link-state) can be used in NDN.

- NDN’s forwarding semantics is a superset of the IP model.

Differences:
- replace IP prefixes with name prefixes
- calculate a list of next-hops for each name prefix
- Propagate routing updates using Interest/Data packets
Routing Mechanism in NDN

Any routing algorithm that works for IP (e.g., link-state) can be used in NDN.

- NDN’s forwarding semantics is a superset of the IP model.

Differences:
- replace IP prefixes with name prefixes
- calculate a list of next-hops for each name prefix
- Propagate routing updates using Interest/Data packets

A can reach /cnn/video

A

/cnn/video

B

E

D

C

/cnn/video
Routing Mechanism in NDN

Any routing algorithm that works for IP (e.g., link-state) can be used in NDN.

- NDN’s forwarding semantics is a superset of the IP model.

Differences:
- replace IP prefixes with name prefixes
- calculate a list of next-hops for each name prefix
- Propagate routing updates using Interest/Data packets
Routing Mechanism in NDN

Any routing algorithm that works for IP (e.g., link-state) can be used in NDN.

- NDN’s forwarding semantics is a superset of the IP model.

Differences:
- replace IP prefixes with name prefixes
- calculate a list of next-hops for each name prefix
- Propagate routing updates using Interest/Data packets
Routing Mechanism in NDN

Any routing algorithm that works for IP (e.g., link-state) can be used in NDN.

- NDN’s forwarding semantics is a superset of the IP model.

Differences:
- replace IP prefixes with name prefixes
- calculate a list of next-hops for each name prefix
- Propagate routing updates using Interest/Data packets

A can reach /cnn/video
B can reach /cnn/video
D’s FIB: /cnn/video A, B, E
Motivation for a Native Name-based Routing Protocol

- Need a routing protocol for NDN networks
  - Populate RIB/FIB so routers can forward interests
  - Not necessarily point to the nearest cache.

- Previous work: OSPFN - simple extension of IP OSPF
  - Advertises “name” by OSPF Opaque LSA
  - Difficult to manage IP addresses and GRE tunnels
  - Only single path in most cases
  - Security relies on password or secret shared among all routers on a subnet.

- Build a routing protocol purely on top of NDN
Named-data Link State Routing (NLSR)[1]

- Use a mature routing algorithm: link state
- NDN native
  - Names, not addresses (networks, routers, processes, data, keys)
  - Interest/Data are used to distribute routing info.
- Multipath support: modified Dijkstra’s algorithm to produce a ranked list of next-hops
- Security
  - Routing data is signed by originating router and verified by receivers based on a trust model.
  - a trust model for intra-domain routing

Naming in NLSR

- Follow the hierarchy within a network
  - Easy to identify the relationship among entities
  - Easy to associate keys with key owners

- Router
  - /<network>/<site>/<router>: e.g., /ndn/memphis/rtr1

- Updates
  - /<network>/NLSR/LSA/<site>/<router>/<process>/<type>/<version>

- Keys
  - NLSR key: /<network>/<site>/<router>/<process>/key
  - Router key, operator key, ...
Link State Advertisement (LSA)
Link State Advertisement (LSA)

Prefix LSA

\[ /\text{<network>}/\text{NLSR}/\text{LSA}/\text{<site>} /\text{<router>}/\text{prefix}/\text{<version>} \]

- number of prefixes
- name prefix 1
  
  
  
- name prefix n

signature
Link State Advertisement (LSA)

**Prefix LSA**

```plaintext
/<network>/NLSR/LSA/<site>
/<router>/prefix/<version>
```

<table>
<thead>
<tr>
<th>number of prefixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name prefix 1</td>
</tr>
<tr>
<td>....................</td>
</tr>
<tr>
<td>name prefix n</td>
</tr>
</tbody>
</table>

**signature**

Advertises reachable name prefixes by router
Link State Advertisement (LSA)

**Prefix LSA**

- `/<network>/NLSR/LSA/<site>/`<br>`/<router>/prefix/<version>`
- number of prefixes
- name prefix 1
- .....................
- name prefix n
- signature

**Adjacency LSA**

- `/<network>/NLSR/LSA/<site>/`<br>`/<router>/adj/<version>`
- number of adjacencies
- neighbor 1, link cost 1
- .....................
- neighbor n, link cost n
- signature

Advertises reachable name prefixes by router
## Link State Advertisement (LSA)

### Prefix LSA

- **Number of prefixes**
- **Name prefix 1**
- **Name prefix n**

**Signature**

Advertises reachable name prefixes by router

### Adjacency LSA

- **Number of adjacencies**
- **Neighbor 1, link cost 1**
- **Neighbor n, link cost n**

**Signature**

Advertises adjacency state:
- Topology information
Routing Security and Trust Model

- Every NLSR Data packet is signed by the NLSR process key.
- “Key locator” includes information about the key.
- Receiver retrieves the NLSR key and verifies the Data signature.
- Then receiver verifies the NLSR key, router key, operator key, and site key (the root key is preconfigured).

<table>
<thead>
<tr>
<th>Entity</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root key</td>
<td>/&lt;network&gt;/key</td>
</tr>
<tr>
<td>Site key</td>
<td>/&lt;network&gt;/&lt;site&gt;/key</td>
</tr>
<tr>
<td>Operator key</td>
<td>/&lt;network&gt;/&lt;site&gt;/&lt;operator&gt;/key</td>
</tr>
<tr>
<td>Router key</td>
<td>/&lt;network&gt;/&lt;site&gt;/&lt;router&gt;/key</td>
</tr>
<tr>
<td>NLSR key</td>
<td>/&lt;network&gt;/&lt;site&gt;/&lt;router&gt;/NLSR/key</td>
</tr>
<tr>
<td>Data</td>
<td>/&lt;network&gt;/NLSR/LSA/&lt;site&gt;/&lt;router&gt;/&lt;type&gt;/&lt;ver&gt;</td>
</tr>
</tbody>
</table>
NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

From Flooding to Synchronization

NLSR uses ChronoSync [2] to synchronize LSDB.
- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.


www.named-data.net
NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.
- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

NLSR uses ChronoSync [2] to synchronize LSDB.

- Every node periodically sends a digest of LSDB to others in Interest packets.
- When a node has a new LSA, its digest changes and it will reply to others’ Interests with name of new LSA.
- Other nodes fetch new LSA.

Development Status

• NLSR 0.1 was released on 8/25/14.
  - Supports both link state and hyperbolic routing
  - Uses ChronoSync to synchronize routing data
  - Uses a hierarchical trust model for routing within a single administrative domain (validation rules are configurable).

• Running on NDN testbed.

• Code and doc:
  - http://named-data.net/doc/NLSR/0.1.0/
Lessons from practice

- NLSR provides a real use case for several NDN features, and drives the development of these features.
  - Security and trust model in NDN-CXX library
  - RIB management and prefix management in NFD

- Using ChronoSync simplifies protocol design and implementation.
  - ChronoSync fixed a number of problems of previous sync/repo.

- Testbed deployment helped discover problems in NLSR, NFD, NDN-CXX.
  - E.g., clock out-of-sync, flaky UDP tunnels.
Scaling Solution 1: Map-and-Encap

- Number of application names is unbounded: over 200 million 2nd-level DNS names
  - Map edge network addresses to transit network addresses.
NDN can follow Map-n-Enap approach.

- Map application name prefixes to routable name prefixes, which usually belong to ISPs.
- Packets carry the routable name prefix (and application name).
- Routers store only routable (ISP) name prefixes.
APT-NDN

- Concrete realization of Map-and-Encap in NDN
- Status: implemented in ndnSIM.
Scaling Solution 2: Hyperbolic Routing

- There are many (emerging) ways to do routing, e.g., Small Worlds, Geographic Hyperbolic, Epidemic percolation.
- In general they’re easier to implement and work better for NDN than for IP:

- Hyperbolic routing
  - Each node and name prefix have a set of semi-static hyperbolic coordinates.
  - Calculate next-hops based on each neighbor’s distance to the name prefix
  - No need to distribute topology (links) and updates.
Greedy Forwarding in Hyperbolic Routing

- To forward packets:
  - Find neighbor who is closest to the destination
  - Forward the packet to the neighbor
Greedy Forwarding in Hyperbolic Routing

- To forward packets:
  - Find neighbor who is closest to the destination
  - Forward the packet to the neighbor
Current status

- Hyperbolic coordinates distributed in NLSR for comparison with link-state routing
- Preliminary emulation results at http://netwisdom.cs.memphis.edu/pythome.html
- Plan: run hyperbolic routing on NDN testbed next month
References


4. A. Afanasyev, Z. Zhu, L. Zhang, The story of ChronoShare, or how NDN brought distributed file sharing back, under review


References (cont’d)


