Performance Study of CCNx

Haowei Yuan

Networking Research Seminar
3/18/2013
My Topic for Today

• Industry participation in content centric networking
  – Emerging networks consortium

• Our performance study of CCNx
  – CCNx baseline benchmarking
  – Content distribution with NDN and HTTP
Emerging Networks Consortium

- An open industry consortium founded by PARC in 2012
  - Companies in different technology and communication sectors
  - ISPs, consumer electronic, content provider, OEMs, hardware manufacturers, and venture capitalists

- Founding members
  - Alcatel-Lucent, BT, France Telecom-Orange, HCL, Huawei, MACH, Panasonic, Samsung

(Logo is taken from www.parc.com)
Examples

• Alcatel-Lucent (Bell Lab)
  – Cache coordination
  – Caesar: a content router for high speed forwarding
  – Content delivery solutions

• France Telecom-Orange
  – Packet forwarding
  – Caching replacement policy

• Samsung
  – Mobile services and applications with CCN

• Huawei
  – Secure mobile virtual group service
Samsung

• Trends
  – Internet video traffic in 2014 is 5x larger than that in 2010*
  – Mobile video traffic in 2014 is 20x larger than that in 2010*

• Focuses on applying CCNx on mobile devices
  – CCNx released its Android version in 2010 by Parc

• Virtual private community (VPC)*
  – Content sharing in closed private community
  – Serverless
  – Requires scalable and secure infrastructure
  – Content access using VPC browser (CCN browser)

*Content is taken from Samsung presentations
Figure is taken from Samsung presentations
Samsung VPC Prototype Experiments

- 40 devices (Galaxy S2/Tab and Linux PC) connected over 4 VPC-2s
  - Easy enrollment & access control, content browsing, push, video streaming
  - Wi-Fi and 3G cellular
  - CCN cache vs. IP-like non-cache performance comparison

Slide is taken from Samsung presentations
Samsung’s Vision

- Runs as an overlay on top of TCP/IP
  - Fast local caches with better security, ease of configuration, mobility
- Starts as connected islands
  - Sparse at first but each new link adds value (incremental deployment)
- Moves to a dominant paradigm over time

This slide is taken from a Samsung presentation
CCNx Performance Study

- CCNx is a reference implementation developed by Parc
  - Runs as an IP overlay
  - Supports both TCP and UDP

- Content Store
  - "Cached Data"

- Pending Interest Table
  - "Remember who asked for what"

- Forwarding Information Base
  - "Where to go if I don't have the data"
User C wants /wustl/data1

Content Store

/wustl/data1  data...
/wustl/data2  data...
/parc/data1/seg1 data...
...

Pending Interest Table

/parc/data2  User A
...
...

Forwarding Information Base

/  Router A
/wustl  Router B
/wustl/data5/seg1  Router C
...
...
NDN Forwarding (Cache Miss)

User C wants /wustl/data3

Content Store

/wustl/data1   data...
/wustl/data2   data...
/parc/data1/seg1 data...
...

Pending Interest Table

/parc/data2   User A
...          ...

Forwarding Information Base

/                    Router A
/wustl                Router B
/wustl/data5/seg1   Router C
...                  ...

NDN Forwarding (Cache Miss)

Content Store

_user C wants /wustl/data3_

Pending Interest Table

| /parc/data2 | User A |
| /wustl/data3 | User C |
| ... | ... |

Forwarding Information Base

| / | Router A |
| /wustl | Router B |
| /wustl/data5/seg1 | Router C |
| ... | ... |
NDN Forwarding (Data Packets)

User C wants /wustl/data3

Content Store

<table>
<thead>
<tr>
<th>/wustl/data1</th>
<th>data...</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wustl/data2</td>
<td>data...</td>
</tr>
<tr>
<td>/parc/data1/seg1</td>
<td>data...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Pending Interest Table

<table>
<thead>
<tr>
<th>/parc/data2</th>
<th>User A</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wustl/data3</td>
<td>User C</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Forwarding Information Base

<table>
<thead>
<tr>
<th>/</th>
<th>Router A</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wustl</td>
<td>Router B</td>
</tr>
<tr>
<td>/wustl/data5/seg1</td>
<td>Router C</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
User C wants /wustl/data3

/wustl/data1
/wustl/data2
/parc/data1/seg1
/wustl/data3

/wustl/data3

/parc/data2
/wustl/data3
/wustl/data3

/ /Router A
/wustl/ /Router B
/wustl/data5/seg1/ /Router C
/.../...
/.../...
Packet Forwarding in HTTP and NDN

<table>
<thead>
<tr>
<th></th>
<th>HTTP</th>
<th>NDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding key</td>
<td>IP address</td>
<td>Name</td>
</tr>
<tr>
<td>Key length</td>
<td>32 bits</td>
<td>Variable</td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>Longest prefix Match</td>
<td>LPM</td>
</tr>
<tr>
<td>Per-packet READ</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Per-packet WRITE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cache lookup</td>
<td>Hash(O(1))</td>
<td>Skiplist (O(logn))</td>
</tr>
</tbody>
</table>

NDN forwarding requires lookups of variable-length names in large prefix tables that must support per-packet updates.
Exp: Measure CCNx Baseline Throughput

- Factor: Caching, data packet payload size
- Topology
Throughput (No Caching)

- Content store does not help
- Outgoing = 16 Interests + 16 Data
- Incoming = 16 Interests + 16 Data
Throughput (Caching)

- Content store does help
- Outgoing = 1 Interests + 16 Data
- Incoming = 16 Interests + 1 Data
CCNx Peak Throughput

- No caching: 150 Mbps
- Caching: 400 Mbps
- NDN peak throughput << 1 Gpbs
NDN Throughput w/ Longer Names

- More processing time for name lookup
  - Factor: number of components in names

![Chart showing throughput with varying number of components](chart.png)
Content Distribution with NDN and HTTP

- HTTP-based content distribution works but is *inefficient*
  - Youtube, ESPN and others distribute massive amount of content
  - Requires expensive globally distributed infrastructure, few can afford

- Named Data Networking (NDN) is an *efficient alternative* we are developing
  - A clean-slate architecture
  - Supports efficient and scalable content distribution

- How does the performance of NDN compare to HTTP
Two Name-Centric Approach Examples: HTTP and NDN

<table>
<thead>
<tr>
<th>HTTP and Web Caching</th>
<th>Named Data Networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Hack” the Internet</td>
<td>A clean slate design</td>
</tr>
<tr>
<td>Runs on top of IP</td>
<td>Replaces IP</td>
</tr>
</tbody>
</table>

- No experimental comparison between HTTP and NDN
Web Caching Improves HTTP

• Problem: popular content increases HTTP server load

• Web caching reduces server load

• Web caching’s problems
  – Needs manual configuration
  – Globally distributed cache network only feasible to large corporations
  – Cloud-based services
    • Lower costs for end-users but at great expense to the provider
NDN Supports Caching Intrinsically

- Cache on every NDN node
  - Servers, routers, laptops
  - Available to everyone

- Feature: Name packets rather than end hosts
  - Names are similar to URLs: ndn://ndn/content/name
  - Packets are forwarded based on names

- Two types of packets: Interest, Data
Experimental Evaluation

- **Goal:** Compare HTTP and NDN performance

- **Real applications on real hardware**
  - Single core machines with 1 Gbps physical links

- **Software**

<table>
<thead>
<tr>
<th>HTTP</th>
<th>NDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server: lighttpd-1.4.28(Youtube)</td>
<td>Prototype Version: 0.4.0</td>
</tr>
<tr>
<td>Cache: Squid-3.1.11 (Wikipedia)</td>
<td></td>
</tr>
</tbody>
</table>
Experiments

- Exp1: File distribution in wired net w/ HTTP & NDN
- Exp2: File distribution in wireless net w/ HTTP & NDN
Exp 1: File Distribution in Wired Net

- Distribute a **100** MB file to up to 40 end-hosts

- Metric: File download time

- Factors

<table>
<thead>
<tr>
<th>Factors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levels of Cache</td>
<td>1, 2</td>
</tr>
<tr>
<td>Number of Clients</td>
<td>5, 10, 15, 20, 25, 30, 35, 40</td>
</tr>
</tbody>
</table>
Wired Network Topology

- No packet drop/delay
Wired Network Results

- **HTTP vs. NDN**
  - 2-level caching improves both HTTP and NDN
  - HTTP is about 10X faster than NDN
Exp2: File Distribution in Wireless Net

- Distribute a 10 MB file to up to 40 end-hosts
- Metric: File download time
- Factors

<table>
<thead>
<tr>
<th>Factors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Loss Rate</td>
<td>5%, 10%</td>
</tr>
<tr>
<td>Link Delay (ms)</td>
<td>0, 50, 100, 200</td>
</tr>
</tbody>
</table>
Wireless Network Topology

1st level cache

2nd level cache (Access Points)

1st level cache

No 2nd level Cache (Access Points)

NDN

HTTP

Server

Clients

Link Delay

Lossy Link
Wireless Network Results (1 / 2)

- NDN performs comparably to HTTP with 5% loss rate and 50ms link delay
  - Link level retransmission improves performance
Wireless Network Results (2 / 2)

- NDN is about 4x faster than HTTP with 10% loss rate and 200ms link delay
Improve NDN Performance

Top 5 Time Consuming Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Percentage(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccn_skeleton_decode</td>
<td>35.46</td>
</tr>
<tr>
<td>ccn_compare_names</td>
<td>12.53</td>
</tr>
<tr>
<td>ccn_buf_match_dtag</td>
<td>8.95</td>
</tr>
<tr>
<td>ccn_buf_advance</td>
<td>6.64</td>
</tr>
</tbody>
</table>

- NDN prototype profiling using gprof
  - Name decoding takes > 60% of the time
  - Names are stored in encoded format
    - Multiple name decoding required for a lookup

- A simple modification: store decoded names directly
Improvement

- Store decoded names
  - File distribution peak throughput increased by 40%

![Graph showing throughput comparison between Original NDN and Modified NDN](image)
Summary

• CCNx baseline throughput measurement
  – 150 Mbps (cache miss)
  – 400 Mbps (cache hit)

• Compare content distribution with HTTP and NDN
  – HTTP is 10X faster than NDN in wired network
  – NDN is comparable with HTTP in wireless network

• NDN prototype improvement
Conclusion and Future Work

• CCNx baseline performance
  – Far from line rate (1 Gbps) requirement

• Was this a fair comparison? Not exactly
  – NDN prototype has not seen much optimization yet
  – NDN authenticates data sources

• A fully functional NDN traffic generator to explore richer distribution of content names and request patterns

• Scalable NDN forwarding plane design
  – Efficient algorithms: Bloom Filters
  – Advanced hardware: TCAM, FPGA
NDN Forwarding Plane Implementation

- Name Prefix Hash Table
  - Propagating Entries
  - Propagating Hash Table
- Forwarding Info Entries
- Pending Interest Table
- Content Hash Table
- Content Skip List
- Content Store
  - Content Array
  - Straggler Hash Table