Network Names in Content-Centric Networking

Cesar Ghali, Gene Tsudik and Christopher Wood
University of California, Irvine
{cghali, gene.tsudik, woodc1}@uci.edu

CCN Names

• Expressed as URIs
  – /a/b/foo

• Encoded as TLVs
Names in CCN Applications

Consumer

Request /a/b/foo

Router

Serve /a/b/foo or forward request

Network

Publish/Serve /a/b/foo

Producer
/a/b

Names in the Network

ROUTER

CS

PIT

FIB

(1) Exact name in CS?

(2) Exact name in PIT?

(3) Prefix in FIB?
Dual Roles for Names

- Applications use names to meaningfully express and identify content
  - Human readability is nice!
- Network entities (routers) use names as sequences of binary strings
  - A router doesn’t care about readability or arbitrarily long components

Q: Why is the same representation used at both layers & in both contexts?

Outline

- CCN Network Names
- Name Translation and Its Implications
- Translation Analysis
- Related & Future Work
CCN Network Names

Goal: translate application names into a format that:
• Facilitates standard network processing (exact match and LPM searches)
• Removes arbitrarily long names from the network
• Removes all location-irrelevant information from the name (as seen by routers)

Cryptographic Hash Function?

Translation should:
- be a bijection or very close to one
- map arbitrarily long strings to fixed-length output
- only apply to location-specific parts of a name.

Name Translation Example

T_NAME locator components possibly contained in FIBs

/Applicationspecific components not Contained in FIBs

Unique Name Fingerprint

T

N_p=H’/a/b/foo/version=0x00/chunk=0x01/PID=0x02

ACM ICN 2016
**Name Encoding**

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_NAME</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>T_MAPPED_SHA256/32</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>H(a), H(a, b), H(a, b, foo)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_PID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 (silly example)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/ name fingerprint = 0xFF &gt;</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>T_VERSION</td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td>0x00</td>
<td>T_CHUNK</td>
</tr>
<tr>
<td></td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T_PID</td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td>0x02</td>
<td></td>
</tr>
</tbody>
</table>

**Application Processing**

- **Consumer:**
  - Maps application names to network names for outgoing interests
  - Inverts mapping for incoming content

- **Producer:**
  - Pre-computes network names for all its locator prefixes
    - Stores in “inversion table”
  - Looks up content corresponding to incoming interests based on this inversion table
  - Signed content objects contain $N_p$
    - might also carry app name
Current Network Processing

Interest pkt
(1) decode_pkt
(2) decode_name
(3) compute_hash x n
(4) HT_lookup x (2+n)

We obviate the need to hash in order to perform: FIB, CS and PIT lookups

Processing Summary

<table>
<thead>
<tr>
<th>Entity</th>
<th>Impact:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer</td>
<td>Increased online processing</td>
</tr>
<tr>
<td>Producer</td>
<td>Increased offline computation &amp; storage of inversion table</td>
</tr>
<tr>
<td>Router</td>
<td>Faster FIB lookup with pre-computed name-prefix hashes</td>
</tr>
<tr>
<td></td>
<td>Faster PIT and CS lookups with N_p</td>
</tr>
<tr>
<td></td>
<td>(Potential) benefits due to fixed-size N_p</td>
</tr>
</tbody>
</table>
Analysis Setup

Questions:
- How big should a hash digest be?
  - What is the impact on packet sizes?
  - What are the resulting collision properties?
- What’s consumer processing overhead?

Unibas URI data set: [http://www.icn-names.net/](http://www.icn-names.net/)
Name Properties

<table>
<thead>
<tr>
<th>Number of segments $n$</th>
<th>Number of names</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13'952</td>
<td>0.002%</td>
</tr>
<tr>
<td>2</td>
<td>141'904</td>
<td>0.016%</td>
</tr>
<tr>
<td>3</td>
<td>71'327'647</td>
<td>8.190%</td>
</tr>
<tr>
<td>4</td>
<td>187'307'048</td>
<td>21.507%</td>
</tr>
<tr>
<td>5</td>
<td>253'652'065</td>
<td>29.148%</td>
</tr>
<tr>
<td>6</td>
<td>144'136'578</td>
<td>16.650%</td>
</tr>
<tr>
<td>7</td>
<td>93'837'904</td>
<td>10.775%</td>
</tr>
<tr>
<td>8</td>
<td>70'875'144</td>
<td>8.138%</td>
</tr>
<tr>
<td>9</td>
<td>25'611'059</td>
<td>2.941%</td>
</tr>
<tr>
<td>10</td>
<td>10'464'092</td>
<td>1.262%</td>
</tr>
<tr>
<td>11</td>
<td>3'973'961</td>
<td>0.456%</td>
</tr>
<tr>
<td>12</td>
<td>4'546'842</td>
<td>0.522%</td>
</tr>
<tr>
<td>13</td>
<td>1'206'905</td>
<td>0.139%</td>
</tr>
<tr>
<td>14</td>
<td>835'124</td>
<td>0.096%</td>
</tr>
<tr>
<td>15</td>
<td>841'552</td>
<td>0.097%</td>
</tr>
<tr>
<td>16</td>
<td>195'491</td>
<td>0.022%</td>
</tr>
<tr>
<td>17</td>
<td>121'486</td>
<td>0.014%</td>
</tr>
<tr>
<td>18</td>
<td>317'628</td>
<td>0.036%</td>
</tr>
<tr>
<td>19</td>
<td>168'228</td>
<td>0.019%</td>
</tr>
<tr>
<td>20</td>
<td>50'742</td>
<td>0.006%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>869'823'752</strong></td>
<td><strong>99.876%</strong></td>
</tr>
</tbody>
</table>

Name Size Impact (for interests)

Ratio of “network name” to “standard name” as size of $T()$/$H()$ grows.

ACM ICN 2016
**N_p Size Overhead (for Content)**

![Bar Chart: Average of overheads and Overhead compared to average name for 256-bit, 384-bit, and 512-bit sizes.]

**Collision Assessment (32-bit hash)**

![Bar Chart: Probability of collision for different number of prefix segments. Experimental and Analytical results are compared.]

ACM ICN 2016
Processing Overhead (consumer)

Per-name costs (µs):
• Average: **1,029.279** (≈1ms)
• Minimum: 3.812
• Maximum: 2,474.567
  → Reasonable compared to network I/O

Throughput (c/b):
• Average: **1,577.688**
• Minimum: 1,218.037
• Maximum: 3,494.538

Experimental setting:
• Intel 2.8 GHz Core i7
• Un-optimized implementation based on PARC Libparc libraries

Ideal setting:
• Use Intel intrinsics for hashing (~9 c/b)
• Work on wire-encoded packets

Wrap-up

• Motivated separating CCN application and network names
• A concrete mechanism for name translation function
• Assessed quality of name translation function and performance implications for all CCN entities
Related Work*

- CCN names
  - Requirements [Ghodsi et al., ICN’11]
  - Location-agnostic names [Van Adrichem et al., Nomen’13]

- Focus on FIB algorithm improvements
  - FIB algorithm modifications based on tries, hash tables, Bloom Filters, etc.
    [Quan et al., Networking’13], [Perino et al., ANCS’14],
    [So et al., ANCS’13], [Fukushima et al., Nomen’13]
  - Several rely on lexicographical name ordering
    - Our scheme breaks this!
  - Hop-by-hop optimizations, e.g., passing length of previously matched name in FIB

… but not on FIB inputs

*See paper for references

Future Work

- Compare performance of current FIB techniques with and without network names

- Play with various hash functions & sizes

- Explore further uses for translation function T()
/this/is/the/end/version=0x00/chunk=0x01/PID=0x02