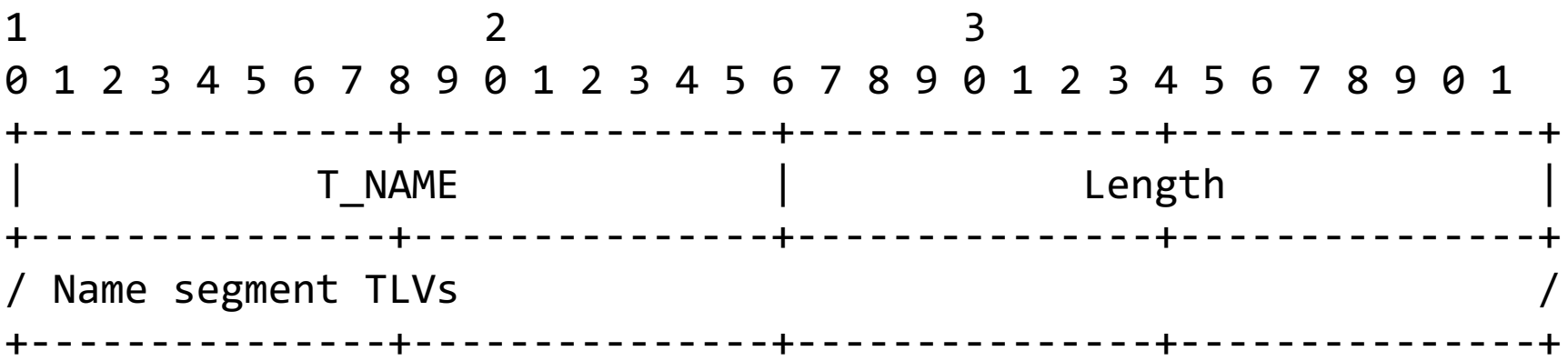
# Network Names in Content-Centric Networking

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

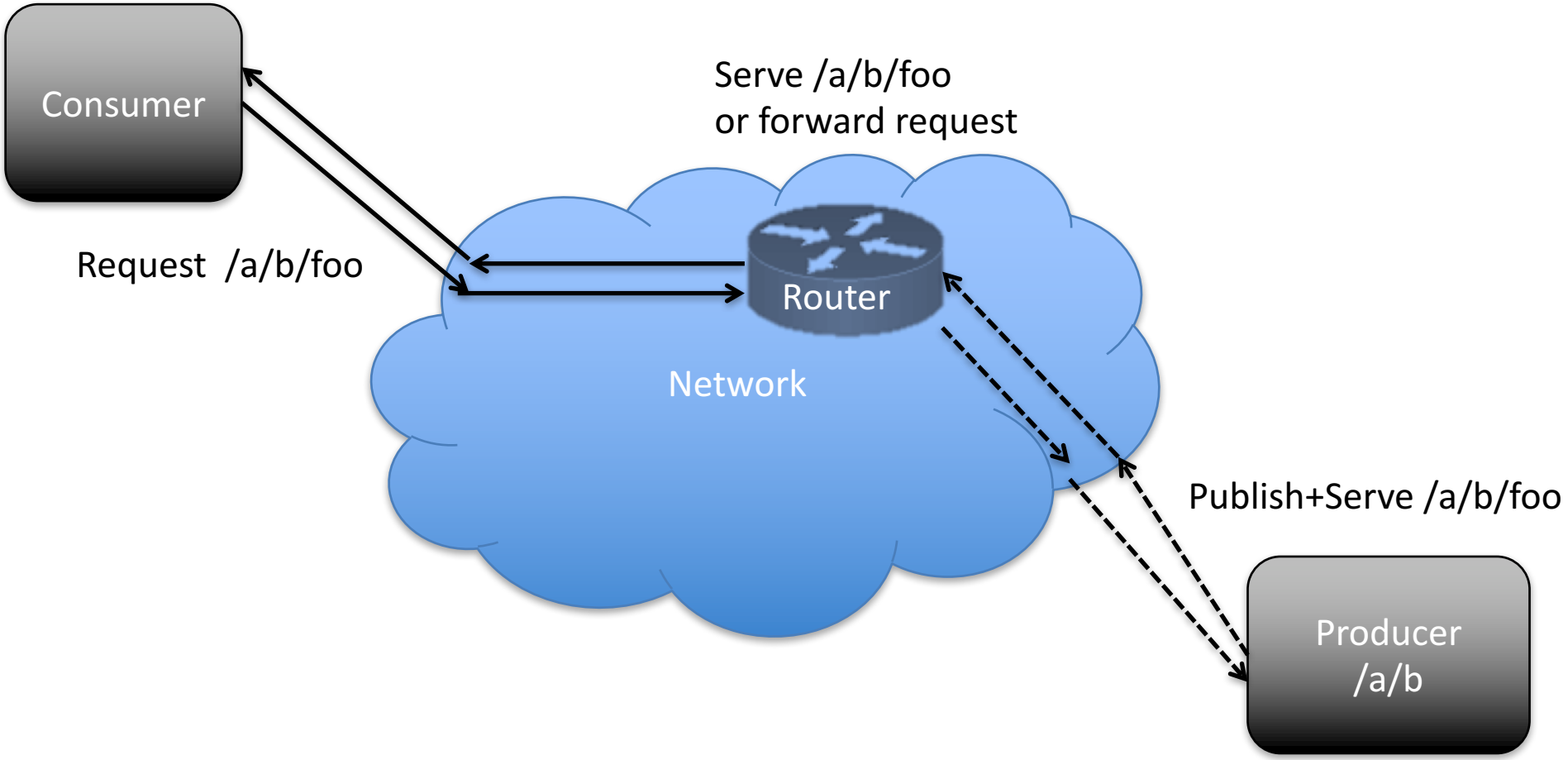
# CCN Names

- Expressed as URIs
  - /a/b/foo
  - /us/edu/uci/cs/tsudik/papers/acm-icn16.pdf

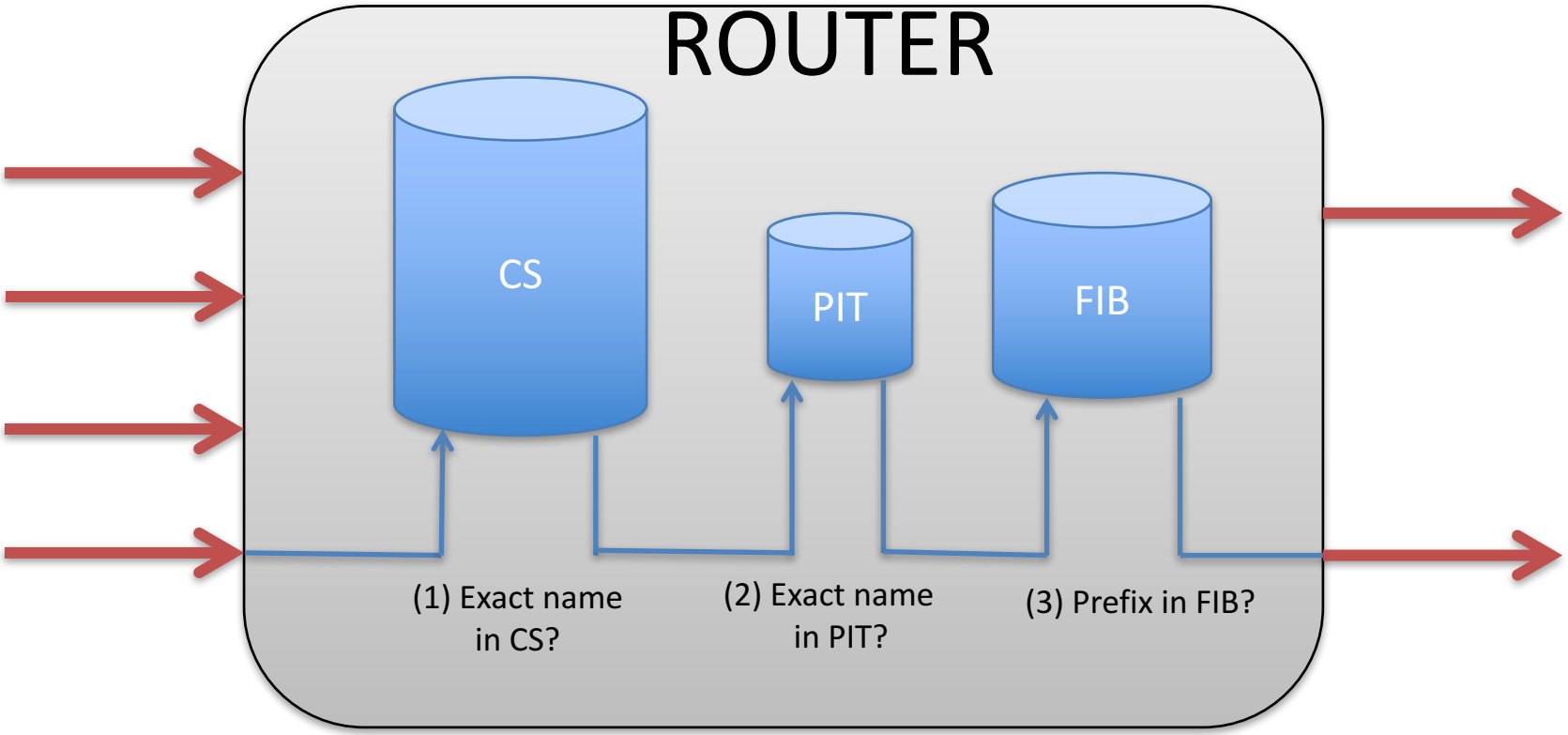
- Encoded as TLVs



# Names in CCN Applications



# Names in the Network



# 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?*

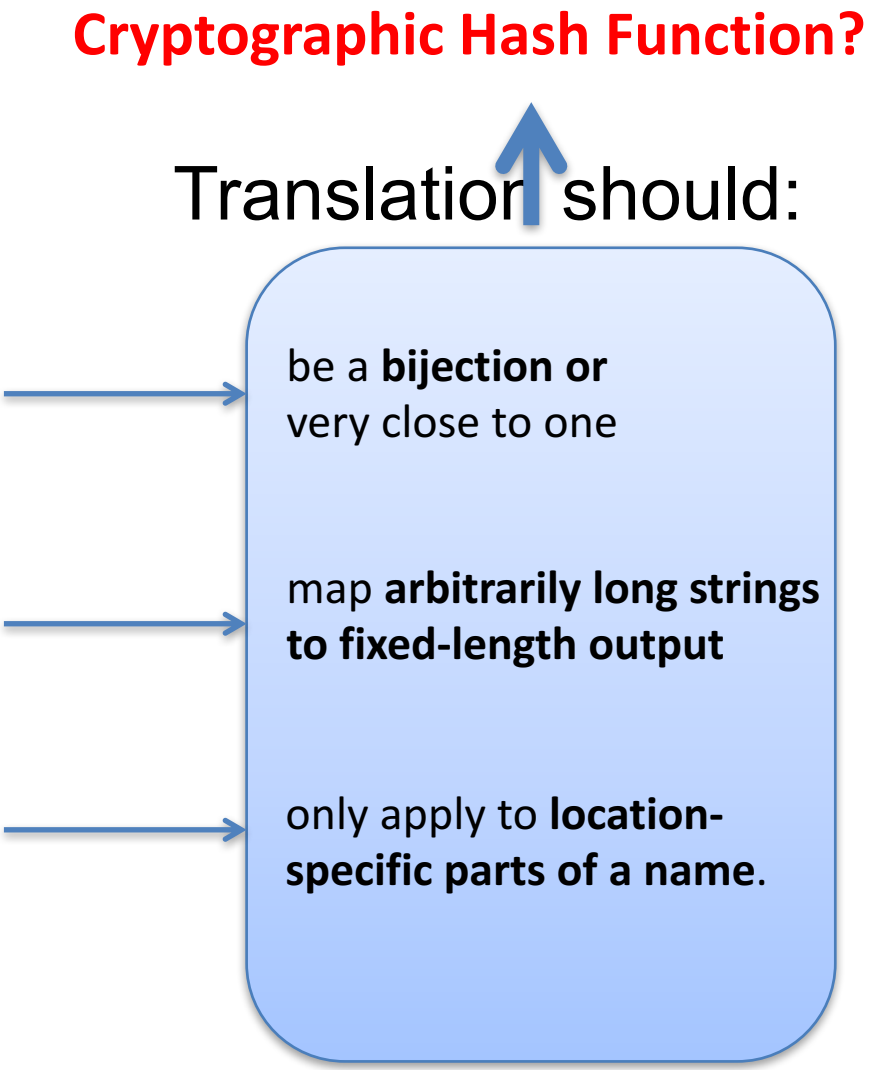
# Outline

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

# CCN Network Names

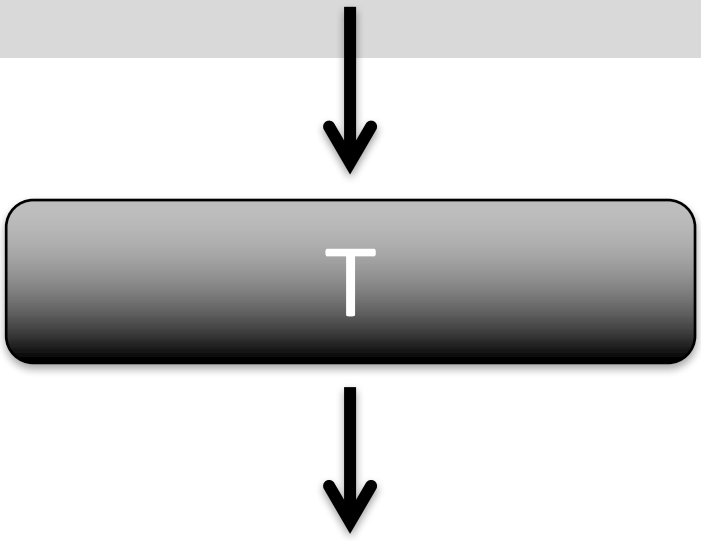
Goal: translate application names into a format that:

- 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)



# Name Translation Example

**/a/b/foo**/version=0x00/chunk=0x01/PID=0x02
payload: <bar>




Application-specific components not Contained in FIBs

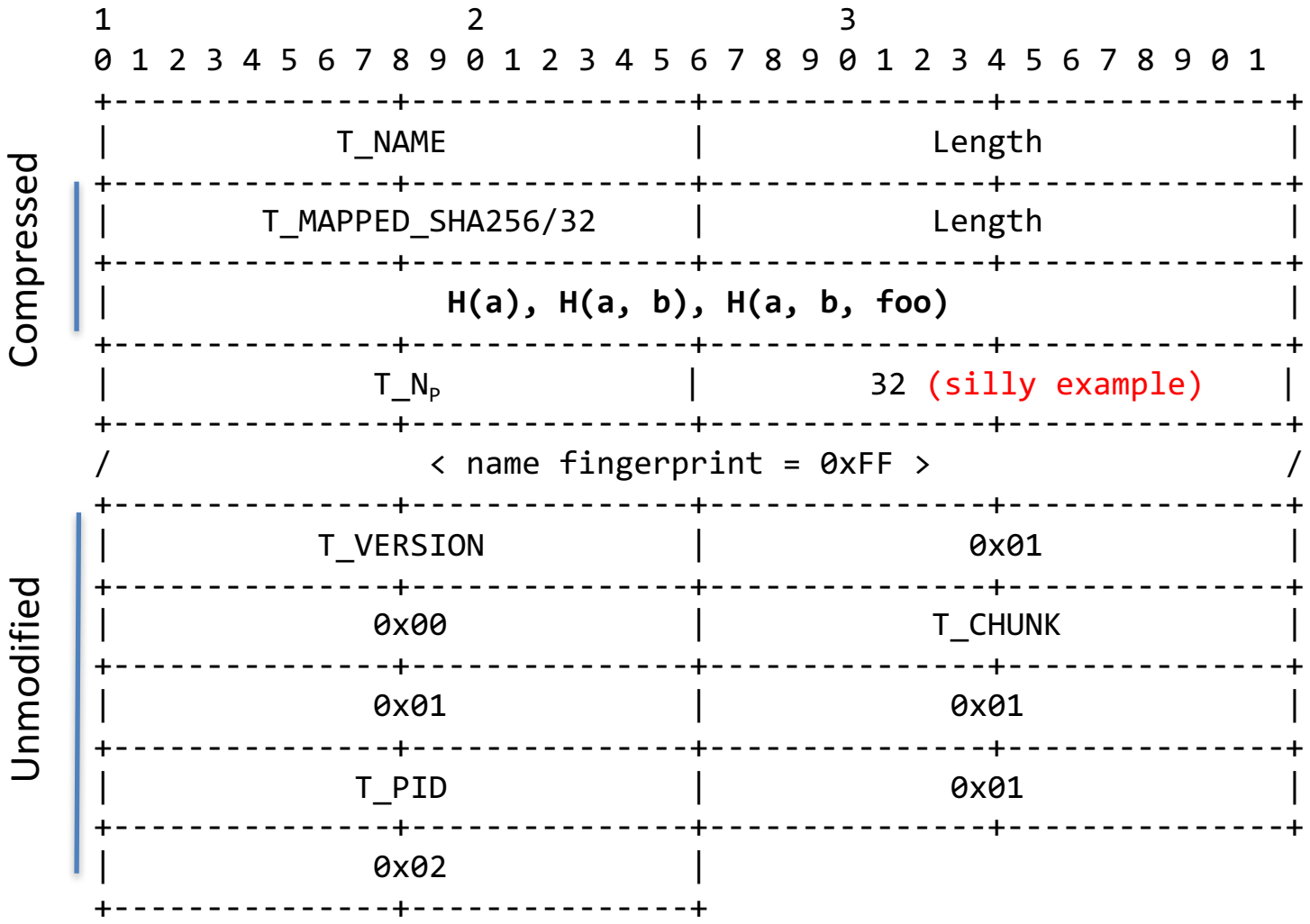
**T_NAME** locator components possibly contained in FIBs

Unique Name Fingerprint

`/H(a)/H(a,b)/H(a,b,foo)/version=0x00/chunk=0x01/PID=0x02/N_P=0xFF`
`payload: <bar>`

$N_P$=H'(**/a/b/foo**/version=0x00/chunk=0x01/PID=0x02)

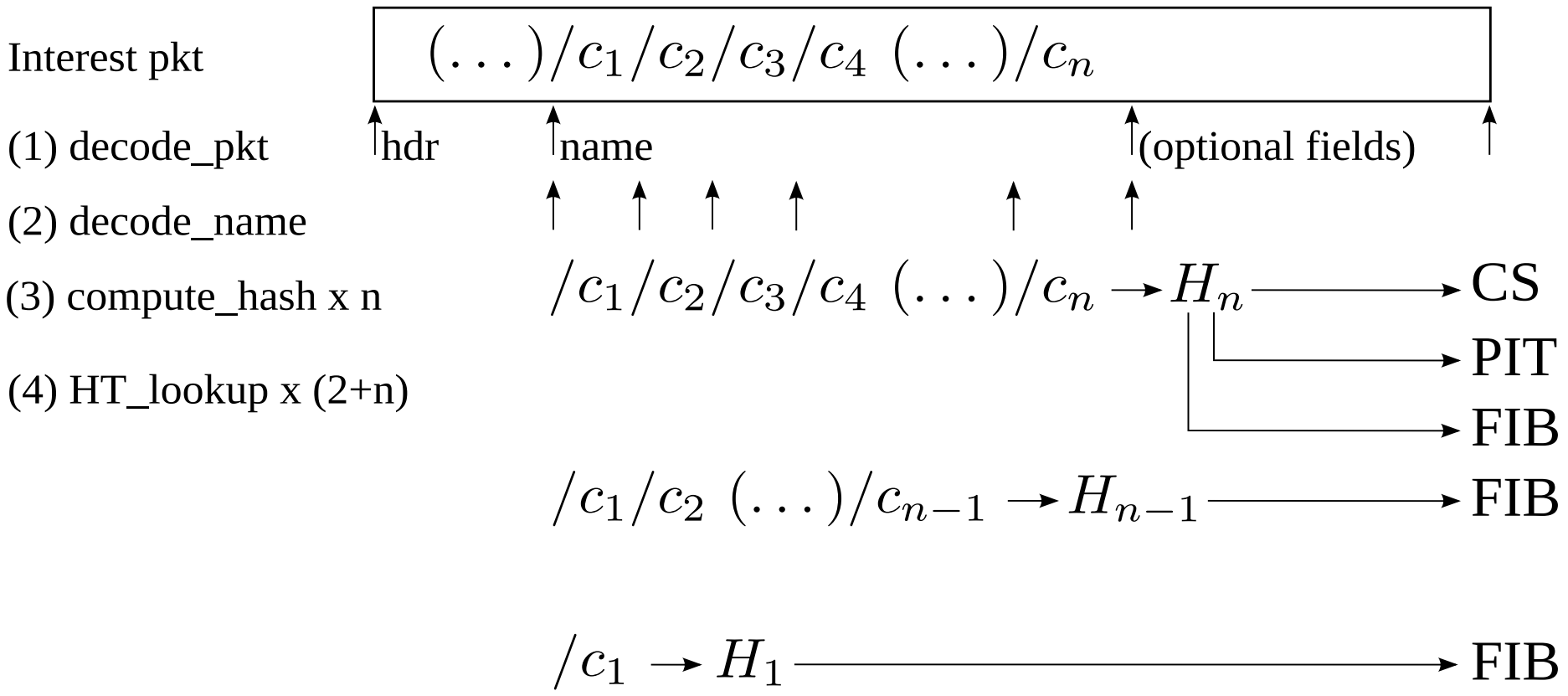
# Name Encoding



# 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 $\mathbf{N_p}$
    - might also carry app name

# Current Network Processing




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

# Processing Summary

| Entity: | Impact: |
| --- | --- |
| Consumer | **Increased** online processing |
| Producer | **Increased** offline computation & storage of inversion table |
| Router | **Faster** FIB lookup with pre-computed name-prefix hashes<br>**Faster** PIT and CS lookups with $\mathbf{N_P}$<br>**(Potential) benefits** due to fixed-size $\mathbf{N_P}$ |

# 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/

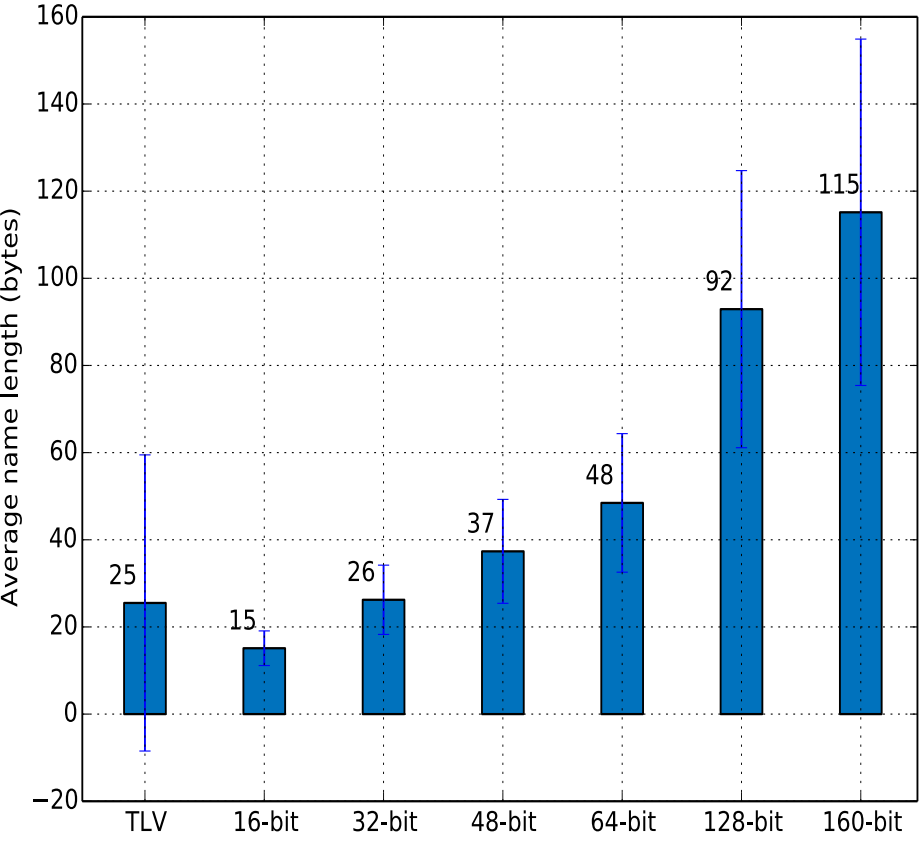
# Name Properties

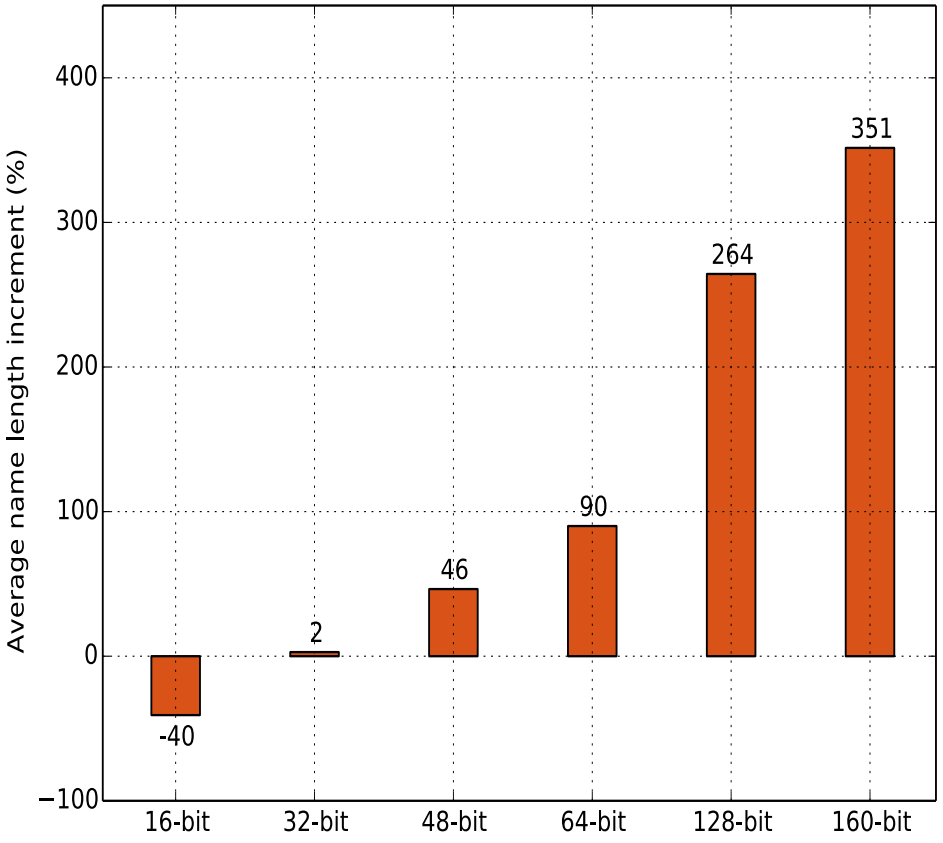
| Names | | Name segments | | Segments per Name | |
|---|---|---|---|---|---|
| Number of names | 870′896′633 | Total number of segments | 4′855′203′042 | Total number of segments | 4′855′203′042 |
| Average name length (bytes) | 57.95 | Average segment length (bytes) | 10.39 | Average segments per name | 5.57 |
| Name length standard deviation | 77.60 | Segment length standard deviation | 30.02 | Segments per name standard deviation | 8.14 |
| Minimum name length (bytes) | 1 | Minimum segments length (bytes) | 1 | Minimum segments per name | 1 |
| Maximum name length (bytes) | 764′867 | Maximum segments length (bytes) | 764′867 | Maximum segments per name | 210′658 |

# Name Properties

| Number of segments $n$ | Number of names | Percentage |
|---|---|---|
| 1 | 13′952 | 0.002% |
| 2 | 141′904 | 0.016% |
| 3 | 71′327′647 | 8.190% |
| 4 | 187′307′048 | 21.507% |
| 5 | 253′852′565 | 29.148% |
| 6 | 144′130′578 | 16.550% |
| 7 | 93′837′904 | 10.775% |
| 8 | 70′875′144 | 8.138% |
| 9 | 25′611′959 | 2.941% |
| 10 | 10′464′092 | 1.202% |
| 11 | 3′973′961 | 0.456% |
| 12 | 4′546′842 | 0.522% |
| 13 | 1′206′905 | 0.139% |
| 14 | 835′124 | 0.096% |
| 15 | 844′552 | 0.097% |
| 16 | 195′491 | 0.022% |
| 17 | 121′486 | 0.014% |
| 18 | 317′628 | 0.036% |
| 19 | 168′228 | 0.019% |
| 20 | 50′742 | 0.006% |
| **Total** | **869′823′752** | **99.876%** |

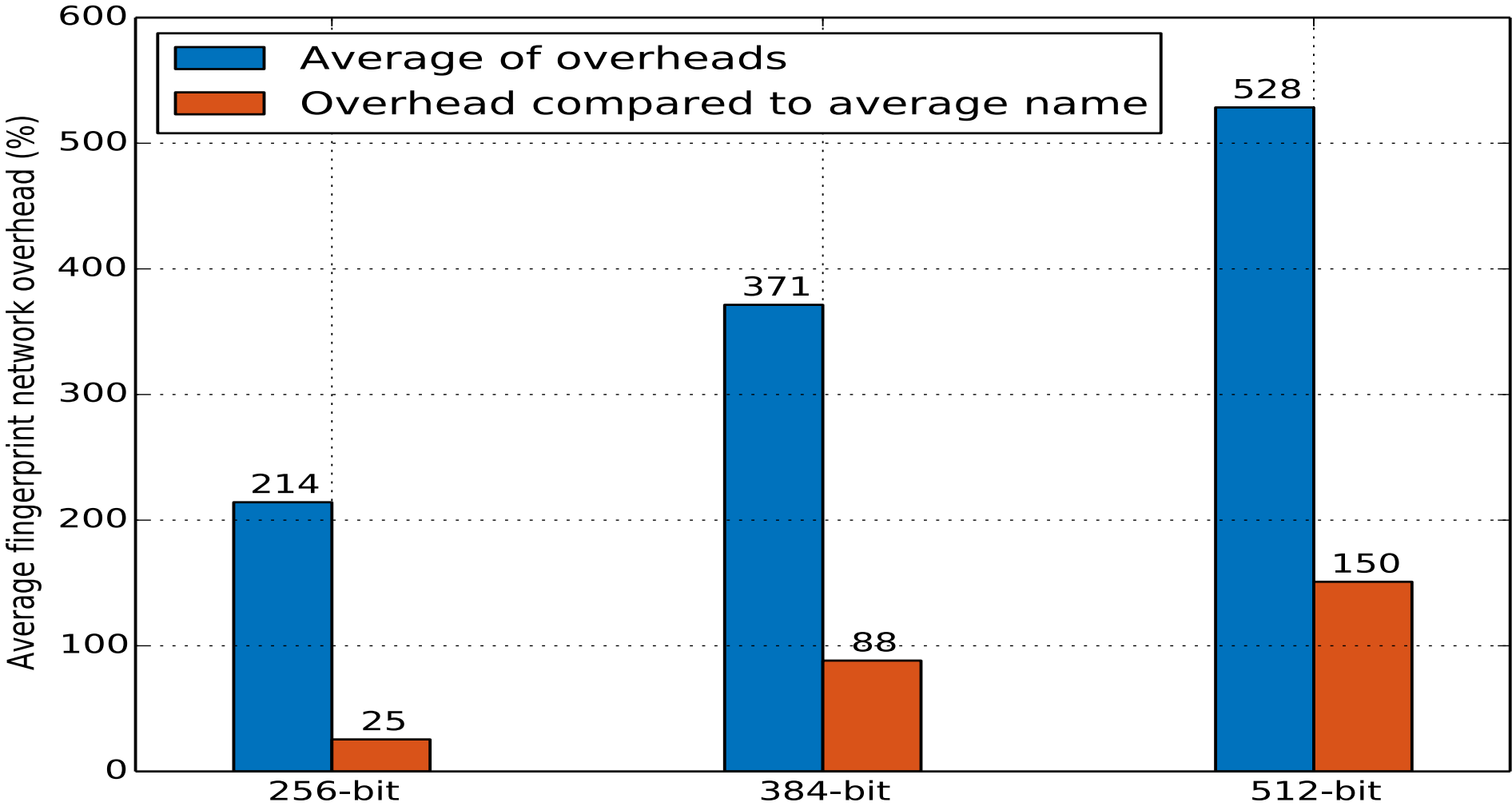
# Name Size Impact (for interests)



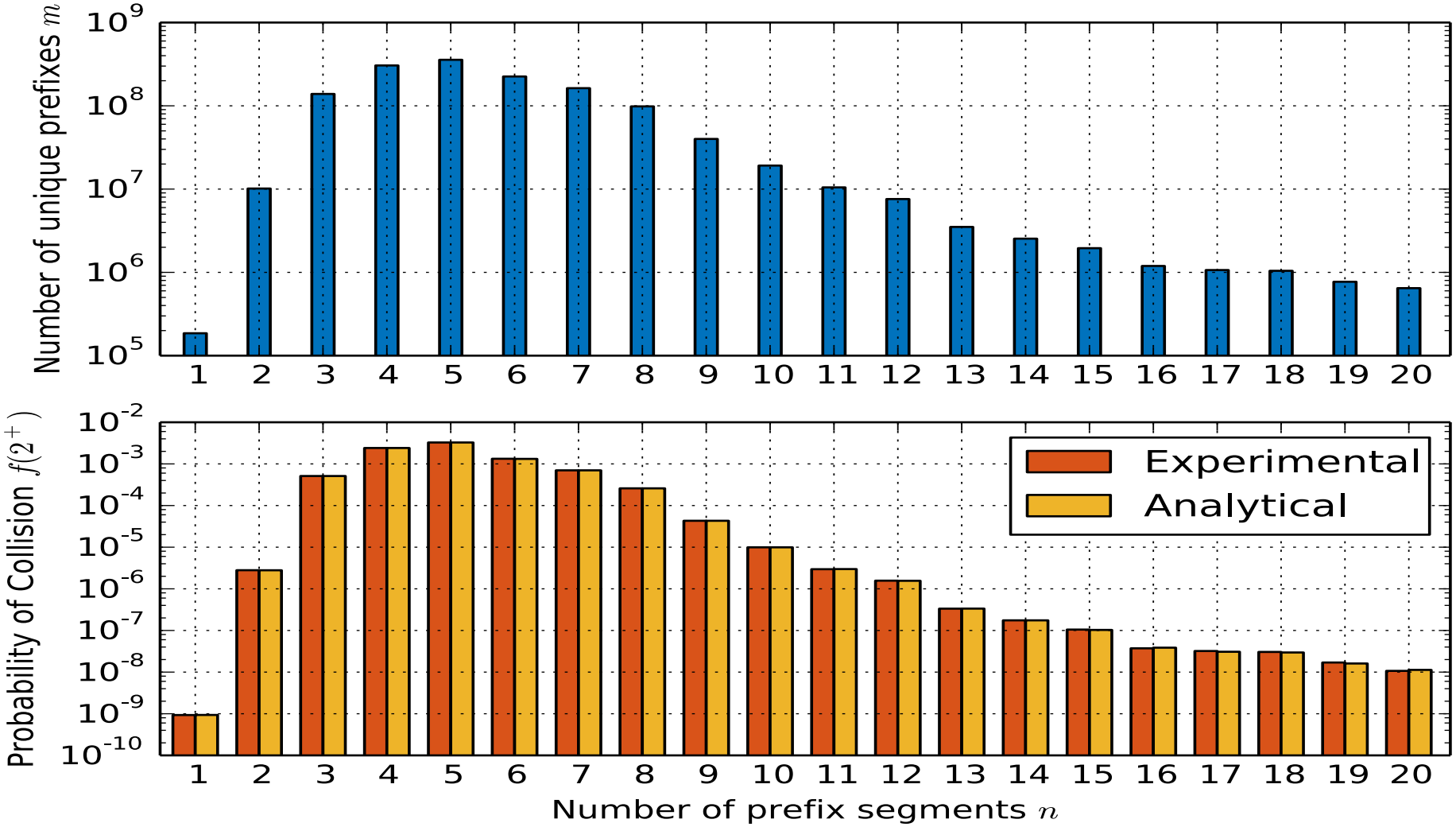




Ratio of “network name” to “standard name” as size of T()/H() grows

# $N_p$ Size Overhead (for Content)



# Collision Assessment (32-bit hash)



# 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