Discovering in-network Caching Policies in NDN Networks from a Measurement Perspective

Chengyu Fan (Colostate), Susmit Shannigrahi (Tennessee Tech), Christos Papadopoulos (Colostate), Craig Partridge (Colostate)
NDN requires measuring in-network states

- Network measurement tools cover various aspects in IP networks
  - Network performance, states (routing, configurations, and topology, etc.), and traffic

- NDN measurements must capture in-network states
  - Caching policies, forwarding strategies, etc.
Goals and Assumptions

• Our goal: **first work** to detect caching decisions from a measurement perspective
  – Caching is a central feature of NDN
  – Caching policy = caching decision + cache replacement
  – Multiple caching decisions may exist in NDN networks, and they may interact poorly

• Assumptions
  – The best-route forwarding strategy and uniform caching decision policy are used
  – Priority-FIFO cache replacement policy is used (by default)
  – Only one producer exists
List of caching decisions developed for NDN

- Caching Everything Everywhere (CEE)
  - Cache every Data chunk locally

- Leave Copy Down (LCD)
  - Move down the cached copy one hop down

- Label-caching
  - Pre-decide assign labels to routers, caching chunks whose ID%N match the label value

- Static probabilistic caching (Prob-20, Prob-50, Prob-80)
  - Pre-define the probability value, and compare it with the generated random number for each chunk

- Dynamic probabilistic caching (ProbCache, ProbCache-inv)
  - Dynamically calculate a cache weight based on the ratio of hop count of Data and Interest
Measurement procedure

Pre-defined the target name prefix, Data payload size, and other Data packet parameters

1. Send out a train (50) of Interests with the given name prefix
   • Each contains a unique name: /<name-prefix>/<chunk-id>

2. Answer each Interest

3. Save the hop count for each chunk

4. Repeat step 1 ~ 3 for ten times (cached copy can satisfy duplicate request), and plot the hop count distribution in the figure
Example: LCD caching decision

- Leave Copy Down (LCD) caching decision mechanism
  - The requested chunks is cached only at the cache that below the location of the hit on the path
- Takeaways
  - All chunks are cached at specific hops in each round, and the hop count across rounds differs
Fingerprint of LCD mechanism

- Simulations with ndnSIM
  - A linear topology with 10 routers

- Two metrics uniquely identify a caching decision
  - Hop count distribution in one round
  - The distribution change across multiple rounds
Fingerprints for other mechanisms

- **Caching Everything Everywhere (CEE)** - cache every Data chunk locally
- **Label-caching** - cache chunks whose ID%N match the pre-assigned label value
- **Static probabilistic caching (Prob-X)** - compare the random number with pre-defined probability value

[Graphs showing hop counts and round to send requests for CEE, Label-caching, and Prob-50]
Fingerprints for probabilistic caching mechanisms

Dynamic probabilistic caching:
- Calculate a weight based on the ratio of the hop count of Data and Interest

- ProbCache converges slower than prob-80

- ProbCache-inv converges faster than prob-20

- CacheWeight = \#hop(Data) / \#hop(Interest)
- Rand() < CacheWeight
- Rand() < (1 - CacheWeight)
Cross traffic may hurt measurements

- Cross traffic may exist in networks
  - Occupy cache slots
  - Evict cached probe packets
  - Impact detecting caching policies

- We check the effects by introducing cross traffic at two ends of the linear topology
Robustness to cross traffic

- We can identify the caching mechanisms, as most plot shapes are almost unchanged
Robustness to cross traffic (cont.)

- Shapes for LCD are changed, but it has the unique feature

![Client-side cross traffic](image1)

![Server-side cross traffic](image2)
Estimate static probabilistic value

Shapes based on simulation results

Ideal shapes for 1st round

Shapes based on simulation results
Detecting on real topology

- The NDN stack does not expose the hop count information to applications
  - Can we use delays to infer the correct hops?
- Use topology Rocketfuel 7018 with randomly chosen client and server

Delays may produce “correct” hop counts
Delays do not always infer the correct hops

- In some cases, link delays may not be identical with hops
Estimating hop counts

- Using clustering algorithms (e.g. K-means) to group samples with similar delays
  - The figures approximately show the correct shapes
Conclusion

• Proposed a novel method to extract fingerprints for caching decision mechanisms
• The method can detect caching decisions mechanisms from end hosts
  – Not sensitive to cross traffic
  – Can estimate probability value
• Evaluated the method on a real topology
  – Applications use delays to estimate hop counts
Future work

• Evaluate the method with more caching mechanisms on a real testbed (i.e. NDN testbed)
• Study the robustness of our method with other cache replacement policies
• Integrate the measurement tool with the NDN measurement framework designed by NIST [1] [2]
• Study the scenarios where multiple producers exist and other forwarding strategies are used
Thanks!