Neighborhood Search and Admission Control in Cooperative Caching Networks

Walter Wong*, Liang Wang†, Jussi Kangasharju†‡

*School of Electrical and Computer Engineering, University of Campinas, Brazil
†Department of Computer Science, University of Helsinki, Finland
‡Helsinki Institute for Information Technology, University of Helsinki, Finland
Contents

Motivation
Architecture Design
Caching Strategies
Neighbor Search
Evaluation
Conclusion
Motivation

Fast grow of user-generated content

According to the Cisco survey, the global IP traffic is expected to grow four times from 2009 to 2014. This puts the current Internet infrastructure under high burden. Content generated once, but consumed many times.
Motivation

Middle-mile problem

The infrastructure that interconnects the transit points between different ISPs
Motivation

Improve network efficiency by:
- Reduce the redundant data transfer;
- Provide an extended life for the middle mile infrastructure;

Approach
- Place network-level routers (“Content Routers”) in the network to store popular content
- Implement cooperative look-up between caches
– An exp. how the content routers work in a network topology.
– The CRs use the basic store-n-forward model.
An exp. how the content routers work in a network topology.

The CRs use the basic store-n-forward model.
When the response travels back to the client, every router it passes by will cache the content.
- When the response travels back to the client, every router it passes by will cache the content
– Later, Client 3, maybe on the other side of the network, same content may be requested by different clients.
– Later, Client 3, maybe on the other side of the network, same content may be requested by different clients.
Architecture

Client 1
Client 2
Client 3

Response

Network caches

Web Server
Architecture

Basic store-n-forward model
- Store everything passes by
- Simple to implement
- Limitations - low performance & low utilization of storage
Architecture

Basic store-n-forward model
- Store everything passes by
- Simple to implement
- Limitations - low performance & low utilization of storage
Architecture

Basic model’s limitation is due to lacking of good caching strategies

A good caching strategy should:

- maximize the utilization of network caches
- keep it simple
Caching Strategies

A Caching strategy consists of 3 parts

Admission policy - what to store?

Replacement policy - what to evict?

Cooperation policy - where to search?
Neighbor Search Caching Strategy

Two admission policies - ALL & Cachedbit
- ALL - cache everything passes by
- Cachedbit - cache based on probability

One replacement policy - LRU

One cooperation policy - Neighbor Search
Neighbor Search Caching Strategy - Admission Policy

**ALL** caches everything everywhere

**Cachedbit** is probabilistic

- Each router caches a chunk with uniform prob.
- Set bit in header $\rightarrow$ No caching downstream
Neighbor Search Caching Strategy - Admission Policy

ALL caches everything everywhere
Cachedbit is probabilistic
Each router caches a chunk with uniform prob.
Set bit in header → No caching downstream
Neighbor Search Caching Strategy - Admission Policy

**ALL**
ALL caches everything everywhere

**Cachedbit**
Cachedbit is probabilistic
Each router caches a chunk with uniform prob.
Set bit in header → No caching downstream
Neighbor Search Caching Strategy - Admission Policy

**ALL** caches everything everywhere

**Cachedbit** is probabilistic
- Each router caches a chunk with uniform prob.
- Set bit in header  →  No caching downstream
Neighbor Search Caching Strategy - Admission Policy

**ALL** caches everything everywhere

**Cachedbit** is probabilistic

- Each router caches a chunk with uniform prob.
- Set bit in header → No caching downstream
Neighbor Search Caching Strategy - Admission Policy

**ALL** caches everything everywhere

**Cachedbit** is probabilistic

Each router caches a chunk with uniform prob.
Set bit in header → No caching downstream
Neighbor Search Caching Strategy - Admission Policy

ALL caches everything everywhere

Cachedbit is probabilistic
- Each router caches a chunk with uniform prob.
- Set bit in header → No caching downstream
Neighbor Search Caching Strategy - Admission Policy

ALL caches everything everywhere

Cachedbit is probabilistic

Each router caches a chunk with uniform prob.
Set bit in header \(\rightarrow\) No caching downstream
Neighbor Search Caching Strategy - Cooperation Policy

Exchange information with neighbors
  Maintain neighbors’ states
  Frequency-based update
    – Redundant messages if traffic dynamics is low
    – Need to find a proper broadcast frequency
  Content-based update
    – A proper threshold can reduce overheads
Use Bloom Filter to reduce communication overheads
Neighbor Search Caching Strategy - Example

**Neighborhood Table**

<table>
<thead>
<tr>
<th>cryptoID</th>
<th>CR ID</th>
<th>Time</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1..C1</td>
<td>2</td>
<td>23</td>
<td>IP₂</td>
</tr>
<tr>
<td>A1..EE</td>
<td>5</td>
<td>55</td>
<td>IP₅</td>
</tr>
<tr>
<td>FF..E3</td>
<td>6</td>
<td>5</td>
<td>IP₄</td>
</tr>
</tbody>
</table>

Client 1 → CR₁ → CR₃ → CR₅ → Client 2

Client 2 → CR₅

Client 2 → CR₆ → Web Server

Client 3 → CR₅
Neighbor Search Caching Strategy - Example

![Diagram of network nodes and connections]

<table>
<thead>
<tr>
<th>cryptoID</th>
<th>CR ID</th>
<th>Time</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1..C1</td>
<td>2</td>
<td>23</td>
<td>IP_2</td>
</tr>
<tr>
<td>A1..EE</td>
<td>5</td>
<td>55</td>
<td>IP_5</td>
</tr>
<tr>
<td>FF..E3</td>
<td>6</td>
<td>5</td>
<td>IP_4</td>
</tr>
</tbody>
</table>

Neighborhood Table
Evaluation - Topology

Evaluated on realistic ISP’s network topologies

The topology file is from Rocketfuel project

Both router-level topology and POP-level topology

Router-level exp has better performance due to the longer path

Results are consistent

<table>
<thead>
<tr>
<th>Network</th>
<th>Routers</th>
<th>Links</th>
<th>POPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exodus</td>
<td>338</td>
<td>800</td>
<td>23</td>
</tr>
<tr>
<td>Sprint</td>
<td>547</td>
<td>1600</td>
<td>43</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>733</td>
<td>2300</td>
<td>108</td>
</tr>
<tr>
<td>NTT</td>
<td>1018</td>
<td>2300</td>
<td>121</td>
</tr>
</tbody>
</table>
Evaluation - Experiment Design

Server placement - top-20 nodes with highest degree

Client placement - rest of the nodes

We use software routers to construct an overlay on top of a computing cluster
Evaluation - Trace & Traffic Pattern

Use both realistic trace and synthetic trace

Population follows Zipf distribution

$$f(k; \alpha, N) = \frac{1/k^\alpha}{\sum_{n=1}^{N} (1/n^\alpha)}$$

Realistic trace is from university lab, \(\alpha\) value is 0.93
Synthetic trace - use 0.7, 0.9 and 1.1

Traffic pattern - constant and gravity model

Constant - traffic is homogenous from all the clients
Gravity model - amount of traffic based on the population

Traffic pattern - constant and gravity model
Evaluation - Metrics

Hit rate
How much inter-ISP traffic we can reduce
One packet represents one file object
Hit rate is equivalent to the byte hit rate

Avg. hops
Measure the content locality
Locality represents how close the requested content is to the clients
Evaluation - Metrics

Footprint reduction

How much intra-ISP traffic we can reduce
How many bytes did not go on how many hops?

Example:

N hops to egress, cache hit on 1st hop
Traffic without caching is \( N \times \text{content} \_\text{size} \)
With caching traffic is \( 1 \times \text{content} \_\text{size} \)
Hence, reduction is \( (N-1) \times \text{content} \_\text{size} \)
Footprint reduction: \( (N-1) / N \)
Evaluation - Hit Rate

Sprint, $\alpha=0.9$

- ALL
- Cachedbit
- NbSA
- NbSC

Effect of admission policy

Effect of search

Cache Size

Hit Rate
Main lessons:

- As admission policy, LRU is the worst in all the cases

- Neighbor Search gives a boost in hit rate at a small cost

- Good admission policy is still a must

- The difference varies on different topologies, but consistent
Footprint reduction

How much intra-ISP traffic we can reduce

Large reduction means less intra-ISP traffic
Evaluation - Footprint Reduction

Main lessons:

NBS* might not perform well for small caches
  – the neighbors are unlikely to have the content if a miss happens
  – searching actually causes extra overheads for small cache

Neighbor Search improves quickly as the cache size grows

NbSC is the best strategy in all cases
Evaluation - Locality

Avg. hops

Measure how close the requested content is to the clients

– We see the same behavior in avg. hops as that in footprint reduction
In terms of hit rate, larger radius only gives marginal improvement.

In terms of footprint reduction, larger radius increases intra-ISP traffic, and also increases user latency. The request can go too far.
– In terms of hit rate, larger radius only gives marginal improvement
– In terms of footprint reduction, larger radius increases intra-ISP traffic, and also increases user latency. The request can go too far.
– Large FP rate won’t hurt hit rate too much
– Large FP rate hurts footprint reduction. Requests can be routed further because a router thought his neighbor has the content
Neighbor Search Caching Strategy - Parameters

Key parameters:
- Search radius: 1 hop is enough, more hurts network traffic
- False positive rate: 1% is enough

Main lessons learned:
- Searching neighbors is highly beneficial
- Need admission policy as well
Conclusion & Future Work

Conclusion

Good caching strategy plays an important role in In-network caching performance.

Good admission policy helps a lot

Neighbor Search boosts the performance

Future work

Integration to CCNx prototype.
Thanks!

Questions?