Asymmetric Caching: Improved Network Deduplication for Mobile Devices

Shruti Sanadhya,¹ Raghupathy Sivakumar,¹ Kyu-Han Kim,²
Paul Congdon,² Sriram Lakshmanan,¹ Jatinder P Singh³

¹Georgia Institute of Technology, Atlanta, GA, USA
²HP Labs, Palo Alto, CA, USA
³Xerox PARC, Palo Alto, CA, USA
Introduction

- Network traffic has a lot of redundancy
  - 20% HTTP content accessed on smartphones is redundant\(^1\)

- Network deduplication (dedup) leverages this redundancy to conserve network bandwidth

\(^1\) Qian et al., “Web Caching on Smartphones: Ideal vs. Reality”, MobiSys 2012
The Asymmetry Problem

- What happens when the mobile cache is more populated than the cache at dedup source?

How can all the past cached information at the mobile be successfully leveraged for dedup by any given dedup source?
Motivational Scenarios

- Multi-homed devices
Motivational Scenarios

• Multi-homed devices
• Resource pooling

– BS: Base Station
– RNC: Radio Network Controller
– SGSN: Serving GPRS Support Node
Motivational Scenarios

- Multi-homed devices
- Resource pooling
- Memory scalability

- BS: Base Station
- RNC: Radio Network Controller
- SGSN: Serving GPRS Support Node
Scope and Goals

• Scope
  – Laptops/smartphones using 3G/WiFi
  – Conserving cellular bandwidth
  – Downstream and unencrypted traffic

• Goals
  – Overall efficiency: Using downstream and upstream more efficiently
  – Application agnostic: Applicable to any application
  – Limited overheads: Deployable computational and memory complexities
Mobile cache is more populated than dedup source
On receiving downstream traffic, the mobile selectively advertises portions of its cache to dedup source
Dedup source also maintains a feedback cache
Both regular and feedback cache is used for dedup
When is feedback sent?

- Feedback is sent *reactively*
- Feedback is sent only when there is downstream traffic
- Feedback sent is specific to the ongoing traffic
Where from is feedback selected?

- Hashes at dedup destination can be organized as per:
  - Order of arrival
  ```
  H1  H2  H3  H4  H5  H6  H7  H8  H9  H10
  ```
  - Same flow
    (Src IP, Dest IP, Src Port, Dest Port)
  ```
  H1  H2  H6  H7  H8
  H3  H4  H5  H9  H10
  ```
  - Same object
    (HTML, JPEG or CSS)
  ```
  H1  H2
  H6  H7  H8
  H3  H4  H5
  H9  H10
  ```

- Objects help in effectively matching new and old content
- Application agnostic estimate of objects are flowlets
How are flowlets extracted?

- Sequence of bytes in a flow is a time-series
- *Flowlets* are piecewise stationary segments of a flow
- Check for flowlet boundary at start of each packet
- Consider byte series $B_{[0:m]}$ (1st packet), $B_{[m+1:n]}$ (2nd packet) and $B_{[0:n]}$ as autoregressive processes of order $p$:

  $$B_i = \sum_{1\leq j\leq p} a_i B_{i-j} + \sigma \varepsilon , \quad \varepsilon \text{ is white noise}$$

- $d_{[0:m:n]} = \text{gain}(B_{[0:n]}) - \text{gain}(B_{[0:m]}) - \text{gain}(B_{[m+1:n]})$

  Gain in the noise power when $B_{[0:n]}$ is in one flowlet instead of different flowlets: $B_{[0:m]}$ and $B_{[m+1:n]}$

- If $d_{[0:m:n]} > d_{\text{thresh}}$, then flowlet boundary exists at $m$
How is feedback selected?

- Find best matching past flowlet
How is feedback selected?

- Find best matching past flowlet
How is feedback selected?

- Find best matching past flowlet

<table>
<thead>
<tr>
<th></th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>F1, F3</td>
<td></td>
</tr>
<tr>
<td>H2</td>
<td>F1, F2, F3</td>
<td></td>
</tr>
</tbody>
</table>
How is feedback selected?

• Find best matching past flowlet

<table>
<thead>
<tr>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>F1, F3</td>
</tr>
<tr>
<td>H2</td>
<td>F1, F2, F3</td>
</tr>
</tbody>
</table>

F1: H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, ....
F2: H2, H5, H10, ....
F3: H5, H8, H11, H12, .....  

Last hash matched
How is feedback selected?

• Find best matching past flowlet

<table>
<thead>
<tr>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>F1, F3</td>
</tr>
<tr>
<td>H2</td>
<td>F1, F2, F3</td>
</tr>
</tbody>
</table>

F1: H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, ....

F2: H2, H5, H10, ....

F3: H5, H8, H11, H12, .....  

Last hash matched
How is feedback selected?

- Find best matching past flowlet

<table>
<thead>
<tr>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>F1, F3</td>
</tr>
<tr>
<td>H2</td>
<td>F1, F2, F3</td>
</tr>
</tbody>
</table>

F1: H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, ....
F2: H2, H5, H10, ....
F3: H5, H8, H11, H12, ....

Flowlet 1 (F1) is best matched
How is feedback selected?

- Find best matching past flowlet
  - Flowlet 1 (F1) is best matched

- Find start of next feedback in the best matching flowlet
How is feedback selected?

• Find best matching past flowlet

<table>
<thead>
<tr>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>→</td>
</tr>
<tr>
<td>H2</td>
<td>→</td>
</tr>
</tbody>
</table>

F1: H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, ....
F2: H2, H5, H10, ....
F3: H5, H8, H11, H12, .....  

– Flowlet 1 (F1) is best matched

• Find start of next feedback in the best matching flowlet

Best matching past flowlet

H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, H13, ....
How is feedback selected?

- Find best matching past flowlet

- Flowlet 1 (F1) is best matched

- Find start of next feedback in the best matching flowlet
How is feedback selected?

- Find best matching past flowlet
  
  \[ \begin{array}{c|c}
  H1 & H2 \\
  \hline
  H1 & F1, F3 \\
  H2 & F1, F2, F3 \\
  \end{array} \]

  \[ \text{F1: } H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, \ldots \]

  \[ \text{F2: } H2, H5, H10, \ldots \]

  \[ \text{F3: } H5, H8, H11, H12, \ldots \]

  \[ \text{Last hash matched} \]

- Flowlet 1 (F1) is best matched

- Find start of next feedback in the best matching flowlet

\[ \text{Best matching past flowlet } H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, H13, \ldots \]

\[ \text{Last hash matched} \]

\[ \text{Last hash advertised} \]
How is feedback selected?

- Find best matching past flowlet
  - **Flowlet 1** (F1) is best matched

- Find start of next feedback in the best matching flowlet
  - $\delta$: temporal offset

  | F1 | H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, ....   |
  | F2 | H2, H5, H10, ....                                   |
  | F3 | H5, H8, H11, H12, .....                             |

  Last hash matched

  Last hash advertised
How is feedback selected?

• Find best matching past flowlet

<table>
<thead>
<tr>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>F1, F3</td>
</tr>
<tr>
<td>H2</td>
<td>F1, F2, F3</td>
</tr>
</tbody>
</table>

- Flowlet 1 (F1) is best matched

• Find start of next feedback in the best matching flowlet

Best matching past flowlet

| H1, H2, H4, H5, H6, H7, H8, H9, H10, H11, H12, .... |

- δ: temporal offset
How is the feedback used?

- Dedup source maintains a *feedback cache* along with *regular cache* of baseline dedup
How is the feedback used?

- Dedup source maintains a *feedback cache* along with *regular cache* of baseline dedup
- Regular cache is populated by downstream data
How is the feedback used?

- Dedup source maintains a *feedback cache* along with *regular cache* of baseline dedup
- Regular cache is populated by downstream data
Dedup source maintains a feedback cache along with regular cache of baseline dedup.

Regular cache is populated by downstream data.

Feedback hashes are inserted in feedback cache.

How is the feedback used?
How is the feedback used?

- Dedup source maintains a *feedback cache* along with *regular cache* of baseline dedup
- Regular cache is populated by downstream data
- Feedback hashes are inserted in *feedback cache*

Every downstream packet is deduped using both *regular* and *feedback cache*
Design Summary

• When is the feedback sent?
• Where from is the feedback chosen?
• How are flowlets extracted?
• How is the feedback selected?
• How is the feedback used?

• Reactively
• Flowlets at dedup destination
• Stationarity properties
• Best matching flowlet and pointers in past flowlet
• Stored in the feedback cache for dedup
Trace Based Analysis

• Data collection
  – 25 laptop and 5 smartphone users over 3 months giving 26GB of unsecured downlink data
  – WiFi as well as 3G network
  – Packet sniffing through Wireshark and Tcpdump

• Trace analysis
  – Custom analyzer implemented in Python
  – Mimic mobility by splitting trace into two halves: past and present
  – Past trace populates the initial cache at the dedup destination. This is the data remembered from previous networks access
  – 30 random connections from the present create ongoing traffic
  – Dedup is performed using asymmetric caching
Trace Analysis Results - I

- Redundancy identified

\[
\frac{\text{# Redundant bytes found by asymmetric caching}}{\text{Actual # redundant bytes}} \times 100
\]

Asymmetric caching leverages significant portion of the achievable redundancy
Trace Analysis Results - II

- **Feedback efficiency**
  
  \[
  \frac{\text{# Bytes saved downstream}}{\text{# Bytes sent upstream}}\]

  *Split of total hits across the caches at dedup source*

*Asymmetric caching generates efficient and relevant feedback*
Related Work

• Network layer approaches

• Transport layer approaches
  – Zohar et al, “The power of prediction: cloud bandwidth and cost reduction”, *SIGCOMM 2011*

• Application layer approaches
  – Web browser caches and proxies
  – Content Distribution Networks (CDNs)
Conclusion and Future Work

• A dedup strategy that leverages past remembered on mobile devices to perform dedup at any dedup source

• Application agnostically estimate different objects in a flow by using stationarity properties of different content

• Trace analysis of 30 users shows that asymmetric caching:
  – Leverages 89% of achievable redundancy
  – Gives 6x feedback efficiency

• Prototype implementation on Linux desktop and Android smartphone with deployable overheads

• Future Work:
  – Upstream dedup, i.e. reduce redundant content sent upstream
  – Extending dedup to end-to-end encrypted traffic
  – Study energy impact of asymmetric caching on mobile devices
Questions?