Exploring Cross-Application Cellular Traffic Optimization with Baidu TrafficGuard

Zhenhua Li, Weiwei Wang, Tianyin Xu, Xin Zhong, Xiang-Yang Li, Yunhao Liu, Christo Wilson, and Ben Y. Zhao
China has the world’s largest population of mobile cellular users.
China’s mobile users have notably small cellular data plan.

- 69.1% have no more than 200MB.
- 31.80% have 0-30MB.
- 24.50% have 31-100MB.
- 16.00% have 101-200MB.
- 8.50% have 201-300MB.
- 3.40% have 301-500MB.
- 3.00% have 501-1GB.
- 0.0% have >1GB.

93.6% of China’s users have no more than 500MB of traffic plan per month.

Source: iiMedia Research
Traffic overage is pervasive in China.

59.1% of China’s cellular users overuse their data plan.

Source: iiMedia Research
Efforts towards optimizing mobile traffic

- **Application-specific** data compression

Flywheel: Google’s data compression proxy (NSDI’15)

→ transcode images to the WebP format
→ minify and gzip text content

50+% of mobile traffic reduction with moderate latency penalty.
Mobile web browser traffic is only a (small) portion of today’s cellular traffic.
Baidu TrafficGuard

- Optimizing cross-app cellular traffic
  - data compression
  - content validation
  - cross-app caching

- Filtering out unwanted traffic
  - malicious/ads/overnight traffic

- Two years of deployment experience
  - 10 millions of users & 0.2 millions of active users daily
  - support all Android 4.0+ devices
TrafficGuard Overview

Diagram showing Backend with Software middlebox and Proxy connected by HTTP. The TG mobile app communicates with the Backend via GET and RESP.
In China, HTTP requests dominate the mobile web requests (80%).

TrafficGuard Overview

① Handle all cellular traffic across all apps

② Filter out unwanted traffic (e.g., malicious)

TG mobile app
TG mobile app (client-side support)

- Customized VPN
  - based on TUN
  - manipulate layer-3 packets at user space
  - no root privilege required

- Cooperation w/ the backend
  - Local filter for malicious links
  - Value-based cross-app caching
Backend: web proxy + software middleboxes

Traffic reduction mechanisms:
- Traffic filtering*
- Cross-app caching*
- Content validation
- Data compression

*cooperate w/ the client-side TG mobile app
Backend: web proxy + software middleboxes

Traffic reduction mechanisms

- Traffic filtering*
- Cross-app caching*
- Content validation
- Data compression

*cooperate w/ the client-side TG mobile app
Data compression

- Characteristics of HTTP content types

<table>
<thead>
<tr>
<th>Content Type</th>
<th>% HTTP Requests</th>
<th>% HTTP Traffic</th>
<th>Size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Median</td>
</tr>
<tr>
<td>Text</td>
<td>49.0%</td>
<td>15.7%</td>
<td>0.2</td>
</tr>
<tr>
<td>Image</td>
<td>32.0%</td>
<td>71.0%</td>
<td>5.7</td>
</tr>
<tr>
<td>Octet-stream</td>
<td>10.0%</td>
<td>5.5%</td>
<td>0.4</td>
</tr>
<tr>
<td>Zip</td>
<td>8.1%</td>
<td>5.1%</td>
<td>0.5</td>
</tr>
<tr>
<td>Audio &amp; video</td>
<td>0.03%</td>
<td>2.6%</td>
<td>407</td>
</tr>
<tr>
<td>Other</td>
<td>0.87%</td>
<td>0.1%</td>
<td>0.3</td>
</tr>
</tbody>
</table>

*Text: HTML, CSS, JSON, XML, JS, etc
Text compression

- Text compression is not worthwhile.
  - very small in size (mean: 0.2KB)
  - reduce HTTP traffic by 1.4%
  - considerable computation overhead at both client and cloud side.

TrafficGuard does not compress text objects.
Image compression

• 71% of HTTP traffic are images, and 40% of these images are oversized.

• Challenges:
  → cannot transcode all the images to WebP (Flywheel’s approach) because not all existing apps support WebP
  → cannot use WebP as the transfer format as the client-side transcoding overhead is not acceptable
  → cannot simply rescale images which may distort UI layout and degrade UX
Image compression

- Adjusting the **quality** of the images by tuning QF (Quality Factors) of JPEG images

<table>
<thead>
<tr>
<th>Quality Factor (QF)</th>
<th>114KB (100%)</th>
<th>101KB (88%)</th>
<th>84KB (75%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QF 75</td>
<td>114KB (100%)</td>
<td>101KB (88%)</td>
<td>84KB (75%)</td>
</tr>
<tr>
<td>QF 60</td>
<td>101KB (88%)</td>
<td>84KB (75%)</td>
<td>70KB (61%)</td>
</tr>
<tr>
<td>QF 45</td>
<td>84KB (75%)</td>
<td>70KB (61%)</td>
<td>48KB (42%)</td>
</tr>
<tr>
<td>QF 30</td>
<td>70KB (61%)</td>
<td>48KB (42%)</td>
<td></td>
</tr>
<tr>
<td>QF 15</td>
<td>48KB (42%)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Image compression

- Adjusting the **quality** of the images by tuning QF (Quality Factors) of JPEG images

<table>
<thead>
<tr>
<th>QF</th>
<th>30</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>114KB (100%)</td>
<td>101KB (88%)</td>
<td>84KB (75%)</td>
</tr>
<tr>
<td>70KB (61%)</td>
<td>48KB (42%)</td>
<td></td>
</tr>
</tbody>
</table>

Median QF = 80
Image compression

• Adjusting the **quality** of the images by tuning QF (Quality Factors) of JPEG images

→ **categorize images into Large, Medium, Small & Tiny**
  - based on the Ziproxy standard

→ **tune QF to balance compression ratio and quality**
  - quality is measured by SSIM (Structural Similarity)

→ **transcode PNG & GIF to JPEG if possible**
  - **59.9%** of the images are in JPEG
  - **8-10** times smaller overhead than WebP transcoding

**25.4% of traffic is reduced by such image compression!**
Traffic filtering

- Two-lever filtering cooperated by both the client-side mobile app and the backend

- 29M malicious links
- 102M ads links
- 1M links most frequently visited
- 72%~78% malicious & ad links are filtered out locally.
Total cellular traffic reduction

36% of total traffic reduction

- Traffic Filtering: 9.95%
- Content Validation: 2.72%
- Image Compression: 25.35%
- Value-Based Web Caching: 5.16%
- Original: 0.93%
- HTTP: 0%
- HTTPS: 0%

0% of total traffic reduction in HTTP and HTTPS categories.

Y-axis: Traffic Reduction (0% - 1400%)
X-axis: Categories: Original, Traffic Filtering, Content Validation, Image Compression, Value-Based Web Caching
Per-user traffic saving

Reduce overage instances by 10.7 times!!

55% of our users save over a quarter

20% of our user save over a half
Latency penalty & battery overhead

• **End-to-end latency penalty**
  - worst case: 474ms
  - mean: **282ms** & median: **53ms**

• **Battery consumption**
  - negligible (93mW on average)
  - sometimes reduce battery consumption
Conclusion

• TrafficGuard achieves 36% of cellular traffic reduction across different apps without degrading user experiences.

• Baidu is working on integrating TrafficGuard with cellular carrier infrastructure, which can further reduce end-to-end latency.