On the Cost of Using Happy Eyeballs for Transport Protocol Selection

Giorgos Papastergiou†, Karl-Johan Grinnemo*, Anna Brunstrom*, David Ros†, Michael Tüxen*, Naeem Khademi⁺, Per Hurtig*

†Simula Research Laboratory, *Karlstad University, *Fachhochschule Münster, ⁺University of Oslo
Introduction

• Deployment of new transport protocols is a difficult task

• How to know if a new protocol is supported along the whole end-to-end path?
  – Try it

• If not supported a fallback mechanism is needed
  – Testing protocols serially can be time consuming

• Happy Eyeballs for transport protocol selection
Happy Eyeball Example: TCP and SCTP

- **Concurrent** initiation of TCP and SCTP
- **Preferred** connection attempt attempt wins
Related Work

• Happy Eyeballs introduced as a way of promoting the use of IPv6 [RFC6555]
• Transport Happy Eyeballs was proposed as a mechanism to run HTTP over SCTP [Wing10]
• Ongoing work in IETF TAPS WG
  – “... explain how to select and engage an appropriate protocol and how to discover which protocols are available for the selected service between a given pair of end points”

Cost of Happy Eyeballs?

• Increased server load
  – CPU load
  – Memory usage

• Increased network traffic

• Goal: Assess impact of happy Eyeballs on server load
Experiment

- Length of each test run: 600 s
- Exponentially distributed http reqs.
  - [100,1000] reqs./s
- Metrics:
  - CPU utilization
  - Kernel memory usage
Basic Test Case

- Always results in a TCP connection
- No caching of previous connection attempts
- Unencrypted connections
CPU Utilization in Basic Test Case

- **Requested object size = 1 KByte**
- **Requested object size = 35 KBytes**

![Graph showing CPU utilization for TCP, SCTP, and HE-TCP with different request rates and object sizes.](image-url)
Memory Usage in Basic Test Case

The chart illustrates the kernel memory usage (MBytes) for different request rates (requests/s) and object sizes in Basic Test Case. The y-axis represents the kernel memory usage, while the x-axis shows the average request rate (requests/s) and the requested object size. The chart compares different protocols: TCP, SCTP, and HE-TCP.

- **Requested object size = 1 KByte**
- **Requested object size = 35 KBytes**

The bars indicate the memory usage with error bars showing the variability. The memory usage generally increases with increasing request rates and object sizes for all protocols.
TLS Test Case

- No caching of previous connection attempts
- TLS-encrypted connections
### CPU Utilization in TLS Test Case

<table>
<thead>
<tr>
<th>Average request rate (requests/s)</th>
<th>TCP</th>
<th>SCTP</th>
<th>HE-TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td><img src="chart.png" alt="CPU Utilization Graph" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35K</td>
<td><img src="chart.png" alt="CPU Utilization Graph" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sharing of CPU load in TLS Test Case

- random\_harvestq system process
- kernel system process
- intr system process
- all lighttpd processes (sys CPU time)
- all lighttpd processes (usr CPU time)

Mean CPU Time (seconds)

<table>
<thead>
<tr>
<th>Average request rate (requests/sec)</th>
<th>no-TLS</th>
<th>TLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>400</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>700</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>1000</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>

neat
Memory Usage in TLS Test Case

![Bar chart showing memory usage in different TLS test cases. The x-axis represents the average request rate (requests/s), and the y-axis represents kernel memory usage (MBytes). The chart includes data for TCP, SCTP, and HE-TCP at average request rates of 100, 400, 700, and 1000 requests/s, with two different rates (1K and 35K) indicated for each statistic.]
Cache Test Case

- Caching of previous connection attempts
- Both unencrypted and TLS-encrypted connections
  - HE-TCP: always results in a TCP connection
  - HE-SCTP: always results in a SCTP connection
  - HE-50%: 50% chance TCP/SCTP connection
Cache Hit Ratio vs. CPU Utilization

Unencrypted

1 KiB

35 KiB

TLS-encrypted

CPU utilization (%) vs. Cache Hit Rate

CPU utilization (%) vs. Cache Hit Rate

neat
Conclusion

• Happy Eyeball is a feasible transport-selection mechanism

• Small increase in CPU utilization
  – Around 10% for 35 KiB web objects
  – Encryption has larger impact on CPU load
  – Caching can reduce load further

• Basically no increase in memory usage
Future Work

• More extensive evaluations with more than two feasible transport solutions

• Transport service library with Happy Eyeballs support
  – https://github.com/NEAT-project/neat

• Evaluation of Happy Eyeball in real-world scenarios with middleboxes etc.