Diagnosing Slow Web Page Access at the Client Side

Tobias Flach

University of Southern California
December 9, 2013
Challenge: Explain why web access is slow sometimes, and develop solutions for the root causes.

**Goals**

Explain why web access is slow sometimes

Inform solutions to fix the root causes
Challenges

Web pages have become increasingly complex

Hard to establish which resource(s) are responsible for bad performance

Some performance issues are short-living

Hard to reproduce the problem
Related Work

**Client Side**
- Inject measurement scripts into page (Fathom)
- Persistent network performance analysis (Netalyzr)

**Server Side**
- Collect data for requested resources
Our Solution: A Tool which ...

- Passively monitors browser behavior as well as network traffic
- Actively probes the network when it detects an anomaly
- Classifies the anomaly and determines root causes based on collected data and feature-cause mappings
Tool Architecture

Collects passive measurements continuously

Conducts active measurements after finding indicators for a performance anomaly
Data Collection

Passive
- TCP packets/connections
- Browser data (e.g. DOM)

Active
- Pings and traceroutes

Only done for active tab when indicators for a performance anomaly are detected.
Data Analysis

1. Trigger active measurements and data analysis through indicators for performance anomalies
2. Annotate recorded packets and connections
3. Cluster traces with common properties
4. Map features onto anomaly types
Data Analysis: An Example

1. Trigger: Rendering delayed by multiple seconds

2. Detect out-of-order packets (and packet loss) as well as indicators for bufferbloat on some connections

3. Low-performance connections are grouped together when clustering by common subpath

4. Inferring network congestion and excessive buffer space on shared subpath
## Sample Mappings

<table>
<thead>
<tr>
<th>Feature(s)</th>
<th>Anomaly type</th>
</tr>
</thead>
<tbody>
<tr>
<td>High RTT and filled receiver window</td>
<td>BDP limitation</td>
</tr>
<tr>
<td>High packet loss</td>
<td>Network congestion</td>
</tr>
<tr>
<td>Tail packet loss</td>
<td>TCP limitation</td>
</tr>
<tr>
<td>High packet reordering</td>
<td>Packet-based load balancing</td>
</tr>
<tr>
<td>HTTP request without response</td>
<td>Backend error / reachability issue</td>
</tr>
<tr>
<td>Consecutive RTOs (HTTP GET)</td>
<td>Backend error / reachability issue</td>
</tr>
<tr>
<td>Low throughput on all connections</td>
<td>Access bandwidth limitation</td>
</tr>
<tr>
<td>Low throughput, and high throughput on other connections</td>
<td>Unfair bandwidth allocation</td>
</tr>
</tbody>
</table>
Conclusion

Proposing a tool to detect transient performance anomalies

Currently working on an implementation for the Chrome browser

Supplements existing frameworks which focus on detecting persistent issues