Improving the performance of SPDY for mobile devices

Junaid Khalid[1] Sharad Agarwal[2] Aditya Akella[1] Jitendra Padhye[2]

University of Wisconsin - Madison[1], Microsoft Research[2]

SPDY [2] is an application layer protocol developed for reducing web latency. It is estimated that about 3.6% of all websites use SPDY today [3], but that statistics hides how many web transactions use SPDY. It has rapidly gained adoption across Google, Facebook, Twitter, Yahoo, Reddit, and WordPress. While only a small fraction of all websites use SPDY today, SPDY-capable proxies are being deployed to provide the benefits of SPDY over the "last mile" while traffic between proxies and the origin webserver remains HTTP1.x for the time-being.

SPDY improves web performance through various mechanisms including multiplexing multiple HTTP transfers over a single connection, header compression, HTTP request prioritization, and server push; which sends down content before it is requested. Push can help when the client is underpowered and is slowly processing Javascript or app code, or is waiting for the user to interact with the displayed content.

The relative benefits of multiplexing, header compression and request prioritization have been measured and analyzed by several blogs and research papers [6, 1, 4]. However, the proactive pushing of HTTP content to the client has received relatively less scrutiny. The promise of push in SPDY is that through machine learning on prior access patterns or otherwise, the server can intelligently push down content to the client before it realizes it is needed, thereby significantly reducing user-perceived latency.

Push is problematic for mobile devices because it can waste both battery and bandwidth. This happens either because the server blindly sends down the content which is already in the client's cache or it ends up sending content to the client which is never consumed by the user.

A reasonable solution may be to simply turn off push when the mobile client is on a limited data plan or has low battery. This binary decision mitigates the worst case scenario, while retaining the powerful latency advantage of push only in the ideal scenario. We can do better.

We propose two basic mechanisms that the HTTP 2.0 standard should adopt to *dynamically* adjust the overall performance (speed, battery consumption, data consumption) of mobile clients:

- **Cache hints** – we propose a lightweight mechanism for the client to indicate what cached content it has. The server can then modulate its intended set of push objects. When the client initiates the connection, it sends the server an array of bloom filters with the first HTTP request. The Bloom filter at index n of that array will represent the objects which expire in less than 2^n seconds.

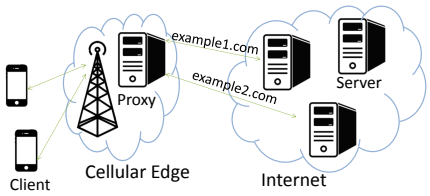

Figure 1: Architecture Overview

- **Half-push and half-pull** – recognizing the widespread deployment of web proxies that already have or soon will be upgraded to SPDY, we propose half-push and half-pull to explore a new dimension in the proxy design spectrum. As shown in Figure 1, we envision proxies that are deployed at the edges of cellular networks and that provide capabilities to both servers and clients to cache content. The client can explicitly request a "half-pull" that will cause the content to be temporarily held at the proxy and not traverse the "last mile" until it is necessary. Similarly, the server can "half-push" content to a proxy that is closer to the client without incurring a battery and data cost.

Our preliminary evaluation based on the data from prior work [5] of 6 popular apps shows that half-push and half-pull reduce the bytes transferred by up to 4x for some applications as compared to regular push or prefetching. Half-pull or half-push to a proxy incurs additional delay when the client issues the regular pull – this delay is roughly 10ms assuming an LTE connection with 12.7 Mbps throughput and 3ms RTT to such a proxy co-located at the LTE tower.

We achieve this data reduction with a minimal proxy storage overhead. The peak memory used at the proxy is less than 500 KB for at least 95% of user session for each app.

1. REFERENCES

[1] Not as SPDY as you thought. `http://www.guypo.com/technical/not-as-spdy-as-you-thought/`.

[2] SPDY IETF Draft. `http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00`.

[3] Usage of SPDY for websites. `http://w3techs.com/technologies/details/ce-spdy/all/all`.

[4] J. Padhye and H. F. Nielsen. A comparison of spdy and http performance. Technical Report MSR-TR-2012-102.

[5] L. Ravindranath, S. Agarwal, J. Padhye, and C. Riederer. Procrastinator: Pacing mobile apps' usage of the network. In *ACM MobiSys*, 2014.

[6] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. How speedy is spdy? In *USENIX NSDI*, 2014.