Initial Plan

Details:

Title: "Web Protocol: A modern replacement for HTTP"
Author: Christopher Jamie Hall
Supervisor: Prof. Omer F. Rana
Moderator: Prof. Ralph Martin
Module: CM0343
Credits: 40

Description:

Initially developed in 1991, HTTP was designed to be highly extensible and to be used by command-line interfaces (CLI). In the following two decades, use of the web has exploded and the way it's used has changed dramatically. Instead of writing requests by hand or by script, content is navigated by complex, feature-rich web browsers. Rather than single pages with the occasional image, modern web pages are incredibly intricate, containing images, videos, stylesheets, scripts, icons, and external ads. HTTP is being used for tasks never considered at its time of creation.

With bandwidths soaring to over 100 Mbps, the main bottleneck to internet responsiveness is swiftly becoming the latency; the time it takes for the packets to travel from the client to the server and back. Our ability to decrease this round-trip-time (RTT) will not continue for much longer and as we approach the time it takes the packets to travel that distance, the only way to improve performance is to reduce the number of trips made. Google has created the SPDY protocol to attempt to reduce the number of trips made by HTTP by multiplexing multiple streams onto a single TCP stream.

However, HTTP also suffers from the inefficiency of a text-based protocol, which takes substantially more processing to parse, since sanitisation for capitalisation and spacing may need to take place, and then a text parsing is used to process the packets. By comparison, binary protocols like TCP and IP can be parsed and constructed incredibly quickly, with highly-optimised kernel code.

Web Protocol (WP) uses various methods to provide greater efficiency and features for the modern web like request suggestions and content pushing. A binary protocol in its entirety, WP is far faster to read and write than HTTP, and has far smaller headers. By multiplexing its streams onto a single TCP session, WP uses far fewer transmissions than HTTP; improving network efficiency and performance.

To measure the difference in efficiency objectively, the Chromium open source web browser and Apache httpd web server will be supplemented to support a reference implementation of WP; providing a functional testing environment. The Webkit Web Inspector network tools built into Chromium can then be used to compare transmission, creation, and parsing times between WP and HTTP from an end-user's perspective; giving realistic, unbiased data for comparison.

At the conclusion of the project, the protocol specification and reference implementations will be released under a BSD open source license, since proprietary licensing would tie an unmanageable weight around the protocol's feet.

Aims and Objectives

- Write the protocol specification document for version 1.0 of the Web Protocol.
- Create a reference implementation of the protocol in a customised version of the Chromium web browser and Apache httpd web server to represent both client and server in real-world applications.
- Perform performance comparison experiments on WP and HTTP.
- Analyse the resulting data to provide an indication of the performance gains (and losses) given by WP.
- Use experience gained in the experimentation to adjust and improve the protocol, amending the reference implementations and specification document accordingly.

For the interim report, a draft specification will be written and a code to allow client and server use of a basic form of WP will be written for basic functionality experimentation. This is not expected to implement the full protocol, but should provide sufficient function to allow initial performance tests to take place. This will be drawn from thorough research into the HTTP protocol and its use, ensuring that WP can perform any task for which HTTP would be suitable.

The final report shall consist of an analysis and discussion of the results gained from thorough experimentation on the final implementation of the protocol, including all features and full functionality. Initial test data will be used to improve the implementation and add/remove features as necessary.

Alongside the final report, the custom versions of Chromium and httpd (wpd) will be made available publicly as open source software, along with the specification document and a wiki explaining the protocol’s intended use and design principles.

Time Plan

NB. During the summer, I did a reasonable amount of initial research into HTTP and two alternative improvements, namely SPDY and WebSocket, and wrote a first draft specification for WP.

Autumn semester:
Week 1: Write basic HTTP server in C for initial testing, generate example HTTP session data.
Week 2: Write basic WP server in C.
Week 3: Write basic WP client in C, generate example WP session data.
Week 4: Perform initial tests comparing WP and HTTP on a basic example website.
Week 5: Write second draft specification, alter and improve WP server and client.
Week 6: Research and perform code review on Chromium for adding a new protocol.
Week 7: Research and perform code review on Apache httpd for adding a new protocol.
Week 8: Begin Chromium WP implementation.
Week 9: Continue WP for Chromium.
Week 10: Submit interim report

Christmas holiday:
Continue WP for Chromium.

Spring semester:
Week 1: Continue WP for Chromium.
Week 2: Begin Apache httpd WP implementation.
Week 3: Continue WP for Apache httpd.
Week 4: Continue WP for Apache httpd.
Week 5: Continue WP for Apache httpd.
Week 6: Perform final efficiency experimentation.
Week 7: Analyse results.
Week 8: Write final report and protocol specification.
Week 9: Submit final report, protocol website with applications and specification.