SBU ID: 114834757 Name: Jai Anish Mehta SUBJECT: FCN

(PAPER: "How Speedy is SPDY?")

SUMMARY 2

The performance of the SPDY protocol proposed by Google has been frequently measured relative to that of HTTP/1.1. But the results of these comparisons have not been very clear, as the performance is influenced by a wide array of external factors. It is observed that SPDY is significantly influenced by the computations in the browser and the dependencies involved in the Page Load Process. If it was to ignore the former factors, SPDY could have a significant advantage over using HTTP. The external factors, variability in Page Load Time (PLT) and dependencies between network activities and browser computations make it challenging to study the performance of SPDY. This shows that it's important to consider the entire parameter space to find the supporting parameters when SPDY benefits and also when it hurts.

In studying the protocols, the authors outline the key differences and challenges between SPDY and HTTP, which aid the performance measurements. Differences like the number of TCP connections, server push and prioritization, and overhead due to the headers. The comprehensive sweep of parameter space helps evaluate the factors affecting SPDY performance and also gives the authors an idea of how to tweak TCP connection to gain the most out of SPDY. Besides being the first ones to isolate the effect of dependencies, one of the major contributions of the authors is the development of the tool Epload, which helps in controlling the inherent variability in computation by making comprehensive controlled page load requests. Epload helps to evaluate the ways to bring down the negative impacts of dependencies and computation. For studying these factors an elaborate measurement was performed using real and synthetic (predefined object sizes and their frequency) web pages, using a client-server setup running HTTP and SPDY requests. To decode the frames of SPDY in TCP payload, authors use the SPDY 3 protocol without SSL and controlling web object sizes.

Developing their own SPDY and HTTTP clients, injecting random packet loss and removing the cross traffic, the authors performed measurements on the web pages. To demonstrate the conditions under which the SPDY helps or hurts, the authors, make use of a predictive model built on the basis of a decision tree, which generates a likelihood giving information about which configuration gives what results. It is observed that branches with a likelihood score over 0.75 are marked as SPDY. Experimenting on synthetic pages, authors inferred that many objects under low packet loss or fewer objects with a good network and wider TCP icwnd are more likely to benefit from the SPDY and other factors like RTT, bandwidth etcetera, don't affect that significantly. For the real pages, SPDY shows a speedup due to a lower retransmission rate and half of the pages have a high frequency of small objects. As SPDY uses a single TCP connection, it can have a large negative impact due to the aggressive reduction in congestion window size upon encountering congestion. Hence the authors contribute by proposing a TCP+ protocol which has concurrent connections within a single connection, thereby reducing the detrimental effect of the congestion on SPDY. The authors also provide a tool to measure the importance of objects using an object-based graph instead of the activity-based dependency graph.

Using the decision tree analysis, it will be fit to conclude that SPDY would help the most with low object sizes in combination with low loss rate and would hurt the most with high packet size along with high packet loss, using a single pipe. To support this authors state that there are high packet losses in multiple TCP connections competing with each other and hence a single connection not only reduces the packet losses but also helps in reducing the pipe idle time. SPDY helps more when a higher number of outstanding objects exist and the dependencies and computations can completely negate the benefit of using SPDY, increasing the PLT, whereas the dependencies help HTTP in making the traffic less bursty. If the dependencies are broken and computations are made faster, then RTT and bandwidth will become influential factors in SPDY performance. For this prioritization and server push can be used, but there has not been enough resource on how to use them. It would be important to identify the importance of objects to employ them. The authors highlight that server push will help over prioritization, but it helps the most under high RTT conditions.

Something that I found really amazing about the research is the tool Epload built by the researchers. The comprehensive nature of the tool that replays a chain of finely controlled page load requests is truly appealing. Not only does the tool just send requests but also imitates the computations and injects random delay to mimic a real network making the work of the researchers noteworthy. Coming to SPDY, I think it is going to be the future from what I feel about the research shown in the paper. Google really has an edge in testing this technology as it has its own servers and clients using the chrome browser. With this kind of setup, if Google is investing in research on this, I think they have some strong lead they might be working towards.

In spite of considering these major factors, there could be more complex loss patterns in real networks affecting the performance of SPDY, which would require even more deeper evaluation, seems to be intimidating, but equally exciting to follow the research.