1 Problem

This paper **compares SPDY**(single TCP connection) **and HTTP's performances**/PLT(Page Load Time) by considering various factors such as RTT, bandwidth, loss rate, TCP initial window, no. of objects on a page and object sizes. Author highlighted various conditions where SPDY outperforms HTTP and vice versa using a **Decision tree**. Author discussed Prioritization and server push policy based on **dependencies** and computation of real page loading. For experimentation purpose, **Epload** was used to control the variability of PLT.

2 Contributions

Author conducted various experiments - comparing SPDY and HTTP as a simple transport protocol which transfers objects from synthetic and real pages. Author came up with decision tree after analyzing various situations where SPDY outperforms HTTP. Epload tool was used to identify the effect of page load dependencies and computation, by allowing to simplify computation while scheduling network requests. Author restructured TCP, and named it **TCP+**, which makes it efficient. Modifications include mimicking the behaviour of concurrent connections using single connection, receiver buffer size to n(no. of parallel connections), rate at which congestion window backs off to slow start phase. TCP+ combined with server push policy by one dependency level enhances SPDY under high RTTs.

3 Conclusions & Support

HTTP opens multiple connections which results in network congestion. SPDY is designed by addressing the drawbacks in HTTP/1.1 with single TCP connection(reducing SSL overhead), prioritization of requests(scheduled transfer of objects), server push policy(caching), Header compression(avoids duplicate copies). Analyzing SPDY performance was challenging due to it's varied performance under different network environment, web page characteristics, network parameters, TCP settings, and variance in PLT, Page dependencies. To understand which factors contribute to performance improvement in SPDY, experiments were conducted in a controlled network environment by downloading home pages of top sites to own server. Later, experimented with synthetic pages considering broad range of parameter settings, Also injected random packet losses. Based on the decision tree, SPDY hurts on high packet loss. Object size, loss rate are the most important factors. SPDY helps on small objects, benefits on having one TCP, degrades under high loss. It showed a 2x speedup by eliminating retransmission while experimenting with real pages. It helps under low loss and good network environments. While prioritizing, Convert activity-based dependency graph to object based graph by eliminating computation while preserving dependencies. Then, calculate the longest path from each object and use this information to prioritize and push. SPDY allows 8 priority levels for clients to request objects. implicit prioritization depends on browser policies, independent of web pages. Since dependencies limit the impact of SPDY, prioritization cannot break dependencies - which means it is unlikely to improve SPDY PLT. Server push policy helps save round trips, but it is non-trivial because of wastage in bandwidth. Server push has the ability to break the dependencies resulting in performance gains of SPDY.

4 Likes

Detailed analysis with the help of experiments in controlled network environment while comparing HTTP and SPDY by considering external factors. server push feature in SPDY. TCP+ in SPDY.

5 Dislikes / Disagreements

TCP+ together with server push by one dependency level improves SPDY performance under high RTTs. However, It is influenced by external factors making it diminish under low RTTs