Hindawi
Journal of Computer Networks and Communications
Volume 2018, Article ID 4520312, 9 pages
https://doi.org/10.1155/2018/4520312

Research Article

Utilizing Connection Usage Characteristics for Faster Web Data Transport

Gautam Raj Moktan, Nutti Varis, and Jukka Manner

Comnet, Aalto University, Helsinki, Finland

Correspondence should be addressed to Gautam Raj Moktan; gautam.moktan@aalto.fi

Received 25 October 2017; Accepted 8 April 2018; Published 6 June 2018

Academic Editor: Peter Mueller

The quest for faster data transport to improve web user experience is ongoing and attempts are conducted from various fronts to realize it. On top of improving user experience, the implications of improving web data transport are also on the energy efficiency of wireless devices as well as user retention rates of service providers. HTTP/1.x allow the opening of multiple TCP connections per server and then using those connections for fetching multiple web objects through the use of HTTP pipelining. With the advent of HTTP/2.0, multiplexing is done inside a single connection to fetch multiple objects. In this paper, we analyze the TCP connections between the browser and the servers and examine their characteristics. We describe how an enhanced TCP variant can take advantage of data transport connection patterns. We show the benefits that enhanced TCP system can bring with the understanding of connection usage patterns. We find that such transport protocol can have effect in the page idle times as well as the connection concurrency during web page transfer. The results show significant improvement of page load times for both encryption heavy and unencrypted pages. We discuss the effect of the transport protocol on object transfer, connection duration, idle times during the page load, connections, and concurrency of flows that cumulate into page load times.

1. Introduction

Data transfer mechanism in the World Wide Web has not evolved at the same pace as the services it contains. Information in various formats is generated in web servers, and a browser fetches them through the network and renders them for the users' viewing. On the other hand, the diversity in the types and amounts of such objects has grown significantly, and many mechanisms have been developed in order to transfer and render them faster and more efficiently.

Hypertext Transfer Protocol (HTTP) [1] enables the client-server model through which information is transferred in the web between browsers and servers. The commonly used transport protocol by HTTP is the Transmission Control Protocol (TCP). Thus, a browser and a server first establish a TCP connection over which the HTTP messages are communicated in order to fetch the various objects required to display a web page.

Setting up a TCP connection entails a handshake mechanism that requires 3 trips of message transfer between the browser and the server. This overhead is becoming inefficient considering that a browser generates about 100 requests, on average, to fetch a web page [2]. Thus, to improve efficiency, modern browsers allow reusing the same TCP connection for fetching multiple web objects through the use of HTTP pipelining [3]. HTTP/1.0 uses keep-alive headers for it, and HTTP/1.1 considers connection persistence unless declared otherwise. With the advent of HTTP/2.0, multiplexing is done inside a single connection to a server to fetch multiple objects.

Since the users' machines process information faster than the rate of fetching files from the network, parallel TCP connections enable faster web page loading. HTTP/1.x allows such mechanism, and all major browsers nowadays establish up to 6–8 connections per server. HTTP/2.0 has built-in multiplexing to fetch multiple objects in parallel.

Yet, in the ongoing quest for faster web data transport to improve user experience, attempts are made on various fronts to realize it. Compression, Caching [4], TCP tuning [5], and so on are implemented at the servers as well as middle-boxes. Domain sharding (overcoming the browser's limit of maximum simultaneous connections per domain by downloading resources from multiple domains), DNS prefetching [6], content inlining [7], prioritizing, and so on are some other techniques.

The focus of this paper is to investigate the characteristics of data transport connections in the web and analyze the effect of using an enhanced transport protocol to capitalize on the connection usage characteristics. We study the amount and types of files fetched through established connections. We also analyze the connection usage and reusage patterns for web content download in the web using the home pages of the Alexa top 100 sites [8]. We validate our sample against the whole list maintained at *httparchive* [2].

We implement an enhanced TCP, FLD_TCP, which pushes short flows (flow length smaller than 2MB) faster through the network as compared to traditional TCPs. In order to measure the benefits on the downloads of Alexa top sites to a user's browser, we implement the transport protocol in a proxy server (to emulate the content servers) and measure the performance in a Chrome browser. We show the benefits such a system can bring as compared to the standard TCP with the understanding of connection-level patterns. We find that this novel transport protocol can have positive effect in the connection idle times as well as the object fetching times. The results show significant improvement of page load times through the use of FLD_TCP in both HTTP/1.x and HTTP/2.0 cases. We discuss the different application scenarios and the implications of deploying such a system for accelerating web transfer.

2. Background and Related Works

As a part of an ongoing effort to improve the page load speed on the web, many methods have been proposed. Also several studies are done on the interactions during a page load among the browser, the servers, proxy middle-boxes, and the network. A web page can be made to load faster either by bringing the content closer to the browser or by reducing the number of fetches required to load the page. Moreover, it can also be achieved by optimizing the underlying transport mechanism.

Content caching [4] has been used primarily to reduce the number of requests for the web pages. Depending upon its implementation at different nodes of the web page load interaction, it brings the content closer to the browser. Use of cloudlets [9] to bring the content closer to the users is also gaining momentum. This has been driven by the advent of 5G, Software Defined Networking (SDN), and advancing Content Distribution Networks (CDN) technologies. Webpage bundling has also been purposed to offload much of the computation task into the cloud. WebPro [10], Cumulus (MahiMahi) [11] and PARCEL [12] are some examples where page content is fetched and bundled in the cloud and the browser only need to render the received bundle.

Techniques such as content inlining [7] and CSS spriting [13] are used in the design of today's web pages. These techniques reduce the number of object fetches the browser has to do for loading the web pages. Compression is applied to objects to reduce the number of bytes through the network. Google Flywheel [14] and Opera Turbo [15] are examples of proxies that compress content and apply other latency reducing techniques to improve the web page load performance. DNS prefetching [16] allows eliminating the DNS resolution time for previously accessed domains.

Optimizing the order in which the objects are loaded in the browser also brings benefits. Polaris [17] loads web pages according to the dependency tracking done by the dependency graph generator, SCOUT, in the offline mode. Klotski [18] also capitalizes on the dependency graph to evaluate the optimal prioritization of resources of the page. WebGage [19] prioritizes the loading of webpage sections that catch user's attention more. Prophesy [20] uses SCOUT to recompute the JavaScript heap and DOM tree for web pages so that browsers can render the page faster.

Optimizing the transport mechanism itself also contributes to faster loading of web page. Enabling multiple HTTP requests to utilize the same transport connection has reduced the delay caused by TCP connection establishment overhead. Also, with HTTP/2, multiplexing of requests inside a single TCP connection has allowed better utilization of the connection. HTTP/2 also allows Server Push mechanism where the server can preemptively send objects that are needed to fulfill the page load to the browser in response to the requests, thus saving network transfer time. The servers can also apply TCP tuning techniques [5] to better adapt to various network conditions. TCP WISE [21] demonstrates that with relaxing the constant initial window size, HTTP latency can be significantly improved.

Analysis of interactions among the browser, the servers, proxy middle-boxes, and the network during the page load has been done from different perspectives. Mahimahi [11] conducted the performance comparison between different application level protocols in different emulated network conditions. Gangadhara Rao et al. [22] provide the analysis of web server load by TCP connection establishment phase. Qian et al. [23] provide interesting insights on the interactions of caching, content type, timing characteristics, and connection management. Konorski and Lis [24] analyze through simulation the effect of aggressive TCP configuration in networks. Similar work using game theoretic tools are done by Zhang et al. [25].

To supplement the research in this field, our work quantitatively provides the real measurement analysis of the interactions of TCP connections during web download. We also analyze the interaction when the transport protocol of the connections is enhanced to be more aggressive for short flows which most web resources create. We begin by describing the functionality of the enhanced transport protocol, FLD_TCP.

2.1. FLD_TCP: An Enhanced TCP Variant. Flow-Length Dependent TCP (FLD_TCP) is an experimental modification to the Transmission Control Protocol that prioritizes the short flows to finish faster as compared to long flows. It tries to attain a higher share of the network bandwidth than other TCP flows as long as the flows are short and becomes

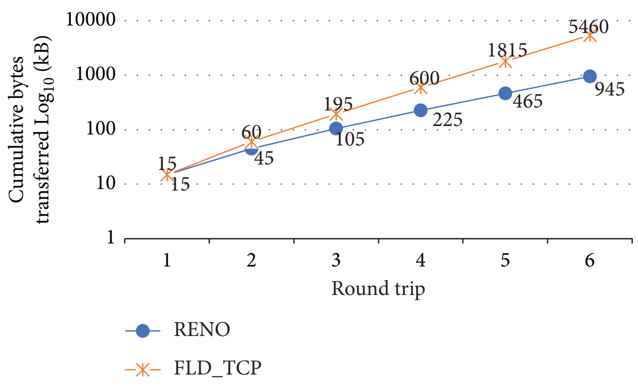


Figure 1: Cumulative bytes transferred per round trip.

TCP-friendly once the amount of transmitted data exceed a threshold value.

With the objective of enabling short flows to finish faster, it also starts with a higher congestion window of 10 segments as done in the newer Linux TCPs. In the Slow-start phase, FLD_TCP is more aggressive than traditional TCP. While traditional TCP doubles its congestion window every round trip in the Slow-start phase, FLD_TCP triples its congestion window.

Figure 1 demonstrates how the modification of congestion window growth factor in the Slow-start phase saves round trips required to finish flows. We compare the cumulative amount of bytes transferred over each round trip Time (RTT) between traditional (RENO) Slow-start and FLD_TCP's Slow-start. We assume flows in both starts with initial congestion window of 10 segments and each segment holds 1500 bytes for simplicity, and they do not hit the bottleneck so that congestion avoidance mode does not kick in. We see that after the 1st RTT, both transfer 15 kB, and by the 2nd RTT, FLD_TCP transfers 60 kB while RENO transfers just 45 kB. This difference increases as RTT progresses. Note that the Y-axis is in logarithmic scale, so the difference between the two mechanisms is quite significant. If we consider a flow of size 500 kB, we see that FLD_TCP would finish the flow in 4 RTT while RENO's Slow-start would require 6 RTT.

Also in the congestion avoidance phase, as long as the amount of transmitted data is within the specified threshold value, FLD_TCP functions in Relentless TCP [26] mode. Instead of multiplicative decrease upon packet loss, this TCP reduces the congestion window by the amount of lost segments only. An advantage to this approach is that this abides by the Van Jacobsons packet conservation principle while being aggressive at the same time.

And after the TCP flow exceeds the threshold value, FLD_TCP functions in traditional RENO behavior thus acting in a TCP-friendly manner. The threshold value is selected to be 2 MB which is enough to accommodate small objects from most connections as shown in Figure 3. Thus, for flows less than 2 MB, FLD_TCP provides an aggressive transport enabling them to gain a bigger share of network bandwidth in the presence of cross traffic at the bottlenecks allowing them to finish faster.

Thus, this enhanced TCP with its aggressive components in the Slow-start, as well as the congestion avoidance phase allows short flows to gain a bigger share of network bandwidth and hence finish the flows faster. The workings of the protocol and its performance evaluation in bulk file transfers are described in papers [27, 28]. In the following sections, we describe the implications of using such transport protocol for web data transport.

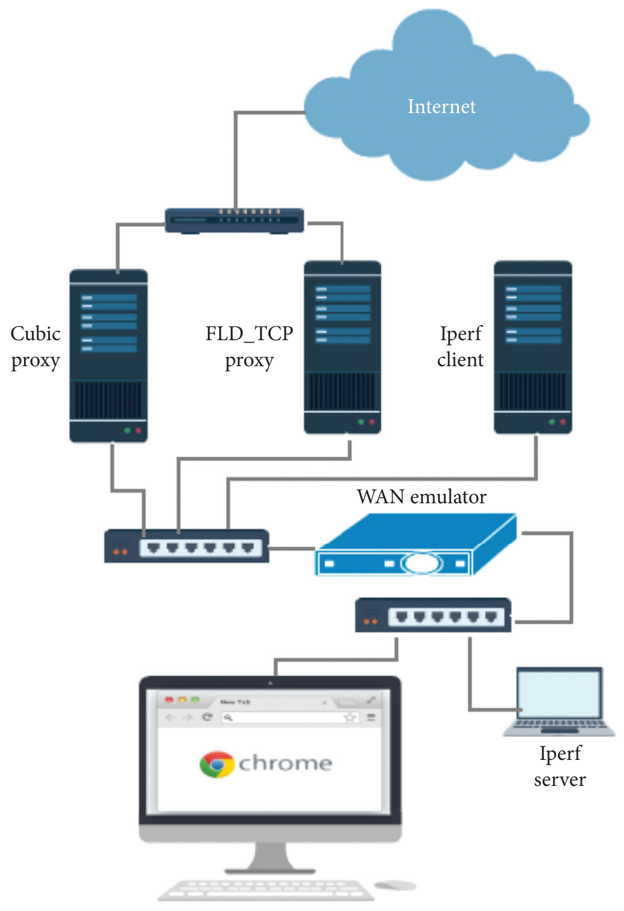


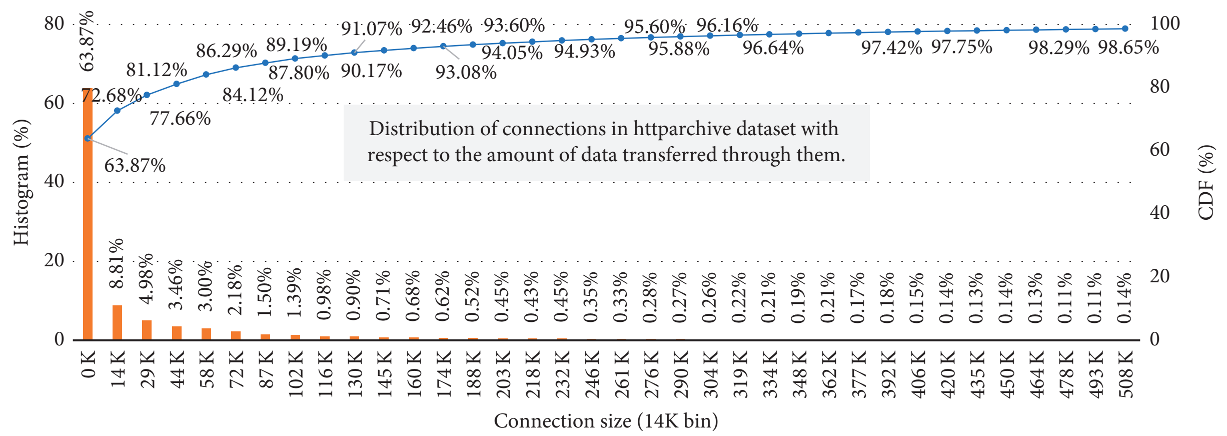
Figure 2: Evaluation setup.

3. Measurement and Analysis Setup

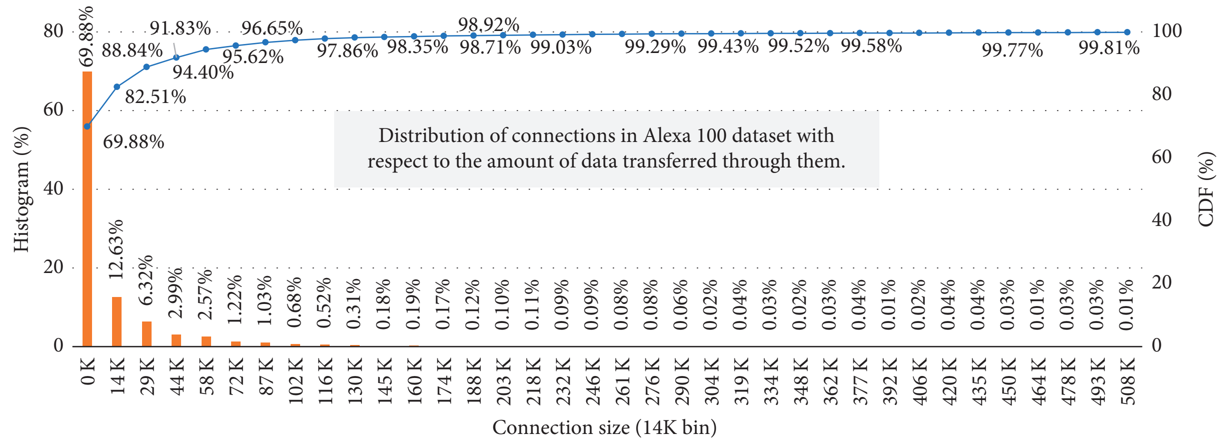
The measurements are done on Google Chrome browser. Through the Chrome remote debugging protocol, various statistics are collected during the loading of the target web pages including connection characteristics. The setup for the experiments run in this paper is laid out in Figure 2.

Since we compare the performance of an enhanced TCP (the enhancement is only sender side modification), against the commonly used Cubic TCP, squid proxies are placed on the path between browser and servers to emulate the servers. This implementation enables us to examine the effect of using a different transport protocol for sending web data through the network. Also, we use a WAN emulator in the path and create background traffic as required using IPerf [29] to study the performance under realistic network conditions.

The WAN network emulator is set up to control the bandwidth and delay parameters of the link between the



(a)



(b)

Figure 3: Comparison of connection sizes distributions in homepage of Alexa 100 sites with the larger dataset of http://httparchive.org.

proxy and the browser. We use the LTE and the 3G network settings for our measurements. The maximum bandwidth is set at 100 Mbps for LTE and 21 Mbps for 3G. To obtain realistic values of delay and bandwidth for the wireless links, we analyzed the data obtained from the measurement platform *Netradar* [30]. We observed that for LTE, the effective average bandwidth is 25 Mbps and delay is 13 ms. For 3G, it was 7 Mbps and 25 ms, respectively. Thus, we introduce background traffic flows using IPerf to bring down the effective bandwith so that it is similar to the real world values. For the test measurement runs, each web page is downloaded 11 times, and the aggregate measures are analyzed to mitigate the effect of network variance.

3.1. Connection Characteristics on the World Wide Web. In order to characterize the TCP connections in web data transport, we conducted measurements with 100 sites from the Alexa top sites list. The home pages of the sites were loaded on the browser, and the statistics were collected.

To validate our sample dataset, we contrasted the distribution of the volume of data transferred through TCP connections across the web pages in our list against all sites maintained at *httparchive.org*. The *httparchive* dataset was obtained from Google BigQuery. Figure 3 shows similar distributions for the two datasets. We see that the data transferred through each connection have a long tail distribution. Almost 85% of connections transport less than 50 kB of data and 99% of connections transport less than 500 kB of data.

TCP connections are increasingly becoming encrypted on the web, and the proxies are typically unable to cache the web objects served from such encrypted TCP connections. For our measurements where we emulate the end server with a proxy, we want to know how much of the web page data are being served from proxy cache and how much are encrypted or served from elsewhere.

Since we install the two different TCPs on the squid proxies to compare them, we need to be aware of the effect of sending rates of the servers into our measurements. The objects that are cached at the proxies have their sending rate fully controlled by the transport protocol of the proxy. But for objects that cannot be cached, the sending rate of the actual content servers and the link condition between the proxy and the content servers can also have effect, although minimal. Figure 4 shows our observation on how many data

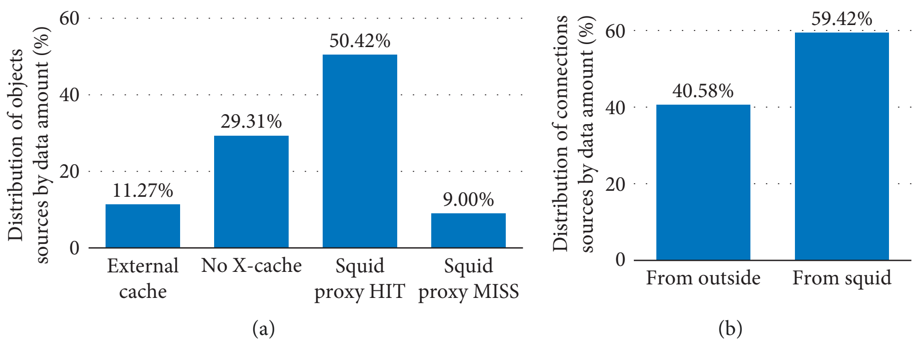


Figure 4: Objects and connections distribution in different cache locations.

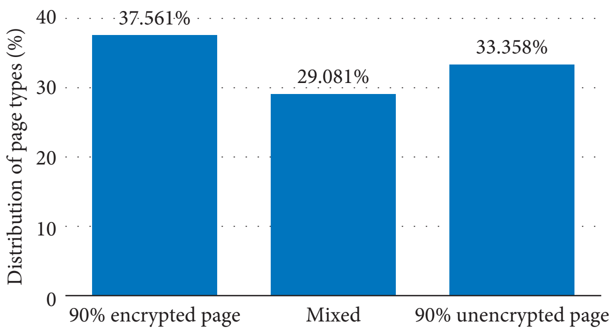


Figure 5: Page types by encryption.

objects are served from different caches as well as how many connections terminate at the proxy.

We see that 59.42% of the web objects are served from TCP connections terminating at the squid proxy (where 50.42% return with cache HIT while 9% object return with cache miss). And 40.58% of objects are served from TCP connections terminating elsewhere (11.27% have different X-Cache header value (which indicates if the HTTP response was served from the proxy or not) from our squid proxy and 29.31% do not have X-Cache header).

We also classified the page types based on whether the connections are encrypted or not. As illustrated in Figure 5, we observed that 37.561% of web pages have more than 90% of the TCP connections encrypted while 33.358% of web pages had more than 90% unencrypted connections serving the pages. 29.081% pages however were served by both encrypted and unencrypted TCP connections. We later compare our analysis on these page types.

Based on these information, we now proceed to measure the effect of the enhanced transport protocol to the page load time of the web pages in our sample list.

4. Results and Analysis

In this section, we present the results of using the enhanced transport protocol FLD_TCP as compared to Cubic TCP for downloading web pages.

Page load duration is obtained from the Chrome browser's *loadEventFired* event. There are idle times during the page load, during which there is no ongoing data transfer from the network to the browser. Thus, we calculate the page net transfer time as the difference between the page load duration and the idle duration during the page load.

Figure 6 shows the effect on page net transfer time of the three different page types on different network characteristics upon using the two different transport protocols, Cubic and FLD_TCP. We see that FLD_TCP reduces the page net transfer time for all pages types (encrypted, nonencrypted, and mixed). The reduction in page load time is more in 3G network setup than LTE. For the rest of the analysis, we will look into the LTE network setup only for the sake of conciseness.

Now we look deeper into the page load process and analyze the distribution of idle time (mentioned above) during page load. The idle time constitutes of the duration where the page is not receiving any objects since we are concerned about network-level downstream activity. Figure 7 shows the CDF and histogram of the idle time duration during the page load indexed against the total page load duration. In general, there is significant idle time during a page load with median number of pages having around 50% of page load time as idle from network data reception perspective. Comparing the FLD_TCP and Cubic graph in the figure, we observe that FLD_TCP increases the idle time proportion in receiving objects for the page. Since FLD_TCP finished object transfers faster than Cubic, an increase in the proportion of idle time during page load occurs for FLD_TCP.

Next, we look into the concurrency of data transfer between the browser and the servers during a page load. We

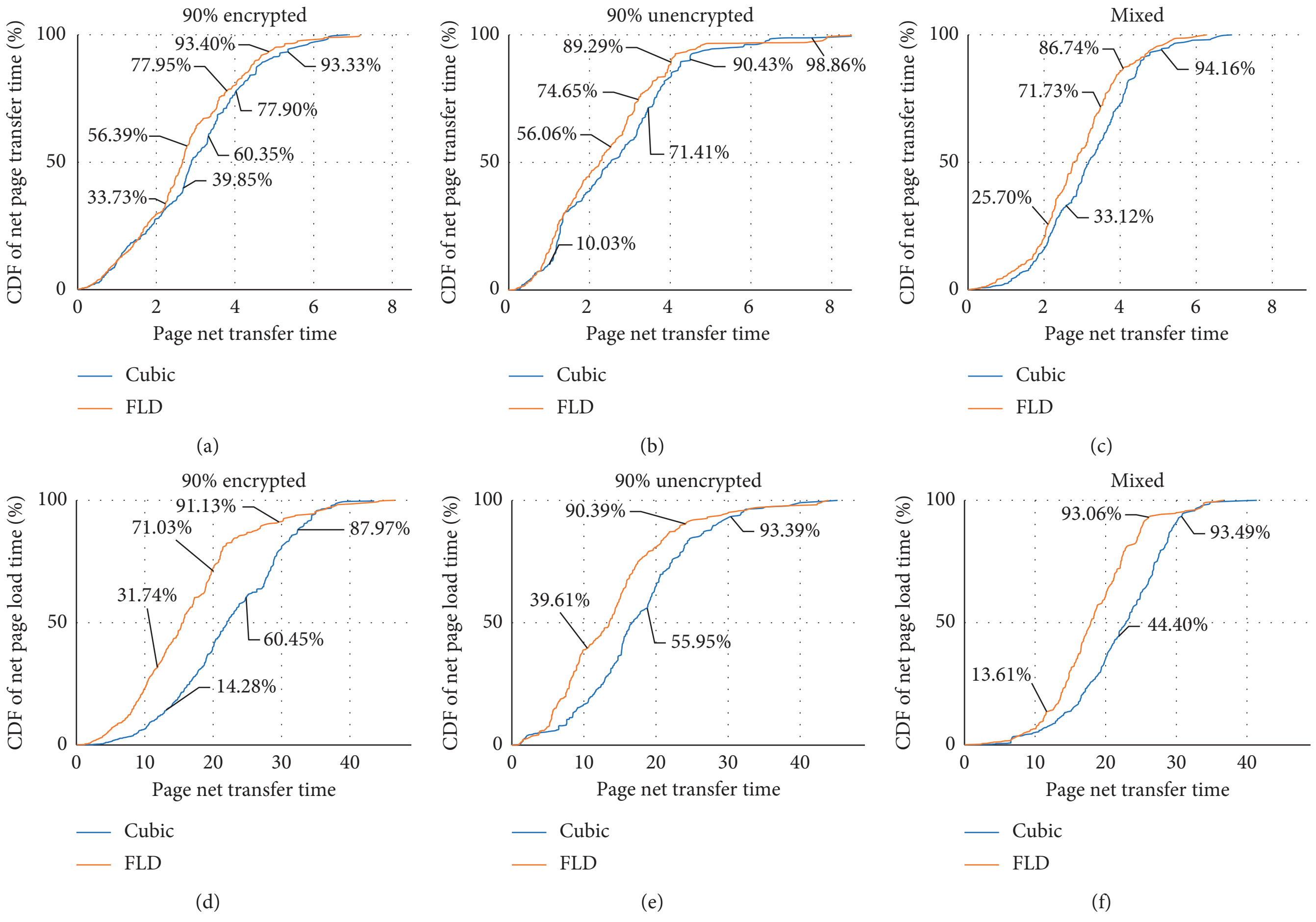


Figure 6: Page net transfer time (LTE upper figure and 3G lower figure) (seconds) obtained for different page types for Cubic and FLD_TCP.

create a page level effective concurrency metric for each page load. By tracking the number of active connections (objects being received through them) and their cumulative duration, we obtain a weighted value for a page that denotes the effective number of connections seen throughout the page load duration. A concurrency level of 1.5 would imply that without any idle time, the page could be served the same amount of data through 1.5 connections. Figure 8 shows that effectively most pages seem to get a concurrency level less than 2. And it also shows that FLD_TCP decreases the effective concurrency of the page, implying that it would require fewer TCP connections to serve a page, leading to improved connection efficiency.

Next, we conduct a connection-level analysis of the two transport protocols. As established earlier, connections are either encrypted or unencrypted. We measure the effects in the duration of both connection types upon using the two protocols. From Figure 9, we observe that FLD_TCP reduces connection duration for both encrypted and unencrypted connections. Since HTTP/2 protocol is gaining momentum, we also analyzed the effect of FLD_TCP on the HTTP/2 connections. Figure 10 shows that the connection durations for HTTP/2 connections are also reduced by FLD_TCP.

For the objects-level analysis, Figure 11 shows the amount of data transferred in each round trip time. The one-way delay for 13 ms was used in our LTE setup that makes the RTT at least 26 ms. Thus, the figure shows the maximum size of objects transferred in each RTT bin. In this analysis, we only consider the first object in the connection that allows us to see the window increase function in action.

We see that in the first RTT of the connection, the maximum data size is 14 KB for both FLD_TCP and Cubic in accordance to the 10 segments initial window size. In the second RTT, Cubic's window size doubles while FLD_TCP's window size triples which allows bigger-sized web objects transfer to also finish within the second RTT. Similarly, in each of the succeeding RTTs, FLD_TCP is able to finish the transfer of larger web objects than Cubic.

Thus, the effects of using FLD_TCP at the sender side of TCP connections are viewed at three levels of granularity. On the objects level, RTTs are saved, that is, web objects are fetched faster. On connection level, the connection durations of the TCP connections become shorter for both encrypted (including HTTP/2) and unencrypted connections. These accumulate at the page level, making the web pages load faster thus improving user experience on the web.

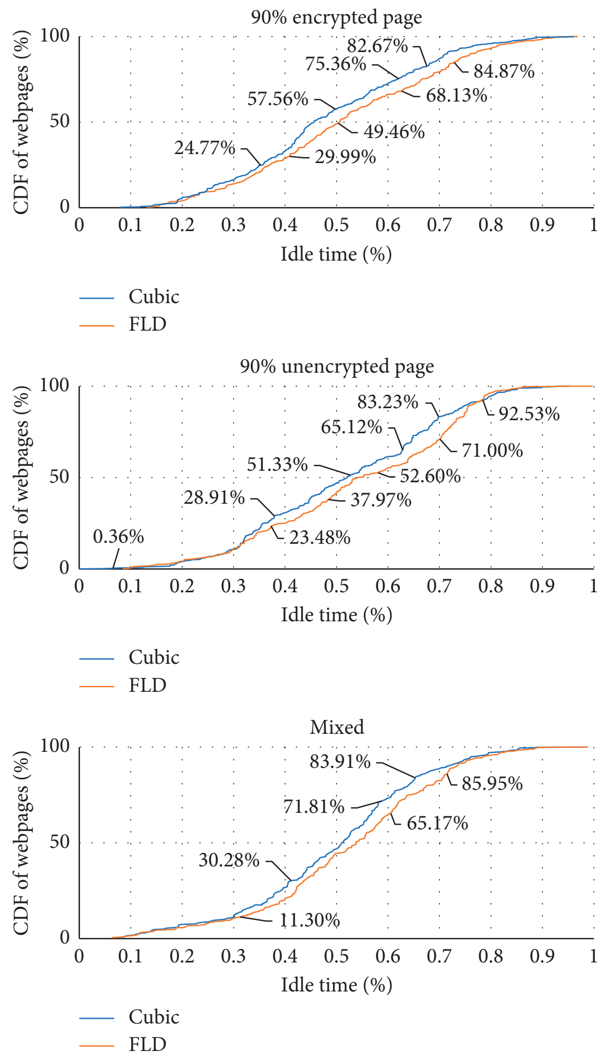


Figure 7: Distribution of idle time inside pages for different page types.

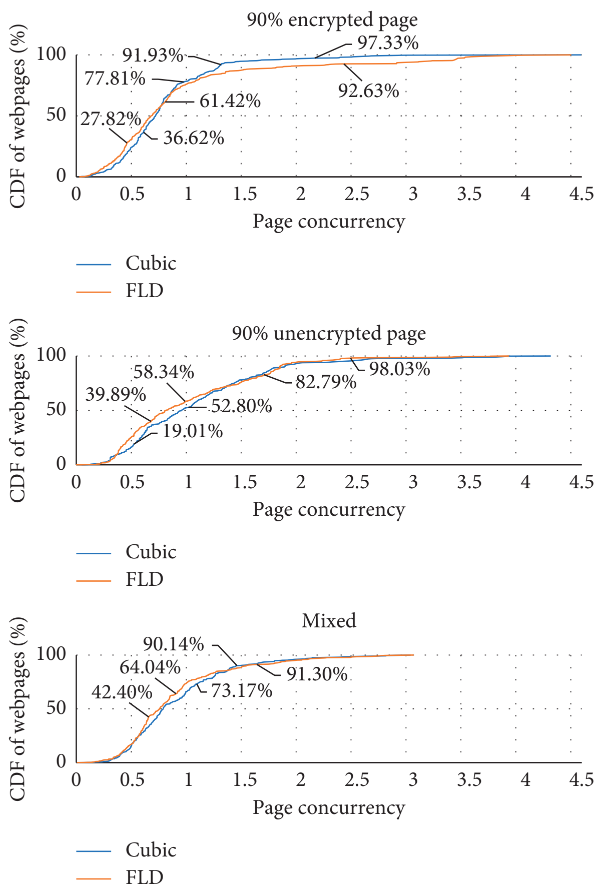


Figure 8: Effective connection concurrency across pages for different page types.

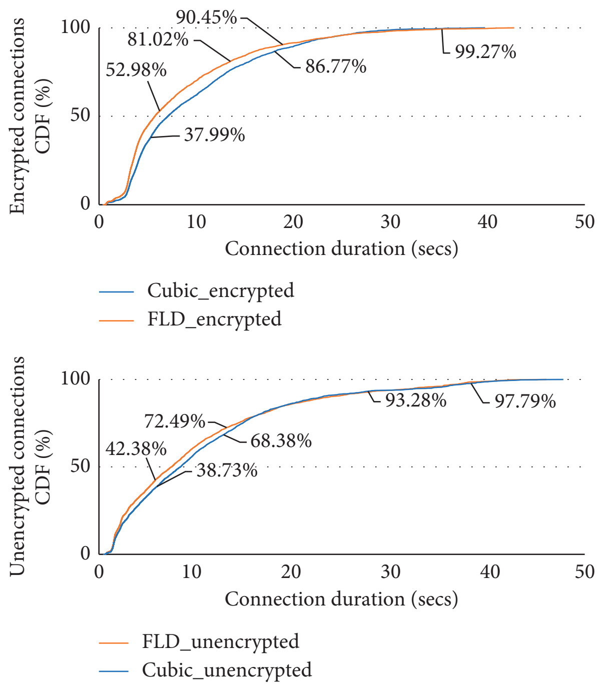


Figure 9: Distribution of connection duration across encrypted and unencrypted connections.

For service providers, even a marginal improvement of web latency translates to significant increase of user retention rates. Thus, using these methods to improve their services can have high economic impact. For modern wireless devices, the energy cost per bit for wireless transmission is 1000 times more than the energy cost for single computation of that bit [31]. By reducing the data transmission duration, there lies a huge energy saving potential as well.

5. Conclusion

In this paper, we analyzed the connection characteristics of web data transport. We also analyzed the effect of using an aggressive transport protocol for web data transfer. We used the transport protocol at a proxy but it can also be used at the content servers to boost the data transport for short flows. We found that the an aggressive transport protocol such as FLD_TCP reduces the load time of web pages by saving RTT while fetching web objects and reducing the connection duration as compared to Cubic TCP. It also decreases the

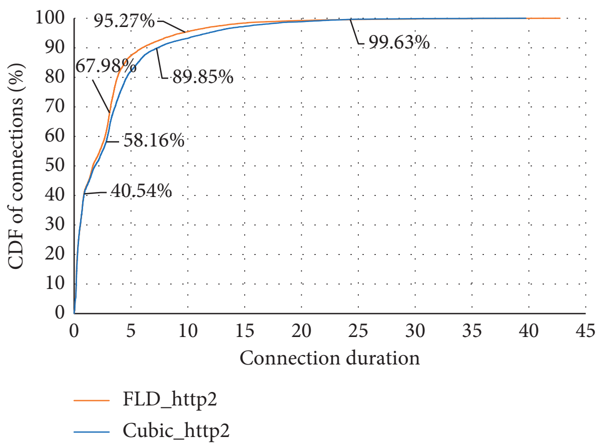


Figure 10: Distribution of connection duration across HTTP/2 connections.

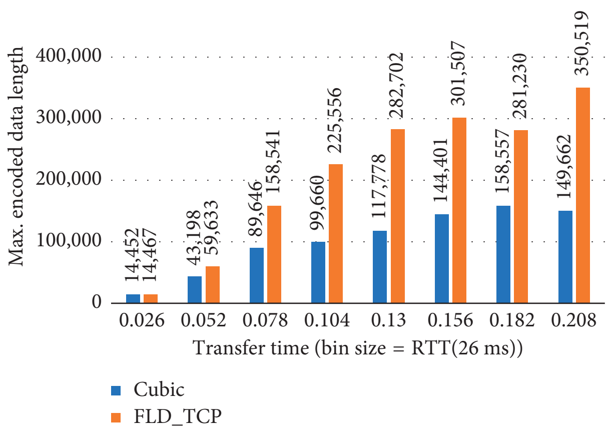


Figure 11: Maximum size of web objects transferred in each RTT bin.

concurrency of flows, thereby creating fewer TCP connections which come with connection setup overheads required to serve a page.

Improving web user experience requires optimization on both browsers and the data transport. Networks are evolving towards 5G, SDN, and cloudlet architectures. These mechanisms of alternate transport protocols will help realize low latency services for improved user experiences in general.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. T. Fielding, J. Gettys, J. C. Mogul et al., *Hypertext Transfer Protocol–http/1.1, RFC 2616, RFC Editor*, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt.

[2] Http Archive, January 2017, http://httparchive.org/index.php.

[3] R. Fielding and J. Reschke, *Hypertext Transfer Protocol (http/1.1): Message Syntax and Routing, RFC 7230, RFC Editor*, June 2014, http://www.rfc-editor.org/rfc/rfc7230.txt.

[4] I. Cooper, I. Melve, and G. Tomlinson, *Internet Web Replication and Caching Taxonomy, RFC 3040*, January 2001, https://rfc-editor.org/rfc/rfc3040.tx.

[5] B. Tierney, "Tcp tuning guide for distributed applications on wide area networks," *USENIX & SAGE Login*, vol. 26, no. 1, pp. 33–39, 2001.

[6] DNS Prefetching-The Chromium Projects, July 2017, http://www.chromium.org/developers/design-documents/dns-prefetching.

[7] Inline Small Resources, July 2017, https://developers.google.com/speed/pagespeed/service/InlineSmallResources.

[8] The Top 500 Sites on the Web, January 2017, http://www.alexa.com/topsites.

[9] M. Satyanarayanan, "Cloudlets: at the leading edge of cloud-mobile convergence," in *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures, ACM*, pp. 1-2, Vancouver, BC, Canada, June 2013.

[10] A. Sehati and M. Ghaderi, "Network assisted latency reduction for mobile web browsing," *Computer Networks*, vol. 106, pp. 134–150, 2016.

[11] R. Netravali, A. Sivaraman, S. Das et al., "Mahimahi: accurate record-and-replay for HTTP," in *Proceedings of the USENIX Annual Technical Conference*, pp. 417–429, Santa Clara, CA, USA, July 2015.

[12] A. Sivakumar, S. Puzhavakath Narayanan, V. Gopalakrishnan, S. Lee, S. Rao, and S. Sen, "Parcel: proxy assisted browsing in cellular networks for energy and latency reduction," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, ACM*, pp. 325–336, Sydney, NSW, Australia, December 2014.

[13] J. Marszałkowski, J. Mizgajski, D. Mokwa, and M. Drozdowski, "Analysis and solution of CSS-sprite packing problem," *ACM Transactions on the Web*, vol. 10, no. 1, pp. 1–34, 2016.

[14] V. Agababov, M. Buettner, V. Chudnovsky et al., "Flywheel: Google's data compression proxy for the mobile web," in *Proceedings of the Networked Systems Design and Implementation (NSDI 15)*, vol. 15, pp. 367–380, Oakland, CA, USA, May 2015.

[15] Opera Turbo, January 2017, http://www.opera.com/turbo.

[16] T. Callahan, M. Allman, and M. Rabinovich, "On modern DNS behavior and properties," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 7–15, 2013.

[17] R. Netravali, A. Goyal, J. Mickens, and H. Balakrishnan, "Polaris: faster page loads using fine-grained dependency tracking," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), USENIX Association*, Santa Clara, CA, USA, March 2016.

[18] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, and V. Sekar, "Klotski: reprioritizing web content to improve user experience on mobile devices," in *Proceedings of the Networked Systems Design and Implementation (NSDI 15)*, pp. 439–453, Oakland, CA, USA, May 2015.

[19] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving user perceived page load times using gaze," in *Proceedings of the Networked Systems Design and Implementation (NSDI 17)*, pp. 545–559, Boston, MA, USA, March 2017.

[20] R. Netravali and J. Mickens, "Prophecy: accelerating mobile page loads using final-state write logs," in *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), USENIX Association*, Renton, WA, USA, April 2018.

[21] X. Nie, Y. Zhao, G. Chen et al., "Tcp wise: one initial congestion window is not enough," in *Proceedings of the IEEE*

36th International Performance Computing and Communications Conference (IPCCC), 2017, IEEE, pp. 1–8, Orlando, Florida, USA, December 2017.

[22] K. Gangadhara Rao, B. B. Rao, and K. Chandan, "Measurement and analysis of TCP connection establishment phase of web server," *Asian Journal of Computer Science and Information Technology*, vol. 1, no. 2, 2013.

[23] F. Qian, S. Sen, and O. Spatscheck, "Characterizing resource usage for mobile web browsing," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, Services, ACM*, pp. 218–231, Bretton Woods, NH, USA, June 2014.

[24] J. Konorski and J. Lis, "Testing aggressive tcp configurations," in *Proceedings of the 1st International Conference on Information Technology, 2008, IEEE*, pp. 1–4, Gdansk, Poland, January 2008.

[25] H. Zhang, D. Towsley, and W. Gong, "Tcp connection game: a study on the selfish behavior of tcp users," in *Proceedings of the 13th IEEE International Conference on Network Protocols (ICNP 2005), IEEE*, p. 10, Boston, MA, USA, November 2005.

[26] M. Mathis, "Relentless congestion control," in *Proceedings of the PFLDNeT Workshop*, Tokyo, Japan, 2009.

[27] G. R. Moktan, S. Siikavirta, M. Särelä, and J. Manner, "Favoring the short," in *Proceedings of the 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE*, pp. 31–36, Orlando, FL, USA, March 2012.

[28] G. R. Moktan, S. Pokhrel, L. Schulte, M. Sarela, and J. Manner, "Performance analysis of a short flow favoring TCP," in *Proccedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI, 2014), IEEE*, pp. 134–142, Greater Noida, India, September 2014.

[29] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, *Iperf: The TCP/UDP Bandwidth Measurement Tool*, http://dast.nlanr. net/Projects.

[30] S. Sonntag, J. Manner, and L. Schulte, "Netradar-measuring the wireless world," in *Proceedings of the 11th International Symposium on Modeling and Optimization in Mobile, Ad Hoc & Wireless Networks (WiOpt 2013), IEEE*, pp. 29–34, Tsukuba, Japan, May 2013.

[31] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems*, vol. 24, no. 3, pp. 250–291, 2006.