An Anatomy of Mobile Web Performance over Multipath TCP

Bo Han Feng Qian* Shuai Hao Lusheng Ji
AT&T Labs – Research *Indiana University
Bedminster, NJ Bloomington, IN
{bohan, haos, lji}@research.att.com fengqian@indiana.edu

ABSTRACT

Leveraging multiple interfaces (*e.g.,* WiFi and cellular) on mobile devices is known to provide performance gains for applications such as bulk data transfer. In this paper, we complement prior studies by conducting, to the best of our knowledge, the first measurement study of mobile web performance over multipath TCP (MPTCP). Based on experiments using real web pages under diverse settings, we found the upper-layer web protocols can incur complex and sometimes unexpected interactions with the multipath-enabled transport layer. MPTCP always boosts SPDY while may hurt HTTP. MPTCP also helps mitigate SPDY's key limitations of being vulnerable to losses and inefficient bandwidth utilization. We provide in-depth explanations of the root causes, as well as recommendations for improving mobile web performance over MPTCP.

CCS Concepts

•**Networks** → **Network measurement;** *Transport protocols; Application layer protocols;*

Keywords

Multipath TCP; Mobile Web; Page Load Time; HTTP/1.1; SPDY; Multiplexing

1. INTRODUCTION

Mobile devices (smartphones, tablets, laptops *etc.*) are usually equipped with multiple network interfaces supporting diverse access technologies, such as WiFi and cellular. Prior research has investigated various aspects of multipath networking, including performance [11, 13], energy consumption [18, 19, 23], and mobility [12, 26]. In particular, the usage of multiple interfaces has been greatly facilitated by MPTCP, which is being standardized by the IETF [15] and has been implemented in the Linux kernel [21].

In this paper, we conduct a measurement study of the performance of web browsing, a key application on mobile devices, over MPTCP. It is well known that Internet users, including mobile customers, are very sensitive to page load time (PLT): an extra delay of 500 milliseconds may lead to up to 20% of traffic loss for some popular content provider [30]. MPTCP provides a convenient, transparent, and potentially effective way to enhance mobile web performance. Notably, in this way, not only can browsers benefit from MPTCP, but also millions of smartphone apps that use standard HTTP interface to fetch web pages. In fact, many smartphone apps are simply customized programmable browsers (*e.g.,* `WebView` in Android) [24].

The vast majority of prior efforts focus on understanding the performance and/or energy characteristics of bulk data transfer over MPTCP [11, 13, 19]. Compared to that, web browsing has several striking differences. First, loading a web page is a complex procedure interleaved with network transfer and local computation, with the latter contributing to a considerable fraction of the PLT [29]. Second, web browsing usually employs many short-lived TCP connections with only a few round-trips, which may limit the performance gains from MPTCP. Third, there exist different web protocols (*e.g.,* HTTP/1.1, SPDY [5] and HTTP/2 [3]) that may cause diverse and complex interactions with the underlying multipath-enabled transport layer. Our study is motivated by all above factors. We investigate the following questions unanswered by prior efforts, under the multipath scenario that concurrently accesses WiFi and cellular networks, using professionally designed real web pages.

- How much PLT reduction can off-the-shelf MPTCP achieve, compared to using single-path TCP (SPTCP)?
- How does the gain change under different network conditions (*e.g.,* various latencies and loss rates) of each path?

CoNEXT '15 December 01-04, 2015, Heidelberg, Germany
 ISBN 978-1-4503-3412-9/15/12…$15.00
DOI: http://dx.doi.org/10.1145/2716281.2836090

• How do different web protocols interact with MPTCP? We study two representative protocols with distinctive traffic patterns: HTTP/1.1 and SPDY. HTTP/1.1 uses a number of concurrent short-lived connections, while SPDY multiplexes web objects into a single connection with a longer duration. Note that HTTP/2, recently standardized in 2015, is based on SPDY and uses a similar multiplexing mechanism.

We summarize our main contributions. First, we develop a cross-layer analysis tool that incorporates multipath information into web performance analysis (§3.1), greatly facilitating our measurement study. Second, we systematically compare HTTP/SPDY performance over SPTCP and MPTCP under diverse settings, and reveal why multiplexing-based web protocol is superior to traditional HTTP/1.1 in a multipath environment (§4.1 and §4.2). Third, based on our findings, we make concrete recommendations on efficient use of MPTCP for web browsing (§4.3).

2. RELATED WORK

MPTCP in Wireless Networks. Besides being used in datacenter networks [25], MPTCP also benefits mobile devices with multiple interfaces. Chen *et al.* [11] and Deng *et al.* [13] evaluated the file downloading performance of SPTCP and MPTCP over WiFi and cellular networks. Peng *et al.* [23] proposed path selection and congestion control algorithms for energy-efficient MPTCP on mobile devices. The mobile kibbutz system [18] leverages MPTCP to consolidate multiple users' cellular traffic onto fewer links, leading to improved energy efficiency. Croitoru *et al.* [12] investigated how to achieve seamless WiFi mobility among multiple APs through MPTCP. Compared to all work above, we focus on MPTCP's application on web browsing, a very important but under-explored topic.

Mobile Web. Numerous works have been done on mobile web performance. To name a few, Erman *et al.* [14] studied SPDY over cellular, and identified performance issues due to cross-layer interactions between TCP and the radio layer. Qian *et al.* [24] characterized bandwidth and energy usage of browsing the top 500 websites on smartphones. PARCEL [27] uses "bundled push" to reduce page load time and energy consumption for mobile web. Klotski [9] improves mobile web performance by dynamically reprioritizing web resources. Flywheel [7] is Google's data compression HTTP proxy for the mobile web, which has been a production system for years. None of the above investigates the performance of mobile web over MPTCP.

3. MEASUREMENT METHODOLOGY

In this section, we describe our analysis tool, experimental testbed, and measurement methodology.

3.1 The tcpdump-mpw Tool

We develop a tool called **`tcpdump-mpw`** ("mpw" stands for "multipath web"). It is an MPTCP-friendly version of **`tcpdump`** that can extract HTTP/SPDY request/response data from raw packet traces. It takes as input a **`pcap`** capture and the SPDY proxy's TLS private key, and then performs the following processing from lower to higher layers: (1) MPTCP subflow assembling, (2) MPTCP logical connection assembling, (3) TLS/SSL decryption, (4) HTTP/SPDY parsing, and (5) web object information extraction. The output of **`tcpdump-mpw`** is a table providing details about each HTTP/SPDY transaction, including its URL, raw size, decoded size, content type, expiration time, and most importantly, the begin/end time and the amount of raw data transferred over each MPTCP subflow. As we will show, **`tcpdump-mpw`** allows us to perform in-depth study of cross-layer interactions between MPTCP and upper-layer web protocols that no prior measurement has revealed. **`tcpdump-mpw`** is built based on Ubidump [24] by adding the MPTCP and SPDY v3.1 logic, as well as detailed statistics output. We added about 2,300 lines of code to Ubidump. We will extensively use it later in our measurement data analysis in §4 and plan to release its source code to the public.

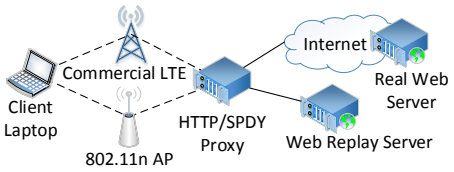

Figure 1: Our multipath measurement testbed.

3.2 Multipath Measurement Testbed

We set up a multipath measurement testbed shown in Figure 1. The Chrome browser (version 41) on a laptop fetches pages from an HTTP proxy (Squid 3.4.7 [6]) or a SPDY proxy (nghttpx 0.7.12-DEV [2]) using HTTP/1.1 or SPDY v3.1, respectively. The proxies are employed for two reasons. First, most today's websites do not yet support SPDY. Second, by making HTTP and SPDY proxies co-located, we ensure the same network path for them to allow fair comparison. The laptop is equipped with a built-in WiFi interface and an external LTE modem. There are two available network paths between the client and the proxy: a commercial LTE network of a major U.S. carrier and an 802.11n WiFi network dedicated for the testbed. Both LTE and WiFi have good signal strengths. The WiFi network is operating on the 5 GHz frequency band to minimize the interference from other users. To emulate diverse network conditions, we use Dummynet [10] to add delay, loss, or throttle throughput to the two wireless paths. The proxy fetches pages from either a replay server [1] hosting pre-recorded pages, or real websites via Internet. The proxy and the replay server are co-located on the same physical machine. The physical server and the WiFi access point are connected to a corporate network and are in the same building. Both the client and the proxy machine run Ubuntu Linux kernel 3.14.33 with the stable release of MPTCP v0.89.5 [21].

Website	# Obj	Size (KB)	# Txt Obj	# JS/ CSS	# Img Other	RTT (ms)
News	188.3	4450.8	33.8	67.7	86.8	20.9
Univ.	68.0	2583.9	3.0	16.0	49.0	30.1
AD	19.0	1459.6	3.0	6.0	10.0	62.0
News	201.2	3821.2	38.9	58.0	104.3	10.7
Tech.	67.0	2152.2	7.0	30.0	30.0	22.2
Video	93.6	2662.9	11.0	32.6	50.0	71.9
Football	99.1	2456.2	9.0	39.1	51.0	20.9
SHOPN	52.2	1000.6	5.0	12.0	35.2	8.97
Radio	66.2	2453.0	16.0	23.0	27.2	3.65
Wiki	28.0	601.2	1.0	6.0	21.0	8.70
Search	2.0	59.9	1.0	0.0	1.0	8.55
ENT	60.9	2170.6	13.4	25.1	22.4	71.8
Image	91.0	3275.2	3.0	15.0	73.0	6.20
Sport	119.0	2651.4	13.0	26.0	80.0	73.4
Social	69.0	1700.2	3.0	17.0	49.0	9.48
Movie	39.0	845.7	6.0	3.0	30.0	9.09
Weather	143.9	3814.2	21.4	39.7	82.7	11.7
RE	26.0	894.2	2.0	10.0	14.0	69.9
Airline	68.0	2148.6	4.0	23.0	41.0	3.15
SHOPN	48.0	2288.9	1.0	24.0	23.0	14.2
Gov.	68.0	2774.7	4.0	13.0	51.0	8.20
Travel	21.0	2000.4	1.0	2.0	18.0	3.85
Dict.	72.4	2223.1	10.8	31.7	29.9	6.30
Finance	39.7	1988.1	2.6	15.1	22.0	6.17
Market	49.0	2032.8	2.0	16.0	31.0	2.97

Table 1: Characteristics of selected 25 websites (AD – Advertisement, SHOPN – Shopping, ENT – Entertainment, and RE – Real Estate).

3.3 Automated Page Loading Experiments

We picked 25 websites summarized in Table 1. They are popular, diverse, and professionally designed. 22 out of the 25 sites are from the top U.S. Alexa-100 sites. Given the increasing proliferation of large-screen smartphones, tablets, and cellular dongles, we purposely used their full version instead of mobile version. We recorded a snapshot of each website in August 2014 using the replay server. We leverage the Chrome remote debugging interface to repeatedly load web pages. Similar to the prior work [14], the PLT is defined as the time span from issuing the *landing* page request to the reception of the **`load`** event fired by the Chrome browser. Each measurement test consists of cold-cache loadings (*i.e.,* caches are cleared) using different configurations to be described in §4, in a random order. Before loading a page, we switch the LTE interface to the high-power state (**`RRC_CONNECTED`**) by sending a few small ping packets over LTE, to avoid additional radio state transition delay. Since the state transition delay of LTE (normally less than 100ms) is much smaller than the PLT (usually around several seconds), the impact of state transition delay on the PLT should be small. The tests were repeated until we obtained results of 100 successful measurements for each website.

Path	Downlink (Kbps)	Uplink (Kbps)	RTT (ms)
WiFi	7040	2020	50
LTE	9185	2286	70

Table 2: Throughput and RTT for baseline experiments.

4. MEASUREMENT RESULTS

We conduct experiments with recorded pages in diverse scenarios (§4.1) and using pages from real websites (§4.2).

4.1 Controlled Experiments

4.1.1 Baseline

Our baseline experiments capture typical network conditions of WiFi and LTE. We employ Dummynet to apply bandwidth throttle and add extra delay on WiFi and LTE links to emulate the characteristics shown in Table 2. To make our emulation realistically reflect the network conditions in real-world settings, we choose the numbers from recent large-scale measurement studies of metropolitan wireless users for LTE [16] and WiFi [28]. For each path, Dummynet throttles the data rate when the actual bandwidth is higher than the target values. If the end-to-end RTT is less than the target RTT, Dummynet matches both numbers by injecting an additional delay. We use WiFi to carry the primary subflow that initiates the MPTCP connection setup. We expect this to be the common MPTCP configuration. Under the above settings, we perform six page loadings for each (test, website) pair: (1) HTTP over SPTCP WiFi, (2) SPDY over SPTCP WiFi, (3) HTTP over SPTCP LTE, (4) SPDY over SPTCP LTE, (5) HTTP over MPTCP, and (6) SPDY over MPTCP. Figure 2 plots the PLT distribution for each of the six modes, across all tests of the 25 websites.

Single-path Results. We first examine HTTP/SPDY over SPTCP. As shown in Figure 2, for both WiFi and (in particular) LTE, HTTP actually outperforms SPDY, by 12.9% and 21.0% (in terms of median PLT across all sites), respectively. This has been observed in previous work [29], which shows factors such as losses and large objects can deteriorate SPDY performance. In our SPDY experiments, the retransmission rate for LTE is non-trivial (0.13%), and is higher than WiFi (0.098%). This may cause the larger disparity between SPDY and HTTP over LTE than over WiFi. We attribute the low retransmission rate of WiFi to the clean 5GHz channel we use, and will consider higher WiFi loss rates in §4.1.2.

MPTCP Results. Now we shift our focus to MPTCP. Figure 2 shows two interesting statistical trends: SPDY over MPTCP performs significantly better than SPDY over SPTCP, but HTTP over MPTCP does not always outperform HTTP over SPTCP. To confirm this, we compute the following metric $G_{\text{HTTP}}(w)$ at a per-website basis:

$$(\min(P^{\text{WiFi}}_{\text{HTTP}}, P^{\text{LTE}}_{\text{HTTP}}) - P^{\text{MPTCP}}_{\text{HTTP}}) / \min(P^{\text{WiFi}}_{\text{HTTP}}, P^{\text{LTE}}_{\text{HTTP}})$$

where w is a website, $P^{\text{WiFi}}_{\text{HTTP}}$, $P^{\text{LTE}}_{\text{HTTP}}$, and $P^{\text{MPTCP}}_{\text{HTTP}}$ are the average PLT of HTTP over WiFi, LTE, and MPTCP for website w, respectively. G_{HTTP} quantifies the performance

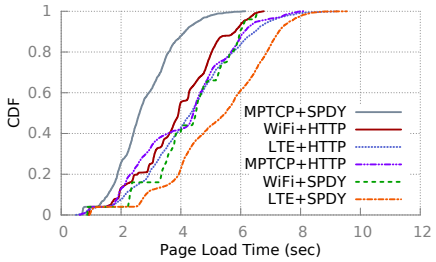

Figure 2: PLT distributions for baseline experiments (SPTCP and MPTCP).

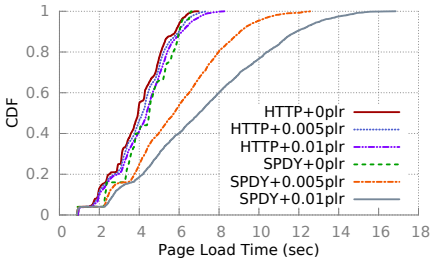

Figure 3: PLT distributions when additional losses added to WiFi (SPTCP).

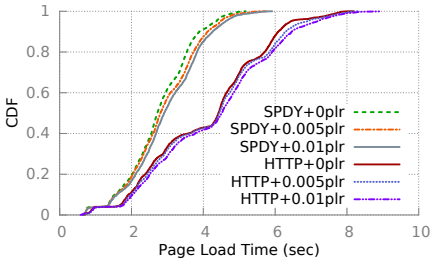

Figure 4: PLT distributions when additional losses added to WiFi (MPTCP).

gain in terms of PLT reduction by MPTCP compared to the best SPTCP case. Similar, for SPDY, we define $G_{\text{SPDY}}(w)$:

$$(\min(P^{\text{WiFi}}_{\text{SPDY}}, P^{\text{LTE}}_{\text{SPDY}}) - P^{\text{MPTCP}}_{\text{SPDY}}) / \min(P^{\text{WiFi}}_{\text{SPDY}}, P^{\text{LTE}}_{\text{SPDY}})$$

Across the 25 websites, G_{SPDY} is always positive ranging from 13.4% to 51.8% with a median of 36.4%, confirming that MPTCP always helps SPDY. In contrast, G_{HTTP} ranges between -41.1% and 30.8% with a median of -0.1%, indicating for 52% of our sites, using MPTCP actually hurts HTTP.

Since a negative G_{HTTP} is not intuitive, we repeated the HTTP experiments under identical settings using a different client, PhantomJS [4], which is extensively used by Netflix, Twitter and LinkedIn for testing. We found similar results: G_{HTTP} ranges from -48.4% to 29.9% with a median of 1.5%, indicating the issue is not specific to Chrome.

We employ `tcpdump-mpw` to explore the causes. As an example, the three subplots in Figure 5 show partial waterfall diagrams of loading the same web page using (a) HTTP over MPTCP, (b) HTTP over SPTCP (WiFi), and (c) SPDY over MPTCP. The sets of objects in each diagram are similar. Clearly, the loading pattern for HTTP over MPTCP is inefficient because objects are transferred sequentially on Connection 8 without being requested by the browser in parallel (we verified there is at least one idle connection for doing so). We found such a pattern is prevalent in HTTP over MPTCP for some sites, thus causing their negative G_{HTTP}. It does not appear in SPTCP or SPDY. We found similar issues for Chrome version 45 released at the end of May 2015.

Figure 5 indicates that MPTCP may cause unexpected interference due to applications' unawareness of it. In our discovered case, it is likely that the browser gives that connection (*i.e.,* Connection 8 in Figure 5(a)) a high priority based on various heuristics for selecting a connection[1]. In contrast, as shown in Figure 5(c), the issue does not appear in SPDY over MPTCP, simply because SPDY only uses a single multiplexing connection.

[1]We found the pattern in Figure 5(a) happens only after the involved Connection 8 is used to transfer a very large object using both paths. One possible reason may be that the browser estimates the connection's congestion window to be large due to its use of MPTCP, and thus the browser may think using that connection to transfer objects sequentially takes shorter time than using other idle connections.

Figure 2 indicates SPDY's performance improves dramatically when switching the transport protocol from SPTCP to MPTCP. The cause is two-folded. First, SPDY is known to suffer from bandwidth under-utilization in particular during slow start, due to its usage of a single connection [29]. Fortunately, this issue is alleviated by MPTCP, whose effective congestion window is the sum of those of its subflows. In contrast, most HTTP flows are short-lived, giving them fewer chances to use both paths. For the two MPTCP modes, all SPDY connections use both WiFi and LTE, while the fraction is only 47.4% for HTTP (examples shown in Figure 5(c) and (a), respectively). Second, SPDY's vulnerability to losses, which is again attributed to its usage of a single connection, is also mitigated by MPTCP. If a loss happens on one path, MPTCP can use another one to minimize the impact of loss. We investigate this in detail in §4.1.2.

4.1.2 Packet Loss

It is known that over SPTCP, packet losses impact SPDY and HTTP differently. They force SPDY's single TCP connection to reduce its congestion window. While doing so is necessary for congestion losses, it is often too conservative for non-congestion (random) losses that commonly occur in WiFi networks. Even in cellular networks, spurious retransmissions can cause a similar problem [14]. In contrast, in HTTP/1.1, a non-congestion loss only affects one of its parallel connections, leading to a smaller impact.

We confirmed this for HTTP/SPDY over SPTCP by adding to the WiFi path additional losses (0.5% and 1%, following a prior study of SPDY [29]). The original WiFi loss rate is less than 0.1%. As shown in Figure 3, SPDY performance quickly deteriorates as the packet loss rate (PLR) increases: the median PLT for 0%, 0.5%, and 1% additional PLR are 4.504s, 5.756s, and 6.992s, respectively, while for HTTP, the corresponding PLT are 3.937s, 4.151s, and 4.369s, respectively, indicating HTTP's robustness to losses.

Does the same disparity occur when using MPTCP? To answer this question, we repeat the above experiment using MPTCP. We inject additional losses (0.5% and 1%) only to the WiFi path as we expect packet losses are more likely to occur due to the contention-based channel access of WiFi. As shown in Figure 4, MPTCP helps significantly mitigate

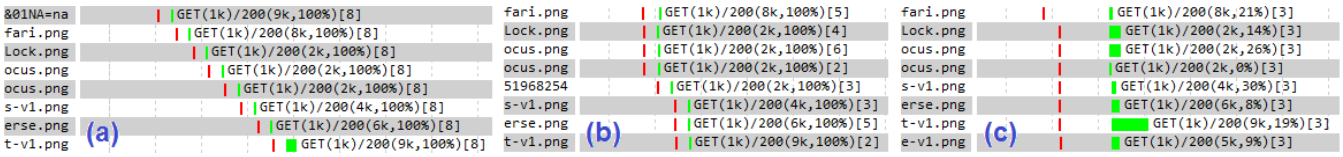

Figure 5: Partial waterfall diagrams of loading the same web page using: (a) HTTP over MPTCP, (b) HTTP over SPTCP (WiFi), and (c) SPDY over MPTCP. The text associated with each URL (only last 8 characters shown) has the following meaning. For example, "`GET(1k)/200(8k,21%)[3]`" means the size of `GET` request is 1KB, the size of its `200OK` response is 8KB, 21% of the response data is transferred over the primary WiFi path, and the whole transaction occurs on (MP)TCP connection 3. The red and green bars correspond to HTTP request and response, respectively.

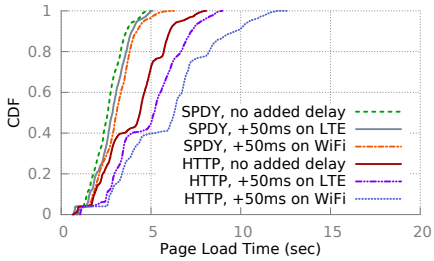

Figure 6: PLT distributions when additional latency added to each path (MPTCP).

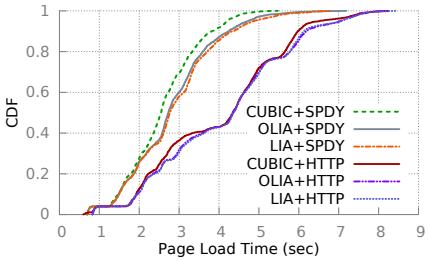

Figure 7: PLT distributions under different congestion control algorithms (MPTCP).

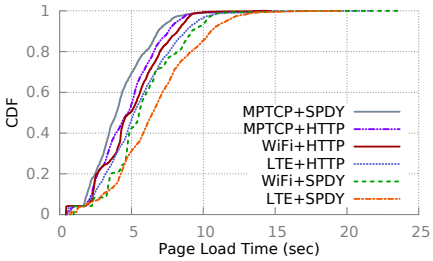

Figure 8: PLT distributions for real websites (SPTCP and MPTCP).

the impact of packet loss on SPDY, making its median PLT differ by no more than 6.92% under the three lossy conditions. This is because MPTCP's default scheduling algorithm prefers to use the less lossy LTE path when losses occur over WiFi. We confirm this by studying the relationship between PLR and the overall fraction of traffic transferred over LTE. For SPDY, under 0%, 0.5%, and 1% additional PLR, 62.2%, 66.3%, and 69.2% (median) of TCP payload is delivered over LTE. For HTTP, the corresponding fractions are 35.5%, 36.4%, and 37.0%, respectively, whose smaller differences are attributed to the fact that HTTP flows are very short and thus the LTE path has much smaller chances of being used.

4.1.3 Latency

Next, we consider how latency affects MPTCP performance by injecting additional delay to each individual path. Figure 6 plots the PLT distributions in three network conditions: the baseline with no additional delay (§4.1.1), with 50ms extra delay only on the WiFi path, and with 50ms extra delay only on the LTE path. We have three main observations. First, similar to what we discussed before, SPDY over MPTCP in general outperforms HTTP over MPTCP, even when additional latency is present.

Second, for both HTTP and SPDY, adding a delay to the WiFi path incurs more performance degradation than doing so to the LTE path. This is explained as follows. The default scheduler of MPTCP prefers the path with a *smaller* RTT, which plays a major role in determining the PLT. Recall that originally the WiFi and LTE links have a 50ms and 70ms RTT, respectively. Adding a 50ms delay to WiFi changes the smaller RTT, leading to a greater performance penalty compared to adding a 50ms RTT to LTE that makes the smaller RTT remain the same. We also confirmed this by measuring per-path traffic volume. For HTTP, originally, 62.3% (median) of downlink payload is delivered over WiFi. After adding a 50ms delay to LTE, the fraction remains similar at 69.0%, while adding a 50ms delay to WiFi dramatically reduces it to 37.1%.

Third, the performance of HTTP over MPTCP is more sensitive to delays compared to SPDY over MPTCP. As illustrated in Figure 6, the maximum difference (in terms of median PLT) among the three delay configurations is 44% for HTTP and only 19% for SPDY. Recall that the overall PLT consists of two components: network transfer and local computation. Since the network time of SPDY is smaller and thus contributes less to the PLT than the network time of HTTP does (assuming HTTP and SPDY incur similar local computation time for the same site), a proportional increase of network time caused by injected delay makes HTTP's overall PLT grow faster than that of SPDY.

4.1.4 Congestion Control

We examine the impact of MPTCP's congestion control (CC) algorithms on PLT. There are currently two major CC algorithms proposed for MPTCP: LIA (Linked Increase congestion control Algorithm) [31] and OLIA (Opportunistic LIA) [17]. Both LIA and OLIA couple the congestion window increases of MPTCP subflows and thus facilitate fairness between MPTCP flows and regular SPTCP flows. We can also directly apply conventional CC (*e.g.,* the default CUBIC CC in Linux) to MPTCP so each MPTCP subflow is treated independently as a regular TCP flow. This is called decoupled CC and is more aggressive than LIA and OLIA.

Figure 7 compares three CC algorithms, LIA, OLIA and CUBIC, whose impact on the PLT is small, in particular for HTTP. This is because HTTP/1.1 uses many concurrent short-lived connections (median duration measured to be 0.371 seconds) most of which terminate at the slow start phase so CC does not yet have a chance to operate (assuming a low loss rate). This is in line with prior comparison of MPTCP CC in the context of downloading small files [11]. Multiplexed TCP flows in SPDY are longer (median duration 2.743 seconds), leading to some disparities among the three CC algorithms in Figure 7. As expected, the decoupled CUBIC exhibits lower PLT than the two coupled CC. However, the difference (8.7% for the median PLT) is not as significant as that of file download time. We use the baseline setting (§4.1.1) to download a 4MB file. The average download time for CUBIC, LIA, and OLIA is 4.37s, 5.45s, and 5.12s, respectively, exhibiting a 25% difference. Differently from file download, due to the local computation overhead in web browsing, its network transfers are less likely to be throttled by the congestion window.

4.2 Real Websites

We now change the proxies' configuration to fetch pages from real websites. This differs from the previous replay experiments in three aspects. First, in replay, the latency between the proxies and the replay server is negligible. This helps us capture the *upper bound* of MPTCP's impact. In contrast, the Internet path between the proxies and the real websites incurs non-trivial latency (summarized in the last column of Table 1) that penalizes both MPTCP and SPTCP. Second, real websites may be overloaded, incurring additional processing delay. Third, most websites have contents from several non-origin sources that contribute a significant fraction of the downloaded bytes [8].

Figure 8 plots the PLT distributions under the same settings used in §4.1.1 except that the proxies now connect to real websites. Due to the aforementioned factors, compared to Figure 2, the disparities among the six schemes become smaller. Nevertheless, we observe similar trends as found in replay experiments: over SPTCP, SPDY performs slightly worse than HTTP while over MPTCP, SPDY non-trivially outperforms HTTP by 17.0% (median PLT). Also, SPDY over MPTCP considerably excels SPDY over SPTCP (WiFi) by 28.6% and SPTCP (LTE) by 38.9%. Note that MPTCP will exhibit more advantages as more web servers support it, because in this case no proxy is required and the shared proxy-server path is therefore eliminated.

4.3 Summary and Recommendations

We summarize our findings and provide recommendations of *when* and *how* to better use MPTCP for web browsing.

- Multiplexing-based web protocols, such as SPDY and HTTP/2, gain more benefits from MPTCP than HTTP/1.1 does. The reason is that MPTCP alleviates their two key limitations: vulnerability to loss and inefficient bandwidth utilization (assuming a single multiplexing connection is used). In contrast, due to its connections being short-lived, HTTP/1.1's chances of utilizing both paths are much slimmer. Given the increasing popularity of HTTP/2, our findings provide an incentive to enable MPTCP for mobile web browsing.

- We found for Chrome, HTTP over MPTCP is sometimes worse than HTTP over SPTCP, possibly because MPTCP's interference leads to suboptimal TCP connection management decisions. At the high level, it implies the importance of browsers being aware of MPTCP and its potential implications such as high latency/bandwidth variation.

- Since web browsing is usually delay-sensitive, when using the default scheduler of MPTCP, the PLT is largely determined by the path with a lower latency when both paths' bandwidth and loss do not differ too much. Browsers can leverage it to determine whether to enable MPTCP: having a secondary path with a small latency (despite low bandwidth) is usually more beneficial than having one with high bandwidth and latency.

- Although MPTCP may significantly boost the performance of network transfer, when page loading is throttled by local/remote computation, the gain of enabling MPTCP may be limited (§4.2). For a similar reason, MPTCP's congestion control plays a minor role in determining the PLT.

5. DISCUSSION

In this section, we discuss several unaddressed issues in this paper.

Client Device. Throughout this study, we used a laptop as the client device and loaded full web pages that are not optimized for handheld screens. Changing the client to a smartphone or tablet may shrink the disparities observed in §4 due to the increased fraction of local computation time [20]. However, we expect the overall trends to remain the same. We plan to test the mobile version of these web pages on smartphones and tablets in our future work.

Energy Consumption. As the key focus of this study is performance, one important aspect we did not examine is energy consumption. Prior study shows that due to high energy cost of LTE radio, simultaneously using WiFi and LTE for file transfer may increase the overall smartphone energy usage [19]. A similar issue exists in web browsing over MPTCP. However, since MPTCP can reduce the PLT, the quantitative impact of MPTCP on the overall energy consumption still needs to be explored in our future work.

Mobility. MPTCP can also facilitate mobility [12, 26]. Consider a scenario where a user keeps browsing the web as she walks out of WiFi coverage so her phone is forced to switch to cellular. To avoid the incurred interruption, one can leverage MPTCP's backup mode [13] to ensure the smooth transition. As web browsing is delay sensitive, the LTE interface can be preemptively enabled when the WiFi signal becomes weaker, to further reduce the transition latency.

Other limitations. We discuss other limitations in our study. (1) We only consider one combination of typical WiFi/LTE network conditions (§4.1.1), followed by changing individual parameters (loss, delay and congestion control) one by one. A more comprehensive approach is to thoroughly evaluate many combinations of both paths' characteristics. (2) We mainly conduct experiments using the Chrome browser (and perform some key tests using PhantomJS) and the nghttpx proxy. Despite their popularity, it is ideal to also test on other browsers and proxies. (3) We are not able to test MPTCP on production web servers since most of them do not yet support MPTCP.

6. CONCLUDING REMARKS

To the best of our knowledge, we have conducted the first measurement study of mobile web performance over MPTCP. Using `tcpdump-mpw`, our cross-layer analysis tool, we systematically compared HTTP/SPDY performance over SPTCP and MPTCP under diverse settings, and discovered unexpected interactions between MPTCP and HTTP/SPDY. Besides the future work mentioned above, we also plan to evaluate mobile web over MPTCP in a range of heterogeneous environments [22], as well as to leverage the insights obtained from this measurement study for improving mobile web performance.

Acknowledgments

We thank Vijay Gopalakrishnan, Ginger Chien, Rittwik Jana and the anonymous reviewers for their valuable comments and suggestions, and in particular David Choffnes for shepherding the paper. This research was sponsored in part by Indiana University Faculty Research Support Program (FRSP) – Seed Funding.

7. REFERENCES

[1] Google Web Page Replay Tool. https://github.com/chromium/web-page-replay/.

[2] HTTP/2 C Library and Tools. https://nghttp2.org/.

[3] Hypertext Transfer Protocol version 2. http://http2.github.io/http2-spec/index.html.

[4] PhantomJS, a headless WebKit. http://phantomjs.org/.

[5] SPDY Protocol – Draft 3.1. http://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3-1.

[6] Squid Caching and Forwarding Web Proxy. http://www.squid-cache.org/.

[7] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin. Flywheel: Google's Data Compression Proxy for the Mobile Web. In *NSDI*, 2015.

[8] M. Butkiewicz, H. V. Madhyastha, and V. Sekar. Understanding Website Complexity: Measurements, Metrics, and Implications. In *IMC*, 2011.

[9] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, and V. Sekar. Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In *NSDI*, 2015.

[10] M. Carbone and L. Rizzo. Dummynet Revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12–20, Apr. 2010.

[11] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *IMC*, 2013.

[12] A. Croitoru, D. Niculescu, and C. Raiciu. Towards WiFi Mobility without Fast Handover. In *NSDI*, 2015.

[13] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both? Measuring Multi-homed Wireless Internet Performance. In *IMC*, 2014.

[14] J. Erman, V. Gopalakrishnan, R. Jana, and K. Ramakrishnan. Towards a SPDY'ier Mobile Web? In *CoNEXT*, 2013.

[15] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, 2013.

[16] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *SIGCOMM*, 2013.

[17] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. L. Boudec. MPTCP is not Pareto-Optimal: Performance Issues and a Possible Solution. In *CoNEXT*, 2012.

[18] C. Nicutar, D. Niculescu, and C. Raiciu. Using Cooperation for Low Power Low Latency Cellular Connectivity. In *CoNEXT*, 2014.

[19] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng. Energy and Performance of Smartphone Radio Bundling in Outdoor Environments. In *WWW*, 2015.

[20] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao. Mobilyzer: An Open Platform for Controllable Mobile Network Measurements. In *MobiSys*, 2015.

[21] C. Paasch, S. Barré, et al. Multipath TCP in the Linux Kernel. http://www.multipath-tcp.org.

[22] C. Paasch, R. Khalili, and O. Bonaventure. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *CoNEXT*, 2013.

[23] Q. Peng, M. Chen, A. Walid, and S. Low. Energy Efficient Multipath TCP for Mobile Devices. In *MobiHoc*, 2014.

[24] F. Qian, S. Sen, and O. Spatscheck. Characterizing Resource Usage for Mobile Web Browsing. In *MobiSys*, 2014.

[25] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *SIGCOMM*, 2011.

[26] C. Raiciu, D. Niculescu, M. Bagnulo, and M. Handley. Opportunistic Mobility with Multipath TCP. In *MobiArch*, 2011.

[27] A. Sivakumar, S. P. Narayanan, V. Gopalakrishnan, S. Lee, S. Rao, and S. Sen. PARCEL: Proxy Assisted BRowsing in Cellular networks for Energy and Latency reduction. In *CoNEXT*, 2014.

[28] J. Sommers and P. Barford. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *IMC*, 2012.

[29] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. How Speedy is SPDY? In *NSDI*, 2014.

[30] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie. How Fast Can Client-Only Solutions Go for Mobile Browser Speed. In *WWW*, 2012.

[31] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *NSDI*, 2011.