Multipath TCP in Smartphones: Impact on Performance, Energy, and CPU Utilization

Swetank Kumar Saha, Abhishek Kannan, Geunhyung Lee, Nishant Ravichandran,
Parag Kamalakar Medhe, Naved Merchant, Dimitrios Koutsonikolas
University at Buffalo, The State University of New York, Buffalo, NY 14260-2500
{swetankk,akannan4,geunhyun,n7,paragkam,navedmer,dimitrio}@buffalo.edu

ABSTRACT

This paper explores the potential benefits and pitfalls of Multipath TCP (MPTCP) in smartphones via an extensive experimental study over real Android applications. We consider different types of applications – upload vs. download intensive, network intensive vs. interactive – and a variety of network conditions, and we study the impact of MPTCP on performance, energy consumption, and CPU utilization. Our results reveal that the benefits of MPTCP in smartphone apps are lower than expected in theory; in several cases, MPTCP in fact can hurt both performance and energy consumption. Our findings can provide insights to smartphone designers and mobile app developers towards improving user experience and extending smartphone battery life.

1 INTRODUCTION

Multipath TCP (MPTCP) [17] is a new standardized transport protocol that allows end hosts to simultaneously use multiple NICs and exploit path diversity. With MPTCP, unmodified applications, designed to run over TCP, can exchange data over different paths via one end-to-end connection. Smartphones are a natural fit for MPTCP, as they typically include both WiFi and cellular interfaces. Apple has been using Multipath TCP on iPhones and iPads since 2013 to support the Siri voice recognition application [1, 6]. In 2015, Korean Telecom launched Gigapath, a commercial service based on MPTCP and the SOCKS protocol that allows Samsung and LG smartphones to combine their 4G and WiFi networks to reach bandwidths of up to 800 Mbps [1].

In spite of the great interest from both industry and academia, the true potential of MPTCP in smartphones has not been fully understood. While in theory MPTCP can provide higher throughput and robustness than single path TCP (SPTCP), it was initially designed for datacenter networks and its benefits may not be the same when used over WiFi and cellular networks. Indeed, initial experimental studies using laptops showed that, for short flows, MPTCP may actually perform worse than SPTCP over the faster path [9, 15, 21]. Additionally, using MPTCP in smartphones can significantly impact the device energy consumption in two ways [21, 31]. On one hand, it can incur an additional energy cost from the combined usage of both NICs; on the other hand, it may reduce the energy consumption by shortening the data transfer time. More importantly, the impact MPTCP can have on real smartphone apps and the potential implications on the energy consumption due to cross-layer interactions is not yet fully understood.

This work fills the gap by exploring the potential benefits and pitfalls of MPTCP in smartphones via an extensive experimental study over real Android apps. We consider different types of apps and different network conditions, and we explore the impact of MPTCP on performance, energy consumption, and CPU utilization. Our results reveal that the benefits of MPTCP in smartphone apps are lower than expected in theory; in several cases, MPTCP in fact can hurt both performance and energy consumption. Specifically, MPTCP improves mainly the performance of data intensive apps, and mostly in locations where the two networks (WiFi and cellular) exhibit similar performance. In contrast, it can hurt the performance of data intensive apps under heterogeneous network conditions, and typically has a negative impact on the performance of interactive apps. Additionally, MPTCP often results in a higher energy cost due to the combined use of both networks. In fact, we found very few scenarios where MPTCP improved both performance and energy efficiency and, in some cases, we observed a tradeoff between the two metrics; in contrast, in most cases, MPTCP resulted in both lower performance and higher energy cost. Finally, similar to the work in [21], we found that MPTCP typically increases the CPU utilization. However, the increased utilization does not result in an increase to the energy consumption. The higher energy consumption of MPTCP compared to SPTCP in most cases is attributed to higher network energy.

The paper is organized as follows: § 2 describes the experimental methodology. § 3 and § 4 analyze the impact of MPTCP on performance, energy consumptions, and CPU utilization in the case of data intensive and interactive apps, respectively. § 5 summarizes the results and the lessons learned from our study. § 6 discusses the related work. Finally, § 7 concludes the paper.

2 METHODOLOGY

For our measurements, we used a methodology similar to that in [11–13], and their publicly available automated test framework [10] with some modifications. We used a Nexus 5 smartphone running Android 4.4.4 with a modified Linux kernel that includes MPTCP v0.89.5. MPTCP was used in Full-MPTCP mode [23]. We used the *fullmesh* path manager, the *default* RTT-based scheduler [24], and the standard *lia* coupled congestion control algorithm. Since today's application servers do not support MPTCP, we configured the smartphone to access the Internet through an MPTCP capable SOCKS5 [20] proxy server. The SOCKS server uses ShadowSocks [5]

MobiWac'17, November 21–25, 2017, Miami, FL, USA

ACM ISBN 978-1-4503-5163-8/17/11…$15.00
https://doi.org/10.1145/3132062.3132066

Table 1: Locations used in our study.

Location	WiFi Bandwidth	LTE Bandwidth
Good WiFi/Good LTE (W_gL_g)	18-25 Mbps	18-25 Mbps
Good WiFi/Bad LTE (W_gL_b)	50-60 Mbps	less than 10 Mbps
Bad WiFi/Good LTE (W_bL_g)	3-5 Mbps	18-25 Mbps
Bad WiFi/Bad LTE (W_bL_b)	3-5 Mbps	3-5 Mbps

and is configured to use the minimum encryption scheme to reduce the overhead. The phone uses the standard ShadowSocks client. This configuration has been used in a number of previous works [11–13, 22] as well as in the Gigapath project by Korean Telecom [1] and has been verified to incur a very small runtime overhead.

2.1 Applications

We used 7 popular Android apps that span several categories.
Data intensive apps: We considered bulk data transfer apps (*Dropbox*, *SpeedTest*) and streaming apps (*YouTube*, *Spotify*). With Dropbox, the automated test framework uploads every time a new file containing 20 MB of random data. With SpeedTest, it launches the app and logs the reported downlink bandwidth. With YouTube, it plays the same HD video (Big Buck Bunny) three times with the quality set to *auto* and watches each time for 25 seconds. With Spotify, it plays new music (shuffle play feature) for 75 seconds.
Interactive apps: We considered browsing (*Firefox*) and two social networking apps (*Facebook* and *Facebook Messenger*). With Firefox, the framework sequentially browses the main page of the following 12 sites with an empty cache: imgur, wikipedia, google, facebook, youtube, yahoo, baidu, amazon, twitter, nytimes, flickr, qq. With Facebook, it first updates the news feed and then repeats the following actions three times in the same order: it posts a new status, shares a photo with a description, and performs a new check-in with a text description. Finally, with Messenger, it sends a text message and a photo to the top contact in the contact list and repeats three times. Note that in the case of Facebook and Messenger, Location Services were also running.

2.2 Measurements

We ran our experiments at four locations with different network characteristics. The uplink/downlink bandwidth at each location (measured with SpeedTest) is shown in Table 1. Note that the WiFi bandwidth at the W_gL_b location (50-60 Mbps) is much higher than the LTE bandwidth at the W_bL_g location (18-25 Mbps); for the cellular provider used in our experiments, the maximum achievable throughput in the test locations is around 25 Mbps. For the W_gL_g location, we also limited WiFi bandwidth to 18-25 Mbps in order to conduct experiments with similar bandwidth for both networks. Note also that the actual application layer throughput may vary for different apps due to a number of reasons, e.g., an app may not utilize the full available bandwidth. We also considered four different network configurations: WiFi only (WiFi), LTE only (LTE), MPTCP with WiFi as the primary subflow (MP-WiFi), and MPTCP with LTE as the primary subflow (MP-LTE). At each location, we repeated each application 5 times with each of the four network configurations. We used *tcpdump* to capture all the packets on the phone and measured the total energy consumption of the device using a Monsoon power monitor [4].

2.3 Metrics

We use three metrics in our study: energy per byte, performance, and normalized CPU utilization.

2.3.1 Energy per byte. The energy per byte (in $\mu J/B$) is the ratio of the energy consumption divided by the total number of bytes exchanged between the phone and the proxy server, excluding TCP/MPTCP retransmissions. Intuitively, for a given data transfer size, poor network conditions will result in a higher number of retransmitted bytes and increased energy per byte cost compared to good network conditions.

In this work, we are interested only in the energy consumption associated with network activity E_{na}. Since it is not possible to measure the power consumption of a single packet transmission/reception with Monsoon, we only consider *data bursts*. A data burst is defined as a time interval $[t_i, t_j]$, $t_i < t_j$ that satisfies the following two conditions: (i) consecutive packets in $[t_i, t_j]$ have inter-arrival times $\Delta T \leq T_{max}$ and (ii) the number of total packets in $[t_i, t_j]$ is $N \leq N_{max}$. To identify the burst intervals for a given app, we manually synchronize the packet trace captured by tcpdump and the power trace captured by the power monitor, as shown in the examples in Figures 1a-1g. The only exception is Firefox, where we identify the data bursts based on the page load time (§ 2.3.2) – one data burst for each page. Due to the different nature of different apps, the thresholds T_{max} and N_{max} have different values for different apps, as shown in Table 2. For example, Figure 1c shows only one very large downlink data burst in YouTube (from 25-38 s), which always contained at least 1000 packets ($N_{max} = 1000$ in Table 2). In contrast, Figure 1g shows several small data bursts ($N_{max} = 10$) in Messenger.

For every synchronized data-power trace, we calculate the energy associated with network activity E_{na} from the segments of the power trace overlapping with data bursts, e.g, for YouTube in Figure 1c we only consider the energy consumption during 25-38 s. Note that, due to the very small burst sizes in Messenger, potential errors in synchronization of the Messenger traces result in higher errors in the estimation of E_{na} compared to other apps. Due to this reason, we expand each Messenger data burst by 0.1 s on each side, i.e., if our algorithm identifies $[t_i, t_j]$ as a data burst, we consider the interval $[t_i - 0.1, t_j + 0.1]$ in the energy calculation.

We further break down the total energy consumption associated with network activity into into two parts:

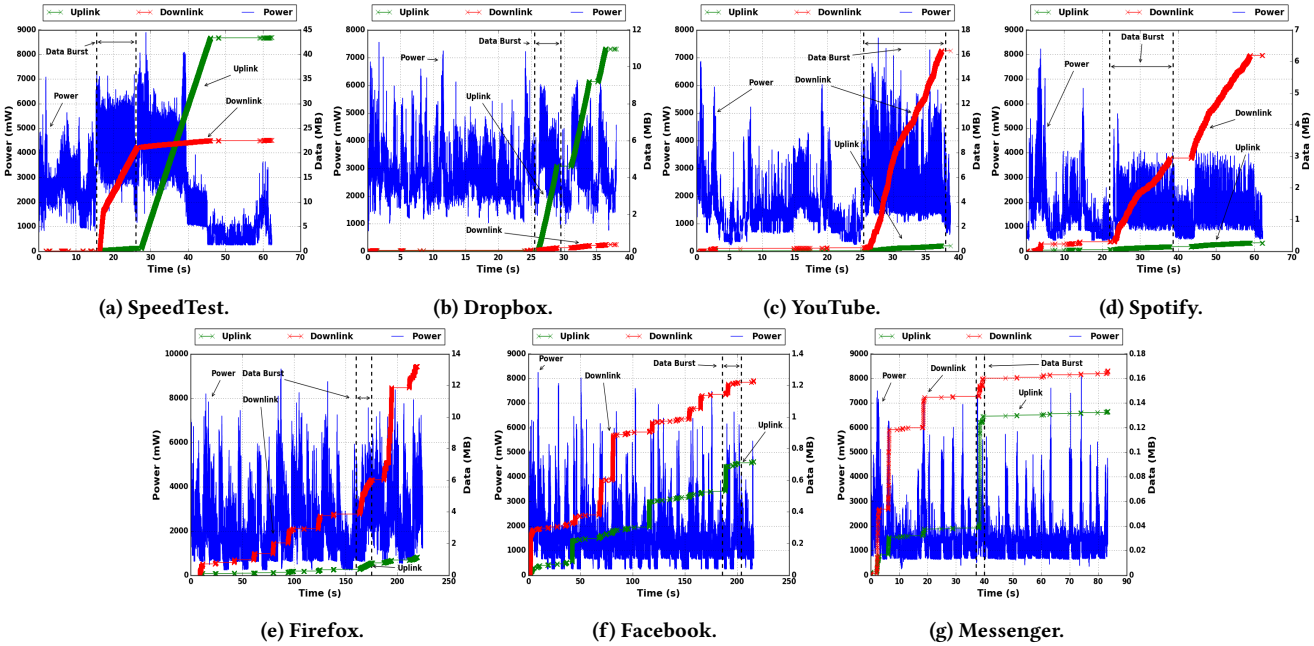
$$E_{na} = E_{cpu} + E_{data} \quad (1)$$

where E_{cpu} is the CPU energy consumption during the intervals of network activity, and E_{data} is the data transfer energy. Note that E_{data} can actually include the energy due to other hardware components (e.g., screen, GPU) which we cannot isolate. We expect this energy consumption of such components to be constant for a given application.

We estimate the CPU power consumption during data bursts E_{cpu} using a CPU power model, which we developed for Nexus 5 following the methodology in [8, 21, 33]. The model takes as input the CPU frequency and utilization for each of the four cores of Nexus 5. We wrote an app to log these values every 100 ms and used the logs as input to the model in post-processing. It was shown in [21] that a 100 ms logging interval provides good accuracy with

Table 2: Data burst thresholds for different apps.

	SpeedTest	Dropbox	YouTube	Spotify	Firefox	Facebook	Messenger
T_{max} (s)	1	0.15	1	2	Defined based on page load time	2	2
N_{max} (packets)	250	300	1000	150	N/A	100	10



(a) SpeedTest. (b) Dropbox. (c) YouTube. (d) Spotify.

(e) Firefox. (f) Facebook. (g) Messenger.

Figure 1: Examples of synchronized data-power traces and data bursts for the seven apps.

negligible overhead. E_{cpu} is the sum of the energy values predicted by the model over each 100 ms interval.

It is well known that cellular interfaces incur a *tail energy* cost, in addition to the energy consumption during a data transfer [7, 19]. While any potential tail power consumption during a data burst is already accounted for in E_{data}, there is an additional amount of tail energy at the end of each data burst, which we do not account for. Hence, our energy results for LTE and MP-LTE in § 3, § 4 are (slightly) conservative, although recent studies [8, 21] have shown that the LTE tail base power is much lower than in the past.

2.3.2 Performance. We use different performance metrics for different apps due to their different characteristics.

SpeedTest Our performance metric is the download bandwidth reported by the app.

Dropbox We calculate the upload throughput as the sum of the bytes of the identified data bursts (ignoring TCP/MPTCP retransmitted bytes) divided by the sum of the data burst durations.

YouTube, Spotify We calculate the download throughput as the sum of the bytes of the identified data bursts (ignoring TCP/MPTCP retransmitted bytes) divided by the sum of the data burst durations. A higher throughput should result in higher video/audio quality.

Firefox We calculated the page load time *PLT* for each of the twelve pages using the Firefox Developer Tools [2] and the Navigation Timing API [3] as

$$PLT = loadEventEnd - responseStart \tag{2}$$

where *responseStart* is the time immediately after the browser receives the first response byte from the server and *loadEventEnd* is the time when the load event of the current page is completed. Note that our definition does not include the DNS lookup, TCP handshake, and the time to send the HTTP GET request. Our final metric is a weighted page load time (WPLT) for N pages defined as:

$$WPLT = \sum_{i=1}^{N} \frac{S_i}{\sum_{j=1}^{N} S_j} PLT_i \tag{3}$$

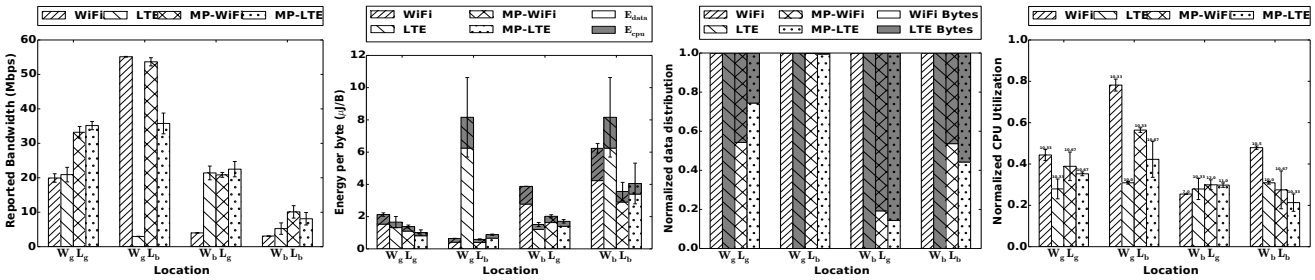
where PLT_i is the page load time of the i-th page and S_i is the size of the i-th page in bytes. In other words, we assign larger weights to larger pages.

Facebook, Messenger We use the average round trip time (RTT) as the performance metric. Intuitively, a shorter RTT should result in shorter response time and better user experience. Following the TCP approach, we do not include RTTs of retransmitted segments in our calculations.

2.3.3 Normalized CPU Utilization. We define a normalized CPU utilization metric NU_i for the i-th core, $i = 1, 2, 3, 4$ as follows:

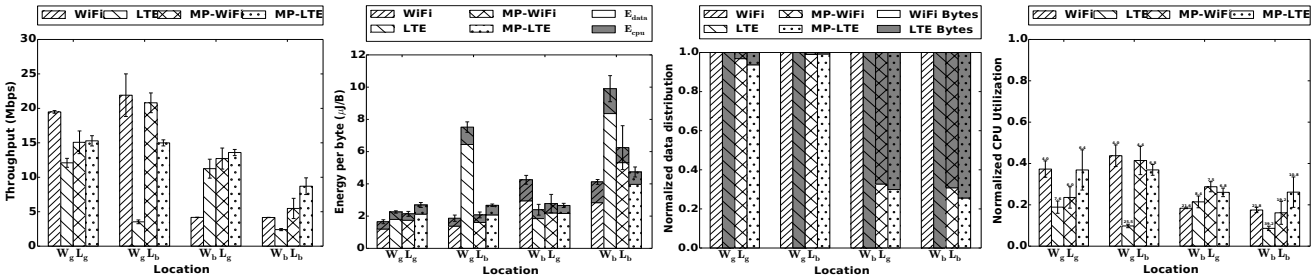
$$NU_i = u_i \cdot \frac{f_i}{f_{max}} \tag{4}$$

where u_i, f_i are the utilization and frequency of the i-th core, respectively, and $f_{max} = 2,265.6\ KHz$ is the maximum CPU frequency in Nexus 5. We further define $NU = \frac{\sum_{i=1}^{4} NU_i}{4}$ and plot the average value of NU over the data burst intervals. In the following, we annotate graphs that plot normalized CPU utilization (e.g., Figure 2d) with the total duration (in s) over which the utilization was calculated. The intuition is that high CPU utilization over a very short duration may not lead to high CPU energy consumption.



(a) Reported downlink bandwidth. (b) Energy consumption. (c) Data breakdown. (d) CPU utilization.

Figure 2: SpeedTest.



(a) Uplink throughput. (b) Energy consumption. (c) Data breakdown. (d) CPU utilization.

Figure 3: Dropbox.

3 DATA INTENSIVE APPLICATIONS

3.1 Bulk data transfer applications

3.1.1 SpeedTest. We begin our study with SpeedTest, as an example of a data intensive app that tries to utilize the full bandwidth. We expect our findings to reflect those of previous studies that conducted measurements using bulk TCP data transfers [9, 15]. We focus on the downlink component of the app.

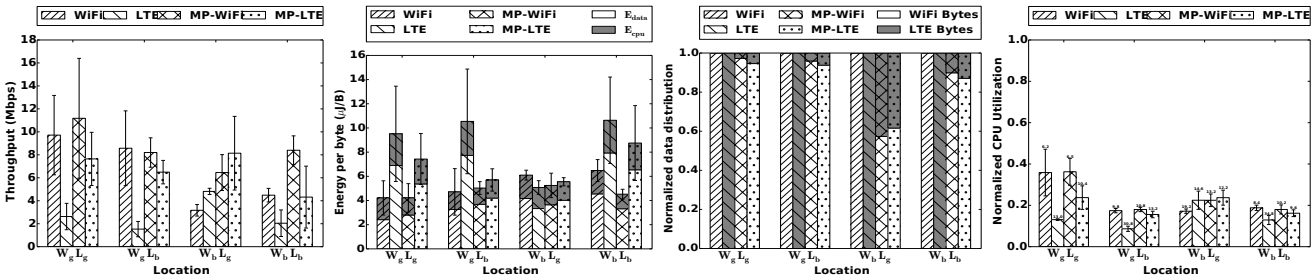
In Figure 2a, we observe that, in the two locations where the performance of the two networks is similar (W_gL_g and W_bL_b), MPTCP helps significantly, increasing the bandwidth by 65-125% over WiFi or LTE only; in particular, in location W_bL_b the MPTCP bandwidth is roughly equal to the sum of the WiFi and LTE bandwidths. Additionally, in these two locations, MPTCP decreases the energy consumption compared to SPTCP (Figure 2b), by reducing both E_{data} and E_{cpu}. In contrast, in locations with high disparity between the WiFi and LTE bandwidth (W_gL_b and W_bL_g), MPTCP is similar to or worse than SPTCP over the best path, in terms of both performance and energy consumption, as was also observed in previous studies [9, 15, 16]. The increase in the total energy is primarily due to an increase in E_{data}.

Figure 2c shows that MPTCP achieves, in general, a good balance between the two paths under homogeneous network conditions but fails to do so in cases of high heterogeneity. In particular, we observe that, in location W_gL_b where the LTE bandwidth was extremely low during our experiments, almost 100% of the bytes were downloaded over WiFi with both MP-WiFi and MP-LTE. Consequently, MP-WiFi performs almost identical to WiFi alone and consumes similar energy (Figures 2a, 2b), while MP-LTE is affected by the extra delay required for the establishment of the (secondary) WiFi subflow and experiences a significant performance drop. On the other hand, in location W_bL_g, a similar, non-negligible amount of data is transferred over the bad path (WiFi) with both MP-WiFi and MP-LTE. As a result, MP-LTE has only slightly higher performance and lower energy consumption than MP-WiFi.

Figure 2d shows that WiFi results in higher CPU utilization than LTE except in location W_bL_g, where the utilization is similar with both networks. Further, the utilization with MPTCP is between that of WiFi and LTE alone in W_gL_g, W_gL_b, sightly higher than both in W_bL_g, and lower than both in W_bL_b. Regardless of the utilization, the fraction of CPU energy over the total energy is the highest with WiFi alone in all 4 locations, and the lowest with MP-LTE.

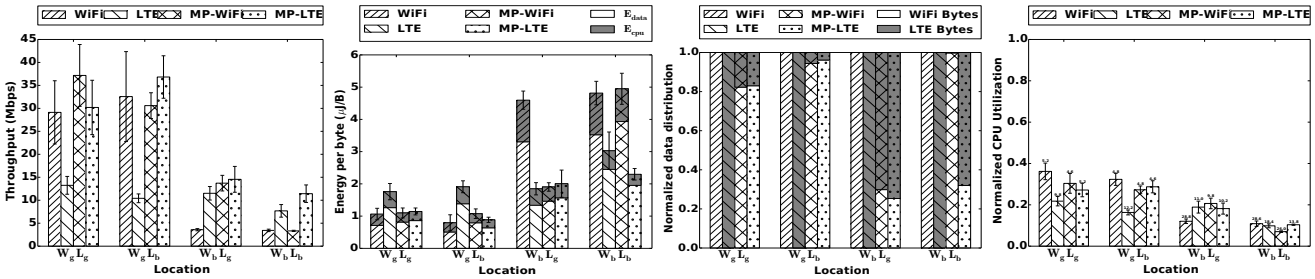
3.1.2 Dropbox. In the case of Dropbox, we study an upload intensive app. Note that Dropbox does not utilize the full bandwidth; we were never able to measure an upload throughput higher than 30 Mbps although SpeedTest reported much higher values. In Figure 3a, we observe that the performance with MPTCP lies between that with WiFi alone and LTE alone in cases of good WiFi (W_gL_g and W_gL_b). In those cases, MPTCP sends almost all the bytes over WiFi (Figure 3c), which exhibits much lower RTT even when the two networks have similar bandwidth. Consequently, MP-WiFi performs similar to (W_gL_g) or better than MP-LTE (W_gL_b). Further, the energy consumption with WiFi alone is the lowest in these two locations (Figure 3b). MPTCP incurs a higher energy cost than WiFi. In the case of MP-WiFi, the increase is due to an increase in E_{data} compared to WiFi alone while in the case of MP-LTE both E_{data} and E_{cpu} increase. As a result, the energy cost of MP-LTE is higher than that of MP-WiFi.

In contrast, MPTCP improves the performance significantly compared to WiFi alone and LTE alone in cases of poor WiFi performance (W_bL_g and W_bL_b). In particular, in location W_bL_b the



(a) Downlink throughput. (b) Energy consumption. (c) Data breakdown. (d) CPU utilization.

Figure 4: Spotify.



(a) Downlink throughput. (b) Energy consumption. (c) Data breakdown. (d) CPU utilization.

Figure 5: YouTube.

throughput with MP-LTE is slightly higher than the individual throughputs over WiFi and LTE alone. Interestingly, in both cases, the performance improvement comes at a cost of a small increase in the energy cost compared to the most energy efficient SPTCP. In location W_bL_g, the energy cost of both MP-WiFi and MP-LTE is similar, slightly higher than the cost of LTE alone due to a small increase in E_{data}. In location W_bL_b, MP-LTE is more energy efficient than MP-WiFi. Interestingly, MPTCP decreases E_{cpu} compared to WiFi alone but increases E_{data}.

Figure 3d shows that, similar to SpeedTest, WiFi results in higher CPU utilization except in W_bL_g, where the utilization is similar with both networks. In contrast to SpeedTest, MPTCP here increases the utilization both in W_bL_g and W_bL_b. The contribution of the CPU to the total energy consumption is again the highest with WiFi alone in all 4 locations, and the lowest with MPTCP except for W_gL_b, where the CPU energy is minimum with LTE alone.

3.2 Streaming

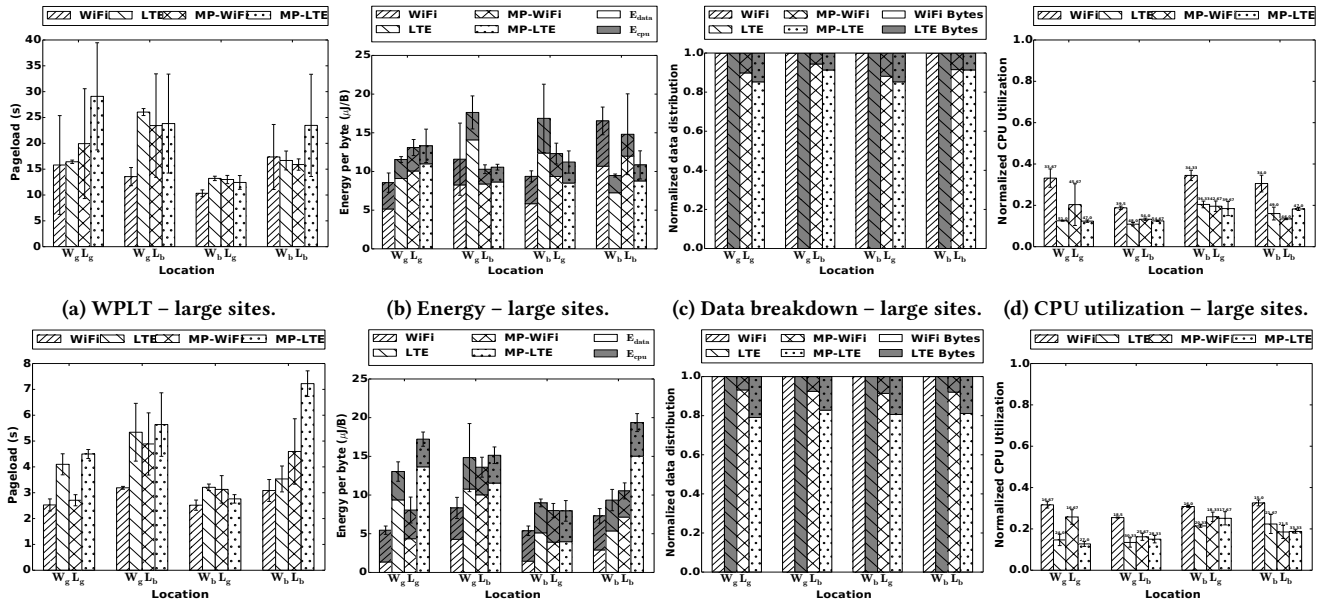
3.2.1 Spotify. Figure 4a shows that, in locations with good WiFi, where MPTCP sends again almost all the bytes over WiFi (Figure 4c), MP-WiFi performs better than (W_gL_g) or equal to (W_gL_b) the best SPTCP (WiFi) while the performance of MP-LTE is lower than that of the best SPTCP. Figure 4b shows that the energy cost with MP-WiFi is similar to that of WiFi alone in both cases while the energy cost of MP-LTE is higher due to an increase in E_{data}.

On the other hand, MPTCP improves the performance significantly compared to SPTCP in cases of poor WiFi (W_bL_g and W_bL_b). In location W_bL_g, MP-WiFi is slightly more energy efficient than MP-LTE but both MPTCP versions are slightly less energy efficient than LTE alone, due to a small increase in E_{data}. In location W_bL_b, only MP-WiFi improves performance compared to SPTCP, interestingly, with a simultaneous reduction in the energy cost due to a decrease in both E_{data} and E_{cpu}. In contrast, the performance of MP-LTE remains similar to that of best SPTCP (WiFi) albeit with large standard deviation, and the energy cost is higher compared to WiFi alone due to an increase in E_{data}.

Similar to the previous two apps, Figure 4d shows that WiFi results in higher CPU utilization except in location W_bL_g. The utilization with MPTCP is either between that of WiFi and LTE alone or equal to the highest of the two but never increases further. The contribution of CPU to the total energy consumption is again the highest with WiFi and the lowest with MP-LTE.

3.2.2 YouTube. In Figures 5a, 5b, we observe that MPTCP improves the download throughput in all 4 locations compared to SPTCP and results in energy consumption lower than (W_bL_b, due to a decrease in both E_{data} and E_{cpu}) or similar to (other three locations) the energy consumption of the most energy efficient SPTCP (e.g., WiFi in W_gL_g, LTE in W_bL_g). However, we noticed that the video quality remained the same in all the experiments (480p) regardless of the throughput. Note that, in location W_bL_b, the improvement comes only with MP-LTE (expected as LTE throughput was higher than WiFi throughput during this set of experiments), while MP-WiFi results in both lower throughput and higher energy cost (due to an increase in both E_{data} and E_{cpu}) compared to LTE alone. Note that 99% of the bytes with MP-WiFi in W_bL_b are sent over WiFi (Figure 5c), potentially due to poor interactions in the congestion control mechanism. A similar result for MP-WiFi is observed in location W_gL_b, here due to a small increase in E_{data}.

Similar to the previous apps, Figure 5d shows that WiFi results in higher CPU utilization except in location W_bL_g. MPTCP results



(a) WPLT – large sites. (b) Energy – large sites. (c) Data breakdown – large sites. (d) CPU utilization – large sites.

(e) WPLT – small sites. (f) Energy – small sites. (g) Data breakdown – small sites. (h) CPU utilization – small sites.

Figure 6: Firefox.

in higher utilization in locations W_gL_g, W_gL_b, W_bL_g and lower in W_bL_b. The contribution of CPU to the total energy consumption is again the highest with WiFi and the lowest with MP-LTE.

4 INTERACTIVE APPLICATIONS

4.1 Browsing

We divide the 12 web sites in 2 groups: large sites (qq, flickr, nytimes) and small sites (imgur, wikipedia, google, facebook, youtube, yahoo, baidu, amazon, twitter). The total number of downloaded bytes from the three large sites varies from 6-15 MB (depending on the number of embedded objects, images in flickr, etc. that change at every visit). On the other hand, the total number of downloaded bytes from the nine remaining sites is always less than 5 MB.

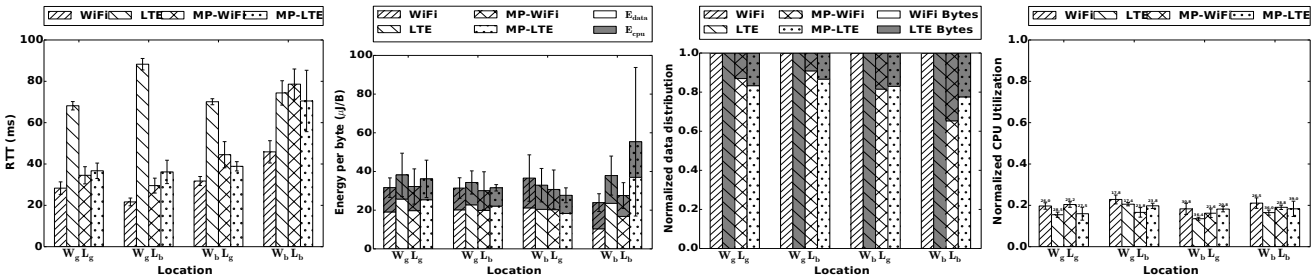
Figures 6a, 6e show that WiFi alone always results in the lowest PLT for both large and small sites and MPTCP results in increased PLT. The increase is often significant, in the range of 75-130% (large sites: W_gL_g, W_gL_b, small sites: W_gL_g, W_gL_b, and W_bL_b). The only exception is large sites at W_bL_b, where MP-WiFi yields a small decrease in the PLT compared to WiFi and LTE alone. Overall, this result is even worse than the result in [22] (1-7% improvement over the best single-path TCP PLT) and similar to the result in [15]. Additionally, Figures 6b, 6f show that MPTCP results in higher energy cost compared to the most energy efficient SPTCP with the exception of large sites in location W_gL_b (a 15% improvement over WiFi due to a decrease in E_{cpu}). Similar to the PLT, the increase in the energy can be significant and comes primarily from an increase in E_{data}. A similar result was found in [31]. Figures 6c, 6g show that in all cases, most of the bytes are delivered over WiFi with MPTCP, regardless of the primary flow. We confirmed again that WiFi exhibited significantly shorter RTTs than LTE.

In contrast to all the data intensive apps, Figures 6d, 6h show that WiFi alone results in higher CPU utilization in all four locations, even in W_bL_g. Further, MPTCP yields the lowest CPU utilization in several cases, although sometimes the selection of the primary flow matters a lot (e.g., large sites: W_gL_g, W_bL_b, small sites: W_gL_g). On the other hand, similar to the data intensive apps, the contribution of CPU to the total energy consumption is the highest with WiFi and the lowest with MP-LTE (except for small sites at W_bL_g, where LTE alone results in the lowest CPU energy consumption). Another observation from Figure 6f is that in the case of small sites, the contribution of CPU to the total energy consumption is sometimes unusually high; more than 50%.

4.2 Social networking applications

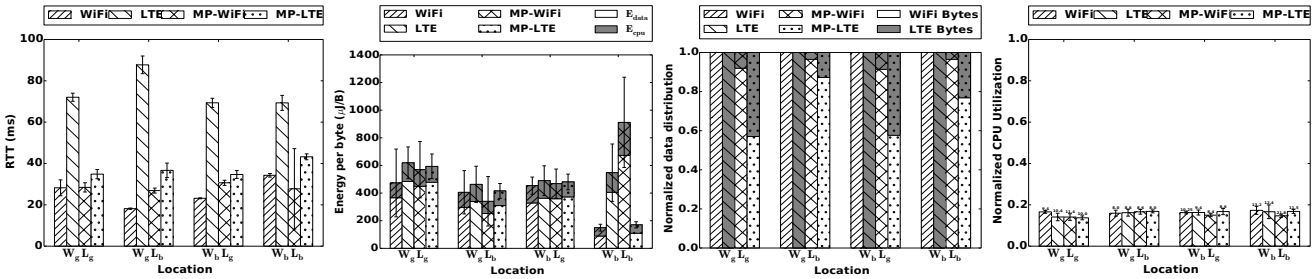
These two apps transfer the smallest amount of data among all the apps we consider, in small bursts. Facebook transfers 1.5-3 MB and Messenger only 50-160 KB.

4.2.1 Facebook. Figure 7a shows that the average RTT with WiFi alone is the lowest at all 4 locations. The average RTT with MPTCP lies between the SPTCP values in all four locations; in W_bL_b, MP-WiFi actually results in an RTT slightly higher than the worst SPTCP RTT. The lower WiFi RTTs result in most bytes being transferred over WiFi in both MP-WiFi and MP-LTE (Figure 7c). Figure 7b shows that MPTCP slightly decreases the energy consumption compared to the most energy efficient SPTCP in W_bL_g, primarily due to a decrease in E_{cpu}, but results in higher (MP-WiFi) and much higher (MP-LTE) energy cost than the most energy efficient SPTCP in W_bL_b, mainly due to an increase in E_{data}. In the remaining two locations, MP-WiFi results in lower RTT and lower energy cost than MP-LTE. Similar to web browsing, Figure 7d shows that WiFi alone results in higher CPU utilization in all four locations although the differences are very small among the four configurations. The contribution of CPU to the total energy consumption is again the highest with WiFi and the lowest with MP-LTE.



(a) RTT. (b) Energy consumption. (c) Data breakdown. (d) CPU utilization.

Figure 7: Facebook.



(a) RTT. (b) Energy consumption. (c) Data breakdown. (d) CPU utilization.

Figure 8: Messenger.

4.2.2 Messenger. Similar to Facebook, Figure 8a shows that WiFi RTTs are the lowest and RTTs with MPTCP are typically between those with SPTCP. The only exception is W_bL_b where MP-WiFi does improve the average RTT although with a large standard deviation. Again, most of the bytes are transferred over WiFi, especially in the case of MP-WiFi (Figure 8c). Figure 8b shows that MPTCP consumes more energy than WiFi alone in W_gL_g and W_bL_g due to an increase in E_{data} while the energy efficiency in the other two locations depends on the selection of the primary flow. In W_gL_b, MP-WiFi is the most energy efficient configuration, reducing both E_{data} and E_{cpu} compared to WiFi alone, while MP-LTE is as good as WiFi alone and better than LTE alone. On the other hand, in W_bL_b, MP-LTE and WiFi alone are the most energy efficient configurations but MP-WiFi results in 5x higher energy compared to WiFi alone, due to an increase in both E_{data} and E_{cpu}. Figure 8d shows that CPU utilization remains similar with all four configurations and always lower than 20%. Further, the contribution of CPU to the total energy consumption is similar with all four configurations in the three first locations. Differently from other apps, in W_bL_b the contribution of CPU is quite high with MP-LTE (38%), similar to that of WiFi alone (40%), while the contribution with MP-WiFi and LTE alone is significantly lower (23%).

5 SUMMARY OF THE RESULTS

Tables 3, 4, 5 summarize our results comparing the performance, energy, and CPU utilization of MPTCP to those of the best SPTCP in each of the 32 scenarios (a scenario is a given app tested at a given location). Column "*Best*" in Tables 3, 4 shows which of the two MPTCP versions achieves higher performance or lower energy consumption, respectively. In Table 4, we add in parentheses which of the two energy components (E_{cpu} or E_{data}) contributes to the increase or decrease of the total energy consumption. E.g., $\uparrow(E_{dt})$ means that MPTCP increases energy consumption and the increase comes from E_{data}. Column "*Lowest NU,* $\frac{E_{cpu}}{E_{na}}$" in Table 5 shows which of the two MPTCP versions achieves the lowest NU and the lowest fraction of CPU energy consumption over the total energy.

Table 3 shows that MPTCP (at least one of the two versions) results in better performance than the best SPTCP in only 13/32 scenarios. In contrast, both versions result in performance degradation in 14 scenarios, while in the remaining 5 scenarios, at least one version of MPTCP retains similar performance to the best SPTCP. Both MPTCP versions perform better than the best SPTCP in only 6 out of 13 scenarios. Among the remaining 7 scenarios, MP-WiFi improves the performance in 5 and MP-LTE in the remaining 2. The location that mostly favors MPTCP is W_bL_b (6/13 scenarios). Intuitively, MP-WiFi typically performs better than MP-LTE at locations with good WiFi (12/16 scenarios at locations W_gL_g and W_gL_b) and often when both networks exhibit poor performance (5/8 scenarios at location W_bL_b). In contrast, MP-LTE outperforms MP-WiFi typically when LTE is faster than WiFi (at location W_bL_g). As expected, MPTCP mostly benefits data intensive apps (11/16 scenarios) but it degrades the performance of interactive apps (14/16 scenarios).

The situation becomes worse when we look at the energy cost in Table 4. MPTCP (at least one version) improves the energy efficiency compared to the most energy efficient SPTCP only in 7/32 scenarios. At least one MPTCP version has similar energy cost to SPTCP in 10 scenarios while in the remaining 15 scenarios, MPTCP results in higher energy cost. Among the 7 scenarios where MPTCP improves energy efficiency, each of MP-WiFi and MP-LTE achieves

Table 3: Summary of performance results. Legend: ↑: better performance than the best SPTCP. ↓: worse performance than the best SPTCP. ≃: similar performance to the best SPTCP.

	W_gL_g			W_gL_g			W_bL_g			W_bL_b		
App	**MP-WiFi**	**MP-LTE**	**Best**	**MP-WiFi**	**MP-LTE**	**Best**	**MP-WiFi**	**MP-LTE**	**Best**	**MP-WiFi**	**MP-LTE**	**Best**
SpeedTest	↑	↑	MP-LTE	≃	↓	MP-WiFi	≃	≃	Similar	↑	↑	MP-WiFi
Dropbox	↓	↓	Similar	↓	↓	MP-WiFi	↑	↑	MP-LTE	↑	↑	MP-LTE
Spotify	↑	↓	MP-WiFi	≃	↓	MP-WiFi	↑	↑	MP-LTE	↑	≃	MP-WiFi
YouTube	↑	≃	MP-WiFi	≃	↑	MP-LTE	↑	↑	MP-LTE	↓	↑	MP-LTE
Firefox/large	↓	↓	MP-WiFi	↓	↓	Similar	↓	↓	Similar	↑	↓	MP-WiFi
Firefox/small	↓	↓	MP-WiFi	↓	↓	MP-WiFi	↓	↓	MP-LTE	↓	↓	MP-WiFi
Facebook	↓	↓	MP-WiFi	↓	↓	MP-WiFi	↓	↓	MP-LTE	↓	↓	MP-LTE
Messenger	≃	↓	MP-WiFi	↓	↓	MP-WiFi	↓	↓	MP-WiFi	↑	↓	MP-WiFi

Table 4: Summary of energy results. Legend: ↑: higher energy cost than the most energy efficient SPTCP. ↓: lower energy cost than the most energy efficient SPTCP. ≃: similar energy cost to the most energy efficient SPTCP.

	W_gL_g			W_gL_g			W_bL_g			W_bL_b		
App	**MP-WiFi**	**MP-LTE**	**Best**	**MP-WiFi**	**MP-LTE**	**Best**	**MP-WiFi**	**MP-LTE**	**Best**	**MP-WiFi**	**MP-LTE**	**Best**
SpeedTest	↓ (both)	↓ (both)	MP-LTE	≃	↑ (E_{dt})	MP-WiFi	↑ (E_{dt})	↑ (E_{dt})	MP-LTE	↓ (both)	↓ (both)	MP-WiFi
Dropbox	↑ (E_{dt})	↑ (both)	MP-WiFi	↑ (E_{dt})	↑ (both)	MP-WiFi	↑ (E_{dt})	↑ (E_{dt})	Similar	↑ (E_{dt})	↑ (E_{dt})	MP-LTE
Spotify	≃	↑ (E_{dt})	MP-WiFi	≃	↑ (E_{dt})	MP-WiFi	↑ (E_{dt})	↑ (E_{dt})	MP-WiFi	↓ (both)	↑ (E_{dt})	MP-WiFi
YouTube	≃	≃	Similar	↑ (E_{dt})	≃	MP-LTE	≃	≃	Similar	↑ (both)	↓ (both)	MP-LTE
Firefox/large	↑ (E_{dt})	↑ (E_{dt})	Similar	↓ (E_{cpu})	↓ (E_{cpu})	Similar	↑ (E_{dt})	↑ (E_{dt})	MP-LTE	↑ (E_{dt})	↑ (E_{dt})	MP-LTE
Firefox/small	↑ (E_{dt})	↑ (E_{dt})	MP-WiFi	↑ (E_{dt})	↑ (E_{dt})	MP-WiFi	↑ (E_{dt})	↑ (E_{dt})	Similar	↑ (E_{dt})	↑ (E_{dt})	MP-WiFi
Facebook	≃	↑ (E_{dt})	MP-WiFi	≃	↑ (E_{dt})	MP-WiFi	↓ (E_{cpu})	↓ (E_{cpu})	MP-LTE	↑ (E_{dt})	↑ (E_{dt})	MP-WiFi
Messenger	↑ (E_{dt})	↑ (E_{dt})	Similar	↓ (both)	≃	MP-WiFi	↑ (E_{dt})	↑ (E_{dt})	Similar	↑ (both)	≃	MP-LTE

Table 5: Summary of MPTCP impact on CPU. Legend: ↑: higher NU than the SPTCP with the lowest NU. ↓: lower NU than the SPTCP with the lowest NU. ≃: similar NU to the SPTCP with the lowest NU.

	W_gL_g			W_gL_g			W_bL_g			W_bL_b		
App	**MP-WiFi**	**MP-LTE**	**Lowest** $NU, \frac{Ecpu}{Ena}$	**MP-WiFi**	**MP-LTE**	**Lowest** $NU, \frac{Ecpu}{Ena}$	**MP-WiFi**	**MP-LTE**	**Lowest** $NU, \frac{Ecpu}{Ena}$	**MP-WiFi**	**MP-LTE**	**Lowest** $NU, \frac{Ecpu}{Ena}$
SpeedTest	↑	≃	MP-LTE,MP-LTE	↑	↑	MP-LTE,MP-LTE	↑	↑	Similar,MP-LTE	↓	↓	MP-LTE,MP-LTE
Dropbox	↑	↑	MP-WiFi,MP-WiFi	↑	↑	MP-LTE,LTE	↑	↑	MP-LTE,MP-LTE	↑	↑	MP-WiFi,MP-WiFi
Spotify	↑	↑	MP-LTE,MP-LTE	↑	↑	MP-LTE,MP-LTE	↑	↑	Similar,MP-LTE	↑	↑	MP-LTE,MP-LTE
YouTube	↑	↑	MP-LTE,MP-LTE	↑	↑	Similar,MP-LTE	↑	↑	MP-LTE,MP-LTE	↓	≃	MP-WiFi,MP-LTE
Firefox/large	↑	≃	MP-LTE,MP-LTE	↑	≃	MP-LTE,MP-LTE	↓	↓	MP-LTE,MP-LTE	↓	≃	MP-WiFi,MP-LTE
Firefox/small	↑	↓	MP-LTE,MP-LTE	↑	≃	Similar,MP-LTE	↑	↑	Similar,LTE	↓	↓	Similar,MP-LTE
Facebook	↑	≃	MP-LTE,MP-LTE	↓	≃	MP-WiFi,MP-LTE	≃	↑	MP-WiFi,MP-LTE	↑	≃	MP-LTE,MP-LTE
Messenger	≃	≃	Similar,MP-LTE	≃	≃	Similar,MP-LTE	≃	≃	Similar,MP-LTE	≃	≃	Similar,MP-WiFi

better energy savings in 3/7 scenarios while the energy consumption is similar in one scenario. Note also that MPTCP improves both performance and energy efficiency in only 4 scenarios. Out of the remaining 9 scenarios in which MPTCP improves performance, the energy cost remains the same with both MPTCP and SPTCP in 4 while a performance-energy tradeoff appears in the remaining 5. Similarly, a performance-energy tradeoff appears in the remaining 3 scenarios where MPTCP improves the energy efficiency. Similar to performance, the location that mostly favors MPTCP in terms of energy consumption is W_bL_b (3/7 scenarios). In contrast to performance, the impact on the energy cost is similar for both data intensive and interactive apps.

Table 5 shows that MPTCP in general results in higher CPU utilization compared to SPTCP. MPTCP reduces NU only in 7/32 scenarios; 5 involving interactive apps and 2 involving data intensive apps. The location that favors MPTCP is again W_bL_b. However, there is no strong correlation between NU and performance or energy efficiency. Out of the 7 scenarios where MPTCP decreases NU, performance improves only in 2, energy efficiency in 1, and both metrics in 1. Note that higher NU typically does not translate in higher CPU energy consumption; in particular for MP-LTE, the fraction of the CPU energy consumption over the total energy is the lowest among the four configurations in most scenarios. As Table 4 shows, the increased total energy cost in most scenarios mainly comes from an increase in E_{data} due to the combined use of both NICs.

6 RELATED WORK

Performance. Initial experimental studies [9, 9, 15, 15, 16, 25, 28, 30, 32] used mostly laptops and focused on file transfers. The most relevant to our work are [9, 15], which measure performance over cellular and WiFi networks and show that, for short flows, MPTCP performs worse than SPTCP over the fastest path. The latter also examines the performance of two real applications (Dropbox and web browsing) over MPTCP using emulation and trace replay. Similarly, Ferlin et al. [16] show that multi-path transfer might actually have a negative impact over heterogeneous wireless networks. Different from these works, Qian et al. [28] study the impact of MPTCP on a single application – web browsing.

More recent studies examine the impact of MPTCP on smartphone apps. De Coninck et al. [11, 13] study eight popular Android apps focusing on their usage of WiFi and cellular networks. The same authors study a crowd-sourced dataset from 12 real smartphone users over a 7-week period [12], focusing on transport layer characteristics – subflow RTTS and utilization, retransmission and reinjection, and handovers. In our work, we use their automated test framework from [11, 13] but we investigate energy-performance tradeoffs for various applications. Nikravesh et al. [22] analyze a larger dataset collected from 15 students over a 4-month period that includes both passive and active measurements, and complement this trace with controlled experiments. They also confirm that MPTCP performs worse than SPTCP for short flows or when one of the two networks is substantially faster than the other one.

Energy consumption. In contrast to performance, MPTCP energy consumption has received relatively less attention. Initial works [26, 27, 29] relied only on simulations and often use outdated measurement studies to obtain the parameters for their energy and throughput models (e.g., [7, 18, 19]) which do not reflect the WiFi and LTE speeds available in modern smartphones.

The majority of experimental studies examine energy consumption in the case of file downloads [14, 15, 21, 23, 31]. Deng et al. only study the energy consumption of the backup mode. Paasch et al. [23] and Croitoru et al. [14] evaluate the energy consumption of cellular/WiFi handover with MPTCP. Nika et al. [21] conduct the first experimental study comparing the energy and performance of radio bundling via MPTCP vs. an ideal protocol. Their results show that MPTCP achieves only a fraction of the total performance gain possible and its total energy consumption is similar to LTE-only and up to 3.5 times higher compared to WiFi-only. Lim et al. develop a model of the energy consumption of MPTCP as a function of the WiFi and LTE throughputs and use it to design an energy-aware variant of MPTCP [31], which tries to reduce the energy consumption with minimal impact to download latency. The only work that examines the energy consumption of specific apps is [22]. However, most of their findings with respect to energy consumption come from trace-based simulations and modeling, which admittedly result in a coarse-grained estimation, and their focus is on comparing the energy contribution of each of the two interfaces. In contrast, our focus is on analyzing the energy consumption of MPTCP (vs. SPTCP over WiFi/LTE) via direct measurements.

7 CONCLUSION

This work, to our best knowledge, is the first that seeks an answer to the question "When is MPTCP beneficial in smartphones?" by jointly considering the impact on performance, energy consumption, and CPU utilization. We answer this question via an extensive experimental study, considering different types of real Android applications and a variety of network conditions. Our results reveal that the benefits of MPTCP in smartphone apps in practice are quite limited; in several cases, MPTCP in fact can hurt both performance and energy consumption, especially in the case of interactive apps or under heterogeneous network conditions. We believe that our findings provide valuable insights to smartphone designers and mobile app developers towards improving user experience and extending smartphone battery life.

8 ACKNOWLEDGMENTS

This work was supported in part by NSF grant CNS-1422304.

REFERENCES

[1] Commercial usage of Multipath TCP. http://blog.multipath-tcp.org/blog/html/2015/12/25/commercial_usage_of_multipath_tcp.html.
[2] Debugging Firefox for Android over USB. https://developer.mozilla.org/en-US/docs/Tools/Remote_Debugging/Debugging_Firefox_for_Android_with_WebIDE.
[3] Measuring Page Load Speed with Navigation Timing. https://www.html5rocks.com/en/tutorials/webperformance/basics/.
[4] Monsoon Power Monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.
[5] ShadowSocks: A secure socks5 proxy, designed to protect your Internet traffic. https://shadowsocks.org/.
[6] Use Multipath TCP to create backup connections for iOS. https://support.apple.com/en-us/HT201373.
[7] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proc. of ACM/USENIX Internet Measurement Conference (IMC)*.
[8] Xiaomeng Chen, Ning Ding, Abhilash Jindal, and Rath Vannithamby. 2015. Smartphone Energy Drain in the Wild: Analysis and Implications. In *Proc. of ACM SIGMETRICS*.
[9] Yung-Chih Chen, Yeon sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. 2013. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *Proc. of ACM Internet Measurement Conference (IMC)*.
[10] Quentin De Coninck and Matthieu Baerts. 2015. MPTCP Analysis Scripts. (2015). http://github.com/multipath-tcp/mptcp-analysis-scripts.
[11] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2015. Poster: Evaluating Android Applications with Multipath TCP. In *Proc. of ACM MobiCom*.
[12] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2016. A First Analysis of Multipath TCP on Smartphones. In *Proc. of Passive and Active Measurement Conference (PAM)*.
[13] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2016. Observing Real Smartphone Applications over Multipath TCP. *IEEE Communications Magazine, Network Testing Series*, 54, 3 (March 2016), 88–93.
[14] Andrei Croitoru, DragoÅ§ Niculescu, and Costin Raiciu. 2015. Towards Wifi Mobility without Fast Handover. In *Proc. of USENIX NSDI*.
[15] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. 2014. WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance. In *Proc. of ACM Internet Measurement Conference (IMC)*.
[16] Simone Ferlin, Thomas Dreibholz, and ÃŰzgu Alay. 2014. Multi-Path Transport Over Heterogeneous Wireless Networks: Does It Really Pay Off?. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*.
[17] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. 2013. TCP Extensions for Multipath Operation with Multiple Addresses. (2013). RFC 1546.
[18] Daniel Halperin, Ben Greenstein, Anmol Sheth, and David Wetherall. 2010. Demystifying 802.11n power consumption. In *Proc. of USENIX Workshop on Power Aware Computing and Systems*.
[19] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2012. A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In *Proc. of ACM Mobisys*.
[20] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. 1996. SOCKS Protocol Version 5. (1996). RFC 1928.
[21] Ana Nika, Yibo Zhu, Ning Ding, Abhilash Jindal, Y. Charlie Hu, Xia Zhou, Ben Y. Zhao, and Haitao Zheng. 2015. Energy and Performance of Smartphone Radio Bundling in Outdoor Environments. In *Proc. of the 24th International World Wide Web Conference (WWW 2015)*.
[22] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z. Morley Mao, and Subhabrata Sen. 2016. An In-depth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design. In *Proc. of ACM MobiCom*.
[23] Christoph Paasch, Gregory Detal, Fabien Duchene, Costin Raiciu, and Olivier Bonaventure. 2012. Exploring Mobile/WiFi Handover with Multipath TCP. In *Proc. of ACM CellNet*.
[24] Christoph Paasch, Simone Ferlin, ÃŰzgu Alay, and Olivier Bonaventure. 2014. Experimental evaluation of multipath TCP schedulers. In *Proc. of ACM CSWS*.
[25] Christoph Paasch, Ramin Khalili, and Olivier Bonaventure. 2013. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *Proc. of ACM CoNEXT*.
[26] Qiuyu Peng, Minghua Chen, Anwar Walid, and Steven H. Low. 2014. Energy Efficient Multipath TCP for Mobile Devices. In *Proc. of ACM MobiHoc*.
[27] Christopher Pluntke, Lars Eggert, and Niko Kiukkonen. 2011. Saving mobile device energy with multipath TCP. In *Proc. of ACM MobiArch*.
[28] Feng Qian, Bo Han, Shuai Hao, and Lusheng Ji. 2015. An Anatomy of Mobile Web Performance over Multipath TCP. In *Proc. of ACM CoNEXT*.
[29] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley. 2011. Opportunistic Mobility with Multipath TCP. In *Proc. of ACM MobiArch*.
[30] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proc. of USENIX NSDI*.
[31] Yeon sup Lim, Yung-Chih Chen, Erich M. Nahum, Don Towsley, Richard J. Gibbens, and Emmanuel Cecchet. 2015. Design, Implementation and Evaluation of Energy-Aware Multi-Path TCP. In *Proc. of ACM CoNEXT*.
[32] Yeon sup Lim, Yung-Chih Chen, Erich M. Nahum, Donald F. Towsley, and Kang-Won Lee. 2014. Cross-layer path management in multi-path transport protocol for mobile devices. In *Proc. of IEEE INFOCOM*.
[33] Yifan Zhang, Xudong Wang, Xuanzhe Liu, Yunxin Liu, Li Zhuang, and Feng Zhao. 2013. Towards better CPU power management on multicore smartphones. In *Proc. of HotPower*.