WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance

Shuo Deng, Ravi Netravali, Anirudh Sivaraman, Hari Balakrishnan
MIT Computer Science and Artificial Intelligence Lab
Cambridge, Massachusetts, U.S.A.
{shuodeng, ravinet, anirudh, hari}@csail.mit.edu

ABSTRACT

Over the past two or three years, wireless cellular networks have become faster than before, most notably due to the deployment of LTE, HSPA+, and other similar networks. LTE throughputs can reach many megabits per second and can even rival WiFi throughputs in some locations. This paper addresses a fundamental question confronting transport and application-layer protocol designers: which network should an application use? WiFi, LTE, or Multi-Path TCP (MPTCP) running over both?

We compare LTE and WiFi for transfers of different sizes along both directions (i.e. the uplink and the downlink) using a crowd-sourced mobile application run by 750 users over 180 days in 16 different countries. We find that LTE outperforms WiFi 40% of the time, which is a higher fraction than one might expect at first sight.

We measure flow-level MPTCP performance and compare it with the performance of TCP running over exclusively WiFi or LTE in 20 different locations across 7 cities in the United States. For short flows, we find that MPTCP performs worse than regular TCP running over the faster link; further, selecting the correct network for the primary subflow in MPTCP is critical in achieving good performance. For long flows, however, selecting the proper MPTCP congestion control algorithm is equally important.

To complement our flow-level analysis, we analyze the traffic patterns of several mobile apps, finding that apps can be categorized as "short-flow dominated" or "long-flow dominated". We then record and replay these patterns over emulated WiFi and LTE links. We find that application performance has a similar dependence on the choice of networks as flow-level performance: an application dominated by short flows sees little gain from MPTCP, while an application with longer flows can benefit much more from MPTCP — if the application can pick the right network for the primary subflow and the right choice of MPTCP congestion control.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; C.4 [**Performance of System**]: Measurement techniques, Performance attributes

IMC'14, November 5–7, 2014, Vancouver, BC, Canada.
Copyright 2014 ACM 978-1-4503-3213-2/14/11$15.00.
http://dx.doi.org/10.1145/2663716.2663727.

KEYWORDS

Multi-Network, Mobile Device, LTE, Multi-Path TCP

1. INTRODUCTION

Access to WiFi and cellular wireless networks are *de rigueur* on mobile devices today. With the emergence of LTE, cellular performance is starting to rival the performance of WiFi. Moreover, when WiFi signal quality is low or in crowded settings, the anecdotal experience of many users is that cellular performance may in fact be considerably better than WiFi performance.

But just how good are LTE and WiFi networks in practice and how do they compare with each other? Should applications and transport protocols strive to select the best network, or should they simply always use Multi-Path TCP (MPTCP) [21]? This paper seeks to answer these questions empirically.

To answer these questions, we implemented a crowd-sourced network measurement tool (Section 2) to understand the flow-level performance of TCP over WiFi and LTE in the wild from 16 different countries over a 6-month period, encompassing 3624 distinct 1-MegaByte TCP flows. We used this data to measure transfer times for different amounts of data transferred.

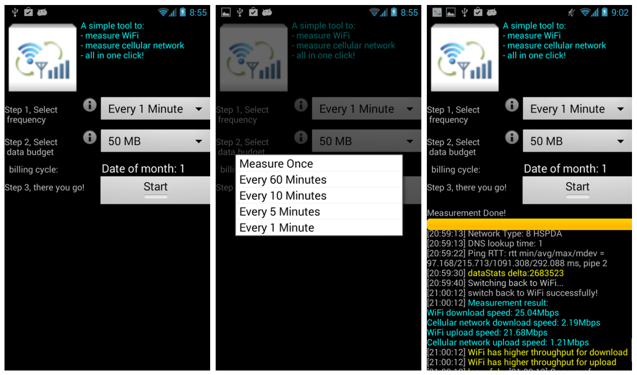
MPTCP isn't widely deployed yet on most phones[1]. As a result, we manually measured flow-level MPTCP performance and compared it with the performance of TCP running over exclusively WiFi or LTE in 20 different locations, in 7 cities in the United States (Section 3). Finally, to complement our empirical flow-level analysis, we used an existing record-and-replay tool to analyze (Section 4) and run (Section 5) mobile apps on emulated cellular and WiFi links, using it to study the impact of network selection on application performance.

Our key findings are as follows:

1. Cellular networks outperform WiFi around 40% of the time in our data set (Figure 3), a proportion considerably higher than we had hypothesized.
2. For short flows (100 KB or lower), MPTCP performs worse than TCP (Figure 7b). Further, it is crucial to select the proper network for the primary MPTCP subflow[2]. For instance, on a 10 KB flow, we found that the choice of the network for the primary subflow can affect MPTCP throughput by upto 60% (Figure 8). For long flows, selecting the proper congestion control algorithm is also important: for a 1 MB flow, for instance, modifying only the congestion control algorithm, while keeping the network used by the primary subflow fixed changes MPTCP throughput by 34% (Figures 13 and 14). On the other hand, modifying only the network used by the pri-

[1]The Apple iOS is an exception [13].

[2]For a description of subflow and other MPTCP-related terms, we refer the reader to Section 3.1



```
[20:59:13] Network Type: 8 HSPDA
[20:59:13] DNS lookup time: 1
[20:59:22] Ping RTT: rtt min/avg/max/mdev =
97.168/215.713/1091.308/292.088 ms, pipe 2
[20:59:30] dataStats delta:2683523
[20:59:40] Switching back to WiFi...
[21:00:12] switch back to WiFi successfully!
[21:00:12] Measurement result:
WiFi download speed: 25.04Mbps
Cellular network download speed: 2.19Mbps
WiFi upload speed: 21.68Mbps
Cellular network upload speed: 1.21Mbps
[21:00:12] WiFi has higher throughput for download
[21:00:12] WiFi has higher throughput for upload
```

Figure 1: *Cell vs WiFi* User Interface.

mary subflow, while keeping the congestion-control algorithm fixed, changes throughput by 25%. (Figure 14)

3. Mobile app traffic patterns largely fall into two groups (Figure 17). We refer to apps that tend to open several connections, each transferring small amounts of data, as *short-flow dominated* apps, and we refer to apps that have fewer number of connections but transfer large amounts of data on each as *long-flow dominated* apps.
4. For *short-flow dominated* apps, MPTCP does not outperform the best conventional "single-path" TCP (over either Wi-Fi or LTE) (Figures 18 and 19). However, it is important to choose the correct network for standard TCP. Our emulation shows that selecting the proper network for single-path TCP can reduce response time by 50% compared to the minimum of the single-path TCP throughputs on LTE and WiFi. On the other hand, using MPTCP reduces application response time by only 35%.
5. For *long-flow dominated* apps, MPTCP does help markedly, provided the appropriate congestion-control algorithm is used and the two links have roughly comparable speeds: our emulation shows that using single path TCP with the correct choice of network reduces application response time by 42%, while using MPTCP with the proper congestion control can also reduce response time by about 50% (Figures 20 and 21).

Our crowd-sourced network measurement tool, *Cell vs WiFi*, is available for Android in the Google Play Store. All our measurement data and analysis tools are available at `http://web.mit.edu/cell-vs-wifi/`.

2. CELL VS WIFI MEASUREMENT

In September 2013, we published an Android app on Google Play, called *Cell vs WiFi* (`http://web.mit.edu/cell-vs-wifi`). *Cell vs WiFi* measures end-to-end WiFi and cellular network performance and uses these measurements to tell smartphone users if they should be using the cellular network or WiFi at the current time and location. The app also serves as a crowd-sourced measurement tool by uploading detailed measurement data to our server including packet-level traces. Over a nine-month period since the app was published, it attracted over 750 downloads. We collected over 10 GB of measurement data from 3632 distinct TCP connections over this duration from these users.

2.1 Cell vs WiFi App

Figure 1 shows the user interface of *Cell vs WiFi*. Users can choose to measure network performance periodically, or once per click. Users can also set an upper bound on the amount of cellular

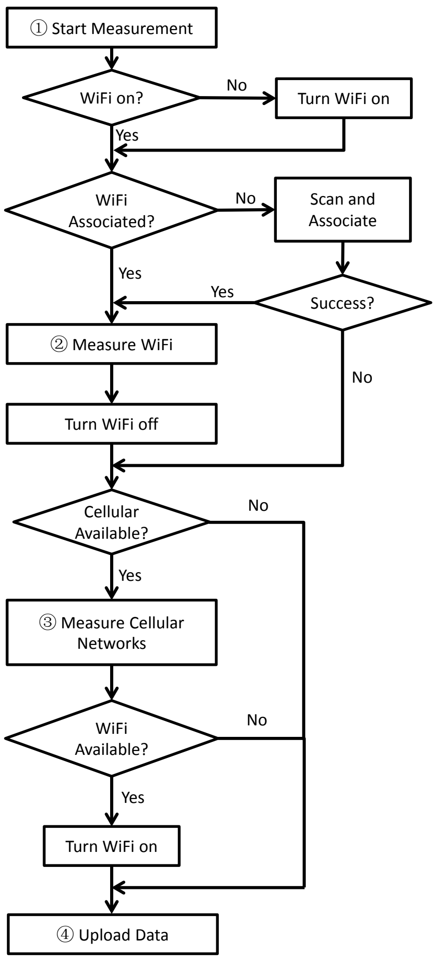


Figure 2: *Cell vs WiFi*: single measurement collection run.

data that the app can consume; especially for devices on a limited cellular data plan.

The flow chart in Figure 2 shows a single measurement-collection run. When the user clicks the `Start` button or the pre-set periodic measurement timer expires, one run of measurement collection starts, shown as Step ① in the figure. If WiFi is available and the phone successfully associates with an Access Point (AP), *Cell vs WiFi* collects packet-level tcpdump traces for a 1 Mbyte TCP upload and a 1 Mbyte TCP download between the mobile device and our server at MIT.

After measuring WiFi, *Cell vs WiFi* turns off the WiFi interface on the phone and attempts to automatically connect to the cellular network. If the user has turned off the cellular data network, *Cell vs WiFi* aborts the cellular measurement. If *Cell vs WiFi* successfully connects to the cellular network, then in Step ③, it collects a similar set of packet-level tcpdump traces for both an upload and a download. Once both WiFi and cellular network measurements are finished, in Step ④, *Cell vs WiFi* uploads the data collected during this measurement run, together with the user ID (randomly generated when a smartphone user uses the app for the first time), and the phone's geographic location, to our server at MIT.

More information about *Cell vs WiFi* can be found at `http://web.mit.edu/cell-vs-wifi`.

2.2 Results

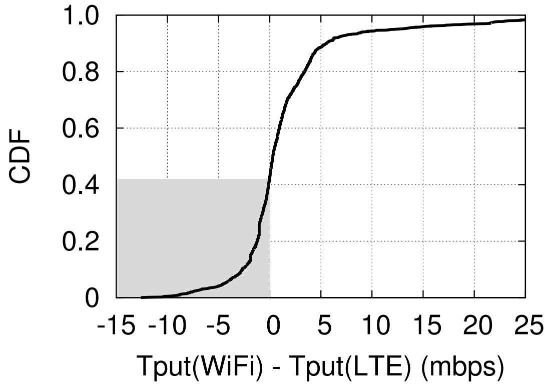
Location Name	(Lat, Long)	# of Runs	LTE %
US (Boston, MA)	(42.4, -71.1)	884	10%
Israel	(31.8, 35.0)	276	55%
US (Portland)	(45.6, -122.7)	164	45%
Estonia	(59.4, 27.4)	124	71%
South Korea	(37.5, 126.9)	108	66%
US (Orlando)	(28.4, -81.4)	92	35%
US (Miami)	(26.0, -80.2)	84	52%
Malaysia	(4.24, 103.4)	76	68%
Brazil	(-23.6, -46.8)	56	4%
Germany	(52.5, 13.3)	40	20%
Spain	(28.0, -16.7)	40	80%
Thailand (Phichit)	(16.1, 100.2)	40	80%
US (New York)	(40.9, -73.8)	24	33%
Japan	(36.4, 139.3)	16	25%
Sweden	(59.6, 18.6)	16	0%
Thailand (Chiang Mai)	(18.8, 99.0)	16	75%
US (Chicago)	(42.0, -88.2)	16	25%
Hungary	(47.4, 16.8)	8	0%
Italy	(44.2, 8.3)	8	0%
US (Salt Lake City)	(40.8, -111.9)	8	0%
Colombia	(7.1, -70.7)	4	0%
US (Santa Fe)	(35.9, -106.3)	4	0%

Table 1: Geographical coverage and diversity of the crowd-sourced data collected from 16 countries using *Cell vs WiFi*, ordered by number of runs collected. The last column shows the percentage of runs where LTE throughput is higher than WiFi

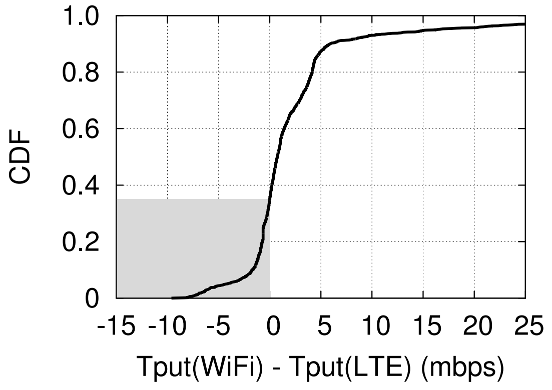
Cell vs WiFi collected network-performance data from locations in five continents: North America, South America, Europe, Africa, and Asia. We observed that some users use this app to measure only WiFi or LTE performance, but not both. We do not consider these measurement runs in this section because our goal is to compare LTE and WiFi performance at nearly the same place and time. To ensure that we only measure performance of LTE or an equivalent high-speed cellular network, such as HSPA+, we use the Android network-type API [2] and pick only those measurement runs that used LTE or HSPA+. When using the term LTE in this section, we mean LTE/HSPA+. After these filtering steps, our dataset contains over 1606 complete runs of measurement, i.e., both LTE and WiFi transfers in both directions.

In Table 1, we group nearby runs together using a k-means clustering algorithm, with a cluster radius of $r = 100$ kilometers; i.e., all runs in each group are within 200 kilometers of each other. For each location group, we also list the percentage of measurement runs where LTE has higher throughput than WiFi.

Figure 3 shows the CDF of difference in throughput between WiFi and LTE on the uplink and the downlink. We can see that the throughput difference can be larger than 10 Mbits/s in either direction. The grey region shows 42% (uplink) and 35% (downlink) of the data samples whose LTE throughput is higher than WiFi throughput. If we combine uplink and downlink together, 40% of the time LTE outperforms WiFi. Figure 4 shows the CDF of ping RTT difference between LTE and WiFi. During our measurement, we send 10 pings and take the average RTT value. The shaded area shows that in 20% of our measurement runs, LTE has a lower ping RTT than WiFi, although the cellular network is commonly assumed to have higher delays.



(a) Uplink



(b) Downlink

Figure 3: CDF of difference between WiFi and LTE throughput. The grey region shows 42% (uplink) and 35% (downlink) of the data samples whose LTE throughput is higher than WiFi throughput.

The simple network selection policy used by mobile devices today forces applications to use WiFi whenever available. However, our measurement results indicate that a more flexible network selection policy will improve the network performance of mobile applications.

3. MPTCP MEASUREMENTS

When WiFi and cellular networks offer comparable performance, or when each varies significantly with time, it is natural to use both simultaneously. Several schemes transmitting data on multiple network interfaces have been proposed in the past [22, 17, 15, 21]. Among these, the most widespread is MPTCP [21]. MPTCP can be used in two modes [16]: *Full-MPTCP* mode, which transmits data on all available network interfaces at any time and *Backup* mode, which transmits data on only one network interface at a time, falling back to the other interface only if the first interface is down. Unless stated otherwise, all experiments in this section use MPTCP in *Full-MPTCP* mode. For completeness, we compare the two modes in Section 3.6. We use a modified version of *Cell vs WiFi* to carry out MPTCP measurements. We observe the following:

1. We find that MPTCP throughput for short flows depends significantly on the network selected for the primary subflow[3] in MPTCP: for example, changing the network (LTE or WiFi)

[3]We define subflows in Section 3.1

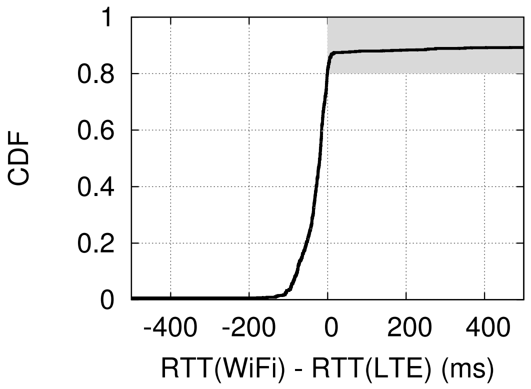


Figure 4: CDF of the difference between average Ping RTT with WiFi and LTE. The grey region shows 20% of the data samples whose LTE RTT is lower than WiFi RTT.

for the primary subflow changes the average throughput of a 10 KByte flow by 60% in the median (Figure 8 in Section 3.4).

2. For long flows, selecting the proper congestion control algorithm is also important. For example, using different congestion-control algorithms (coupled or decoupled) changes the average throughput of a 1 MByte flow by 34% in the median (Figure 13 in Section 3.5).
3. MPTCP's backup mode is typically used for energy efficiency: keeping fewer interfaces active reduces energy consumption overall. However, we find that for MPTCP in *Backup Mode*, if LTE is set to the backup interface, very little energy can be saved for flows that last shorter than 15 seconds (Section 3.6).

3.1 MPTCP Overview

MPTCP initiates a connection in a manner similar to regular TCP: it picks one of the available interfaces and establishes a TCP connection using a SYN-ACK exchange with the server over that interface. Every TCP connection that belongs to a MPTCP connection is called an MPTCP *subflow*. The first established subflow is called the *primary subflow*.

We used the Linux MPTCP implementation for our measurements [14] (Ubuntu Linux 13.10 with Kernel version 3.11.0, with the MPTCP Kernel implementation version v0.88). In this implementation, MPTCP initiates the primary subflow on the interface used as the default route on the machine. Once the primary subflow is established, if there are other interfaces available, MPTCP creates an additional subflow using each new interface, and combines the new subflow with the existing subflows on the same MPTCP connection.[4] For example, a mobile device can establish an MPTCP primary subflow through WiFi to the server, and then add an LTE subflow to the server. To terminate the connection, each subflow is terminated using four-way FIN-ACKs, similar to TCP. In Section 3.4, we study the effect of choosing different interfaces for the primary subflow on MPTCP performance.

There are two kinds of congestion-control algorithms used by MPTCP: *decoupled* and *coupled*. In decoupled congestion control, each subflow increases and decreases its congestion window independently, as if they were independent TCP flows [5]. In coupled congestion control, each subflow in an MPTCP connection increases its congestion window based on ACKs both from itself and from other subflows [21, 10] in the same MPTCP connection. In Section 3.5, we compare the coupled and decoupled algorithms and find that using different congestion-control algorithms has less impact on throughput compared with selecting the correct interface for primary subflows for short flows. However, for long flows, changing congestion-control algorithms results in a substantial throughput difference.

[4]For simplicity, here we only explain how MPTCP works when the server is single-homed (like the server in our experiments), and the client alone is multi-homed.

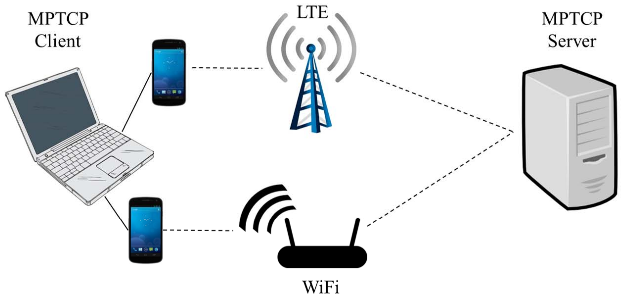


Figure 5: Setup of MPTCP measurement.

3.2 Measurement Setup

Figure 5 shows the MPTCP measurement setup. The MPTCP Client is a laptop running Ubuntu 13.10 with MPTCP installed. We tethered two smartphones to the laptop, one in "airplane" mode with WiFi enabled, and the other with WiFi disabled but connected to LTE (either the Verizon or the Sprint LTE network). The MPTCP server is located at MIT, with a single Ethernet interface, also running Ubuntu 13.10 with MPTCP installed.

We installed a modified version of *Cell vs WiFi* on both phones. The phone with WiFi enabled only measures WiFi performance, i.e., Step ② in Figure 2. The phone connected to LTE only measures cellular network performance, i.e., Step ③ in Figure 2.

The experimental setup also allows us to measure the energy consumption separately for each interface, which we present in Section 3.6.

Each measurement run comprises the following:

1. Single path TCP upload and download using modified *Cell vs WiFi* through LTE.
2. Single path TCP upload and download using modified *Cell vs WiFi* through WiFi.
3. MPTCP upload and download in *Full-MPTCP* mode with LTE as the primary subflow.
4. MPTCP upload and download in *Full-MPTCP* mode with WiFi as the primary subflow.

We conducted the measurements at 20 different locations on the east and west coasts of the United States, shown in Table 2. At each city, we conduct our measurement at places where people would often use mobile devices: cafes, shopping malls, university campuses, hotel lobbies, airports, and apartments. At 7 of the 20 locations, we measured both Verizon and Sprint LTE networks, using both MPTCP congestion-control algorithms: decoupled and coupled. At the other 13 locations, we were able to measure only the Verizon LTE network with coupled congestion control.

In Figure 6, we compare the WiFi and LTE throughput distributions for the data we collected at these 20 locations and the data collected from *Cell vs WiFi* in Section 2. We can see that for both upload and download, the "20-Location" CDF curves are close to the CDF curve from Section 2, implying that the 20 locations that were selected have similar variability in network conditions as the

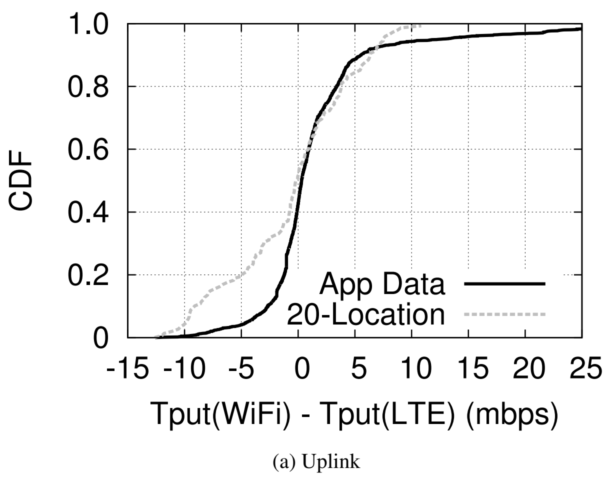


Figure 6: CDF for WiFi and LTE throughput measured using regular TCP at 20 locations (shown as "20-Location") comparing with the CDF in Figure 3(shown as "App Data").

ID	City	Description
1	Amherst, MA	University Campus, Indoor
2	Amherst, MA	University Campus, Outdoor
3	Amherst, MA	Cafe, Indoor
4	Amherst, MA	Downtown, Outdoor
5	Amherst, MA	Apartment, Indoor
6	Boston, MA	Cafe, Indoor
7	Boston, MA	Shopping Mall, Indoor
8	Boston, MA	Subway, Outdoor
9	Boston, MA	Airport, Indoor
10	Boston, MA	Apartment, Indoor
11	Boston, MA	Cafe, Indoor
12	Boston, MA	Downtown, Outdoor
13	Boston, MA	Store, Indoor
14	Santa Barbara, CA	Hotel Lobby, Indoor
15	Santa Barbara, CA	Hotel Room, Indoor
16	Santa Barbara, CA	Conference Room, Indoor
17	Los Angeles, CA	Airport, Indoor
18	Washington, D.C.	Hotel Room, Indoor
19	Princeton, NJ	Hotel Room, Indoor
20	Philadelphia, PA	Hotel Room, Indoor

Table 2: Locations where MPTCP measurements were conducted

Cell vs WiFi dataset. For simplicity, in the rest of Section 3, we only show results of downlink flows from the server to the client.

3.3 TCP vs. MPTCP

A natural question pertaining to MPTCP is how the performance of MPTCP compares with the best "single-path" TCP performance achievable by an appropriate choice of networks alone. To answer this, we look at all four MPTCP variants (two congestion-control algorithms times two choices for the network used by the primary subflow) and both single-path TCP variants (WiFi and LTE) as a function of flow size. Figure 7 illustrates two qualitatively different behaviors.

Figure 7a shows a case where the performance of MPTCP is always worse than the best single-path TCP regardless of flow size.

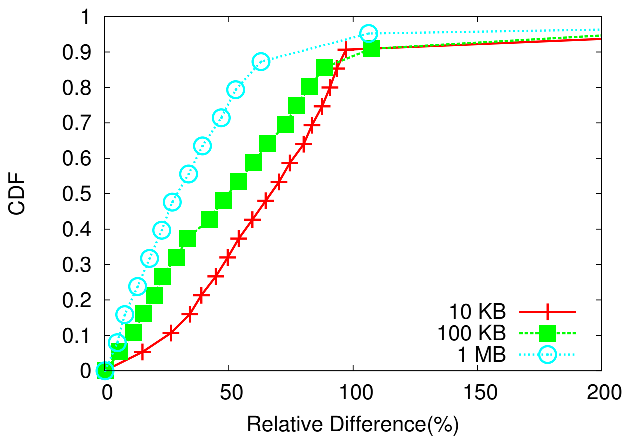
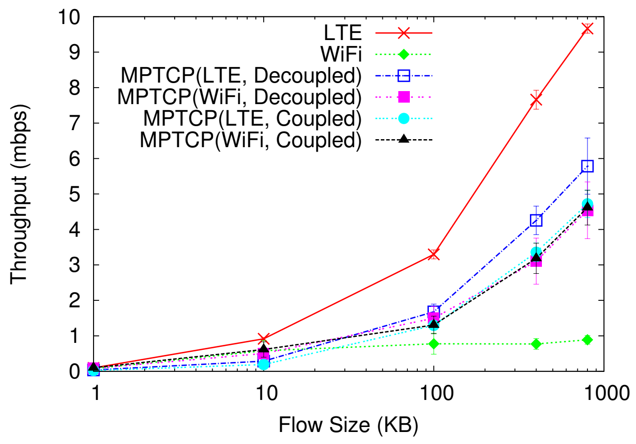


Figure 8: CDF of relative difference between $MPTCP_{LTE}$ and $MPTCP_{WiFi}$, for different flow sizes. The median relative difference for each flow size: 60% for 10 KBytes, 49% for 100 KBytes and 28% for 1MByte. Thus, throughput for smaller flow sizes is more affected by the choice of the network for the primary subflow.

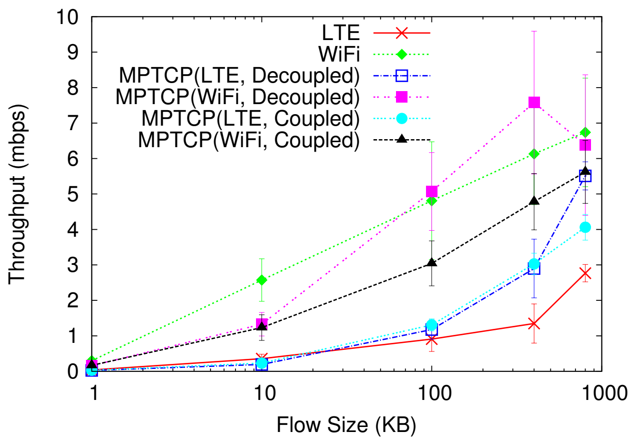
This occurs in a particularly extreme scenario where a large disparity in link speeds between the two networks (LTE and WiFi) leads to degraded MPTCP performance irrespective of flow size. On the other hand, Figure 7b shows an alternative scenario where MPTCP is better than the best single-path TCP at larger flow sizes. In both cases, however, picking the right network for single-path TCP is preferable to using MPTCP for smaller flows. These results suggest that it may not always be advisable to use both networks, and an adaptive policy that automatically picks the networks to transmit on and the transport protocol to use would improve performance relative to any static policy.

3.4 Primary Flow Measurement

We then study how the choice of the network for the primary subflow can affect MPTCP throughput for different flow sizes. To show this quantitatively, we calculate the relative throughput difference as:



(a) MPTCP performs worse than single TCP.



(b) MPTCP performs better than single TCP.

Figure 7: MPTCP throughput vs single path TCP throughput at 2 representative locations. Figure 7a shows a case where MPTCP throughput is lower than the best throughput of single path TCP. Figure 7b shows a case where MPTCP throughput (in this case, MPTCP(WiFi, decoupled)) is higher than that of single-path TCP for large flow sizes.

$$\frac{|MPTCP_{LTE} - MPTCP_{WiFi}|}{MPTCP_{WiFi}}.$$

Here, $MPTCP_{LTE}$ is the throughput achieved by MPTCP using LTE for the primary subflow, and $MPTCP_{WiFi}$ is the throughput achieved by MPTCP using WiFi for the primary subflow (in this subsection, we only run MPTCP using *decoupled* congestion control). Figure 8 shows the CDF of the relative throughput difference for three flow sizes: 10 KBytes, 100 KBytes, and 1 MBytes. We see that using different networks for the primary subflow has the greatest effect on smaller flow sizes.

3.4.1 MPTCP Throughput Evolution Over Time

To understand how using different networks for the primary subflow affects MPTCP throughput evolution over time, we collected `tcpdump` traces at the MPTCP Client during the measurement. From the traces, we calculate the average throughput from the time the MPTCP session is established, to the current time t. In Figure 10 and 9, we plot the average throughput as a function of time. Each sub figure shows the throughput of the entire MPTCP session (shown as MPTCP) and the throughput of the individual WiFi and LTE subflows.

Figure 9 shows traces collected at a location where LTE has much higher throughput than WiFi. At time 0, the client sent the SYN packet to the server. In Figure 9a, it took the client 1 second to receive the SYN-ACK packet from the server over WiFi. MPTCP throughput is the same as the throughput of the WiFi subflow until the LTE subflow is established. Because LTE has much higher throughput at this location and time, once the subflow is established on LTE, it quickly increases its throughput (and therefore MPTCP's throughput). By contrast, in Figure 9b, the client receives the SYN-ACK faster, and MPTCP throughput increases more quickly because the first subflow is on the higher-throughput LTE network. Because of the smaller SYN-ACK RTT and higher throughput on the first primary subflow, the MPTCP connection using LTE for the primary subflow (Figure 9b) has a higher average throughput than the MPTCP connection using WiFi for the primary subflow (Figure 9a).

Similarly, Figure 10 shows traces collected at a location where WiFi has higher throughput than LTE. Here, using WiFi as the primary subflow for MPTCP results in higher throughput.

3.4.2 MPTCP Throughput as a Function of Flow Size

Figures 11a and 12a show how MPTCP throughput changes as the flow size increases. The flow size is measured using the cumulative number of bytes acknowledged in each ACK packet received at the MPTCP client. Figures 11b and 12b show the relative throughput ratio change as flow size increases. The relative throughput ratio is defined as:

$$\frac{MPTCP_{LTE}}{MPTCP_{WiFi}}.$$

Although the absolute value of the difference in throughputs is smaller for small flow sizes (Figures 11a and 12a), the relative throughput ratio, is larger (Figure 11b and 12b). Thus, for a connection with a given flow size, using the correct interface for MPTCP primary subflow can reduce the flow completion time, and the relative reduction can be significant for smaller flow sizes. For example, in Figure 11a, the absolute throughput difference between LTE and WiFi is 0.5 mbps for a 100 KB flow, and about 3 mbps for a 1 MB flow. But in Figure 11b, the relative throughput ratio is 2.2x for 100 KB flow, larger than the 1.5x for a 1 MB flow.

3.5 Coupled and Decoupled Congestion Control

To understand how the choice of congestion-control algorithm within MPTCP affects its throughput, at 7 of the 20 locations, we measured the following different MPTCP configurations on the Verizon LTE network and each location's dominant WiFi network:

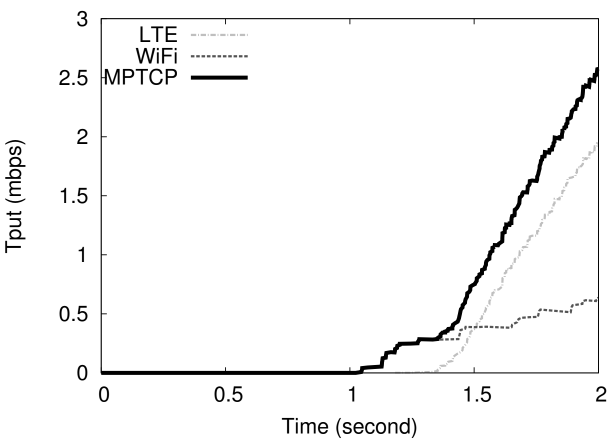
1. LTE for primary subflow, *coupled* congestion control.
2. LTE for primary subflow, *decoupled*[5] congestion control.
3. WiFi for primary subflow, *coupled* congestion control.
4. WiFi for primary subflow, *decoupled* congestion control.

At each location, we measured 10 runs for each of the 4 configurations, along both the uplink and the downlink. Thus, each of the four configurations has 140 data points ($(10+10)\times 7$).

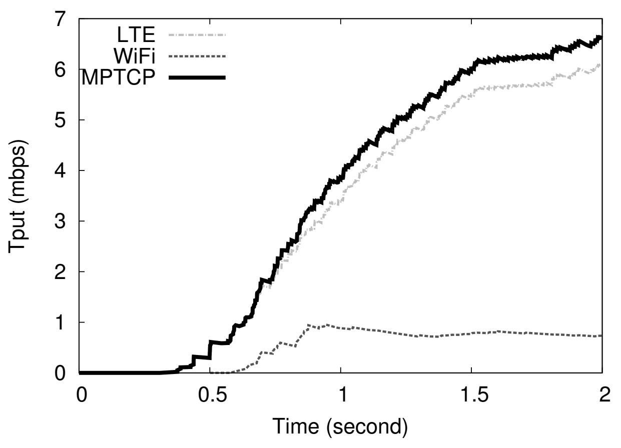
To quantify the throughput difference between configurations, we compute:

$$r_{network} = \frac{|MPTCP_{LTE,coupled} - MPTCP_{WiFi,coupled}|}{MPTCP_{WiFi,coupled}}, \text{ or}$$

[5]Here, the decoupled congestion control uses TCP Reno for each subflow.

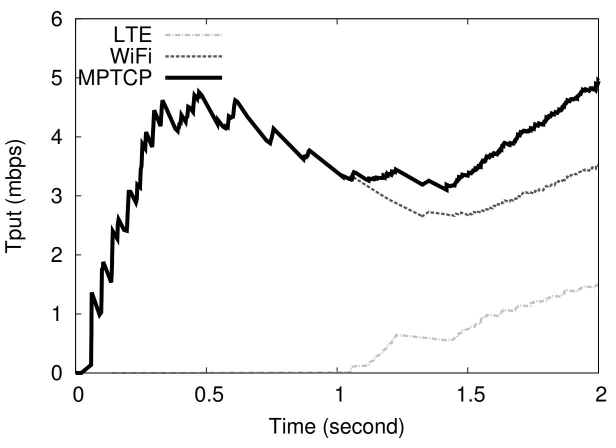


(a) Using WiFi for the primary subflow

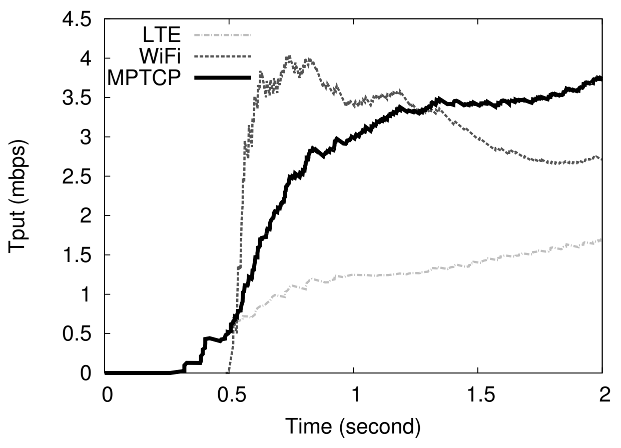


(b) Using LTE for the primary subflow

Figure 9: MPTCP throughput over time, measured at a location where LTE has higher throughput than WiFi. MPTCP throughput grows faster over time when using LTE for the primary subflow.



(a) Using WiFi for the primary subflow



(b) Using LTE for the primary subflow

Figure 10: MPTCP throughput over time, measured at a location where WiFi has higher throughput than LTE. MPTCP throughput grows faster over time when using WiFi for the primary subflow.

$$r_{network} = \frac{|MPTCP_{LTE,decoupled} - MPTCP_{WiFi,decoupled}|}{MPTCP_{WiFi,decoupled}}.$$

Here, $r_{network}$ is the relative throughput difference when using different networks for primary subflow. $MPTCP_{n,c}$ is the throughput measured when using network n for primary subflow and using congestion control algorithm c. We also compute:

$$r_{cwnd} = \frac{|MPTCP_{LTE,decoupled} - MPTCP_{LTE,coupled}|}{MPTCP_{LTE,coupled}}, \text{ or}$$

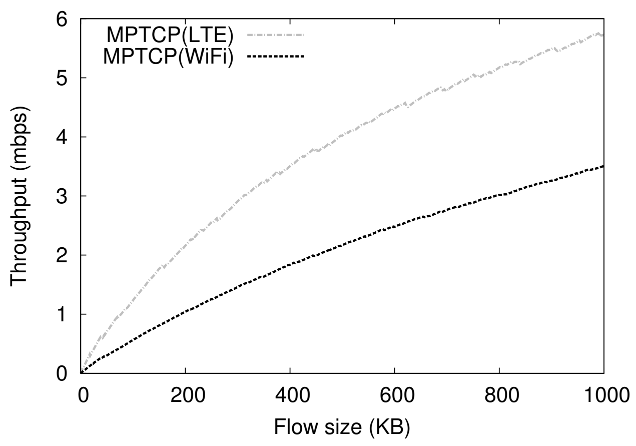
$$r_{cwnd} = \frac{|MPTCP_{WiFi,decoupled} - MPTCP_{WiFi,coupled}|}{MPTCP_{WiFi,coupled}}.$$

Here, r_{cwnd} is the relative throughput difference when using different congestion-control algorithms.

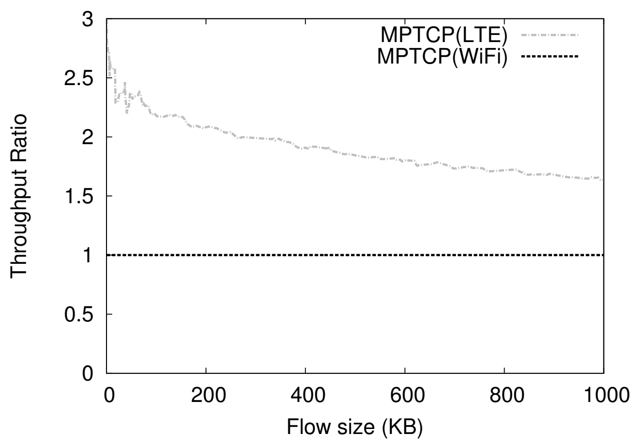
Figure 13 shows the CDF of the relative throughput difference between using coupled and decoupled congestion control for three flow sizes: 10 KBytes, 100 KBytes, and 1 MByte. The rightmost CDF curve corresponds to the relative difference for 1 MByte, while the left-most one is for 10 KBytes. Thus, using different congestion control algorithms has a greater effect on larger flow sizes. In Figure 14, a pair-wise comparison between using different networks (labeled with"Network") and using different congestion control algorithms (labeled with "CC") for each flow size shows the following:

1. For small flow sizes, throughput is more affected by the choice of network for the primary subflow. For example, in Figure 14a and 14b, "Network" is to the right of "CC".
2. For large flow sizes, throughput is more affected by the choice of congestion control (decoupled or coupled) algorithms. For example: in Figure 14c, "CC" is to the right of "Network". However, the choice of network for the primary subflow is also important.

In practice, selecting the best network for the primary subflow is more feasible than changing congestion control algorithms for each MPTCP connection, since the primary flow can be defined solely by the MPTCP endpoint initiating the connection, while the congestion-control algorithm requires support at both endpoints.

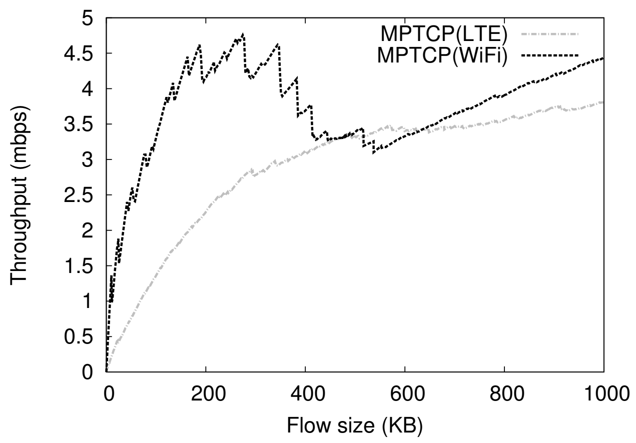


(a) Absolute throughput difference: larger difference between WiFi and LTE for larger flow sizes.

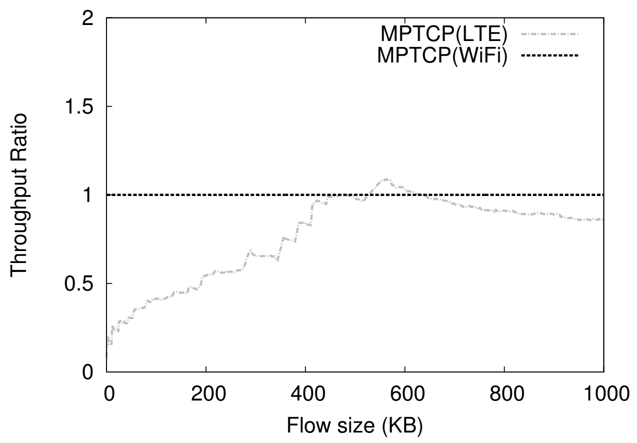


(b) Relative throughput ratio: larger difference between WiFi and LTE for smaller flow sizes.

Figure 11: Absolute throughput difference and relative throughput ratio as a function of flow size when LTE has higher throughput than WiFi.



(a) Absolute throughput difference: larger difference between WiFi and LTE for larger flow sizes.



(b) Relative throughput ratio: larger difference between WiFi and LTE for smaller flow sizes.

Figure 12: Absolute throughput difference and relative throughput ratio as a function of flow size when WiFi has higher throughput than LTE.

3.6 Full-MPTCP and Backup Mode

In Sections 3.4 and 3.5, all measurements were done using *Full-MPTCP* mode, since our focus was on how MPTCP's throughput changes under different configurations, when all paths are fully utilized. *Backup Mode* is an MPTCP mode where only a subset of paths are used to save energy, especially on power-constrained mobile devices. In this section, we first show how *Backup Mode* differs from *Full-MPTCP* at the per-packet level. Then we discuss the energy efficiency of both *Full-MPTCP* and *Backup Mode*.

3.6.1 Packet-Level Behavior of Full-MPTCP and Backup Mode

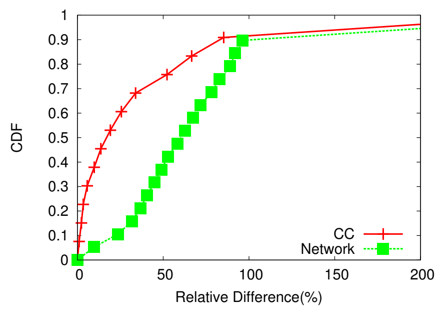
Figure 15 shows the packet-transmission pattern over time for a long flow employing MPTCP, using *Full-MPTCP* and *Backup Mode*. We use `tcpdump` at the MPTCP client to log packet transmission and ack reception times. In Figure 15, we plot a vertical line at time *t* if there is a packet sent or ack received at time *t* in the `tcpdump` trace. $t = 0$ is the time when the first SYN packet is sent.

Each sub-figure contains the packet-transmission patterns on both the WiFi and LTE interfaces for one MPTCP flow. Sub-figures on the left column are packet transmission (sending and receiving) patterns captured when using LTE for the primary subflow, while sub-figures on the right are for using WiFi for the primary subflow.

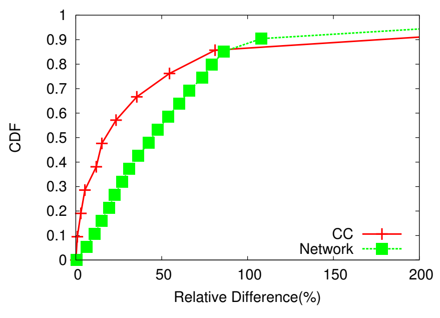
Figures 16a and 16b show that in *Full-MPTCP* mode, packets are transferred through both WiFi and LTE during the entire MPTCP connection.

Figures 15c and 15d illustrate the backup mode where one network is set to be the backup interface. For example, in Figure 15c, when WiFi is set to backup, we only see the SYN and SYN ACK packets transferred during the 3-way handshake procedure at $t = 1$, when the connection establishes a WiFi subflow, as well as FIN and FIN-ACK packets at $t = 19$, when the connection ends. A similar pattern is shown in Figure 15d, when LTE is set to be the backup interface.

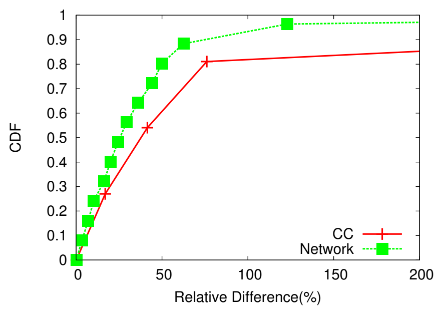
Figures 15e and 15f show packet transmissions in backup mode, when the primary network is disabled mid-flow. We disable the interface by setting the interface to "multipath off" in `iproute`. In Figure 15e, WiFi is set to backup. When the connection starts, it



(a) 10 KB



(b) 100 KB



(c) 1 MB

Figure 14: CDF of relative difference for using different networks for primary subflow (labeled as "Network") vs using different congestion control algorithms (labeled as "CC"), across 3 flow sizes. Median values for CC curves: 16% for "10 KB", 16% for "100 KB", and 34% for "1 MB". Thus, using different congestion control algorithms has more impact on larger flows. Median values for Network curves: 60% for " 10 KB", 43% for "100 KB" , and 25% for "1 MB". Thus, using a different network for the primary subflow has the greatest impact on smaller flows.

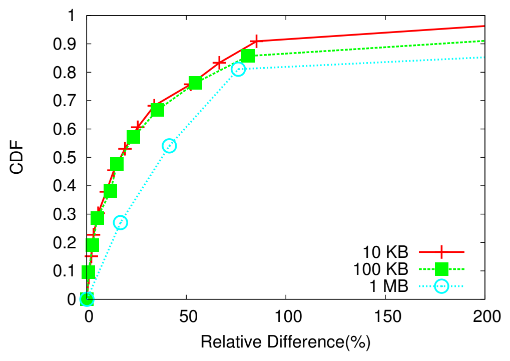


Figure 13: CDF of relative difference between $MPTCP_{coupled}$ and $MPTCP_{decoupled}$, for different flow sizes. The median relative difference for each flow size: 16% for 10 KBytes, 16% for 100 KBytes and 34% for 1MByte. Thus, throughput for larger flow sizes is most affected by the choice of congestion control.

transfers data through LTE. At $t = 7$, we disable LTE, so no data can be transferred over that interface. We see that the subflow over WiFi is brought up and transfers data until the flow ends. A similar behavior is seen in Figure 15f.

In Figures 15g and 15h, we disable one network by unplugging the USB cable connecting the phone to the laptop instead of disabling it using `iproute`. Interestingly, we observe different behaviors in this experiment. Figure 15h shows that when LTE is set to backup and we unplug WiFi in the middle of the transfer (at $t = 6$), the LTE path is brought up immediately to finish transferring the rest of the data. This behavior is similar to when WiFi was disabled by changing `iproute`. However, in Figure 15g, when WiFi is set to backup and we unplug the LTE network in the middle of the transfer (at $t = 3$), the client only transfers one `TCP Window Update` packet to the server through the WiFi subflow and then halts. At $t = 68$, we re-connect the phone with the laptop. Then the connection resumes, transfers the rest of the data through the LTE subflow, and ends the session by sending FIN packets on both path.

The reason why disabling paths by physically disconnecting them can cause different behaviors from disabling them in software is still under investigation.

3.6.2 Energy Efficiency in Backup Mode

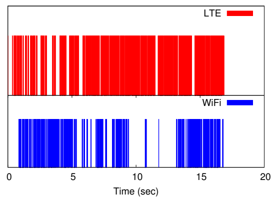
As shown in Figure 15c and 15d, if MPTCP is set to *Backup Mode*, the backup interface still transfers SYN and FIN packets when a connection starts and ends. In Figure 16, we show that in certain configurations, these SYN/FIN packets can consume excessive amounts of energy on a mobile device. Here, we measure the power level of the tethered phones using a power monitor [12], when each phone serves as the backup or non-backup interface. In all sub-figures of Figure 16, the base power consumed is 1 Watt. This is the power consumed when the network interfaces are not active. It is the total power consumed by the other parts of the phone, such as the screen and the CPU.

Figure 16a shows the power level of LTE when it is actively transmitting data, i.e., WiFi is set as a backup interface. Similarly, Figure 16b shows the power level of WiFi when it is active. We can see that the WiFi power level is much lower than that of LTE. Also, in Figure 16a, after the FIN packet is sent, there is a 15-second period in which the LTE power level stays at 2 Watts, instead of the 1-Watt base power level. The energy consumed in this 15-second period is called the "Tail Energy" [3, 7].

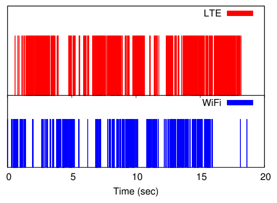
Figures 16c and 16d show the power level when WiFi or LTE is set to be the backup interface. In Figure 16d, the energy consumed by WiFi is negligible. However, in Figure 16c, when a SYN or a FIN packet is transmitted through LTE, the power level stays high for about 15 seconds due to the "Tail Energy" effect. Thus, even if only SYN and FIN packets are transferred through LTE, the LTE interface still consumes an excessive amount of energy. For flows shorter than 15 seconds, little energy can be saved if the LTE interface is set to be the backup interface. To actually reduce energy consumption in this case, *fast dormancy* [9] should be used to quickly put the LTE interface in the low-power mode after a SYN and FIN. Alternatively, the *Backup Mode* should be implemented in a *break-before-make* manner. Prior work [16] has proposed *Single-Path Mode*, which establish a new MPTCP subflow only after the current subflow is inactive, at the expense of two more round-trip times compared with the current *Backup Mode*.

4. MOBILE APP TRAFFIC PATTERNS

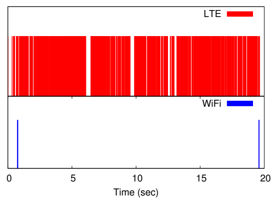
So far, our measurements have looked at the flow-level performance of TCP over WiFi or LTE, and of MPTCP over both WiFi and LTE. We next turn to how the choice of networks for a multi-homed mobile device affects application-level performance as perceived by



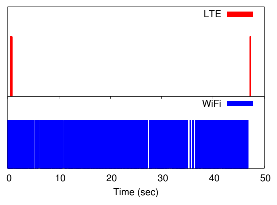
(a) *Full-MPTCP*, using LTE for primary subflow.



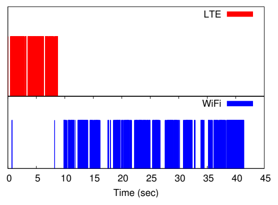
(b) *Full-MPTCP*, using WiFi for primary subflow.



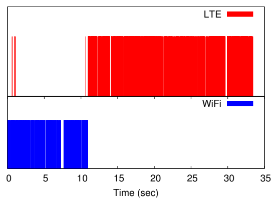
(c) *Backup Mode*, using LTE for primary subflow, WiFi for backup.



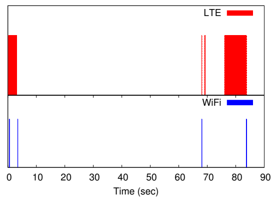
(d) *Backup Mode*, using WiFi for primary subflow, LTE for backup.



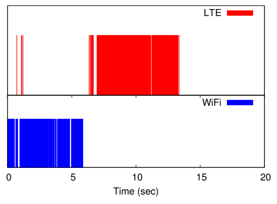
(e) *Backup Mode*, using LTE for primary subflow, WiFi for backup. Then set LTE to "multipath off" at $t = 9$ sec.



(f) *Backup Mode*, using WiFi for primary subflow, LTE for backup. Then set WiFi to "multipath off" at $t = 11$ sec.



(g) *Backup Mode*, using LTE for primary subflow, WiFi for backup. Unplug LTE at $t = 3$ sec. *But WiFi does not continue transferring the rest of the data.* Plug LTE back at $t = 68$ sec.



(h) *Backup Mode*, using WiFi for primary subflow, LTE for backup. Unplug WiFi at $t = 6$ sec.

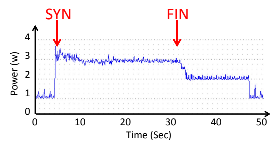
Figure 15: *Full-MPTCP* and *Backup Mode*.

a mobile app that uses one or more of these networks. To measure performance at the level of a mobile app, we first record (Section 4.1) and analyze traffic (Section 4.2) originating from a mobile app, and then replay it under emulated link conditions (Section 5).

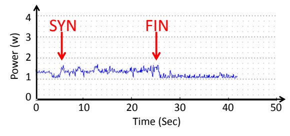
4.1 Record-Replay Tool

Mahimahi [11] is a record-and-replay tool that can record and replay client-server interactions over HTTP. Mahimahi's RecordShell is a UNIX shell that records HTTP traffic and stores it as a set of request-response pairs. Later, during replay, Mahimahi's ReplayShell—another modified UNIX shell—matches incoming requests to stored requests, ignoring time-sensitive fields in the request header (eg. If-Modified-Since) that have likely changed since recording.

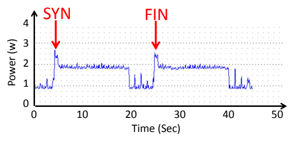
Mahimahi also includes shells to emulate network delays and fixed-rate and variable-rate network links using packet-delivery traces. We extend these capabilities and develop a new shell, MpShell, to emulate multiple links along with their associated link delays. This allows us to mimic a multi-homed mobile phone that can use both cellular and WiFi links. We use a trace-driven approach, as Mahimahi does, to emulate both the cellular and WiFi links.



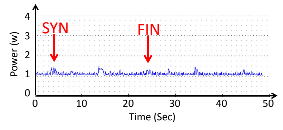
(a) LTE power level when used for non-backup subflow.



(b) WiFi power level when used for non-backup subflow.



(c) LTE power level when used for backup subflow.



(d) WiFi power level when used for backup subflow.

Figure 16: Power level for LTE and WiFi when used as non-backup subflow. LTE has a much higher power level than WiFi in non-backup mode. LTE also consumes excessive amount of energy even in backup mode.

Because Mahimahi is agnostic to the specific client or server that generates the HTTP traffic, we use it to record all HTTP traffic to and from a mobile app running inside an Android emulator. Later, using ReplayShell and MpShell, we run the same mobile app within the Android emulator under appropriately emulated network conditions. This enables us to evaluate how MPTCP—or any other multipath-capable transport—affects application performance of a real mobile app.

4.2 Traffic Patterns of Mobile Apps

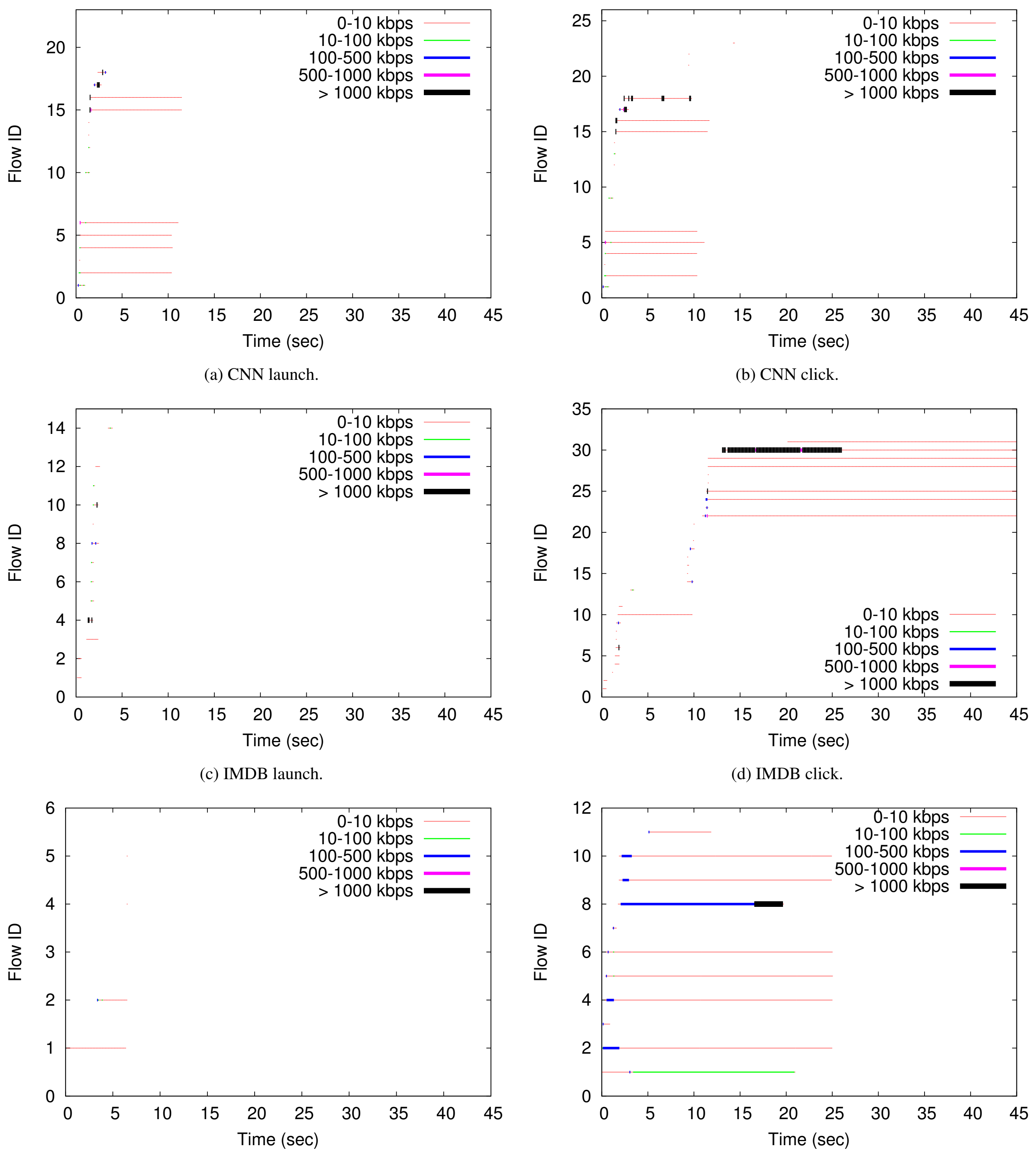
Figure 17 shows typical traffic patterns we observed across different mobile apps run inside RecordShell. We observe that apps tend to initiate multiple TCP connections when launched or in response to a user interaction. Most of these connections only transfer a small amount of data (eg. connection ID 2 in Figure 17c). Some connections, such as connection ID 2 in Figure 17a, persist after small data transfers.

A few connections, such as connection ID 30 in Figure 17d and connection ID 8 in Figure 17f, transfer significant amounts of data, lasting several seconds. The first example (ID 30) occurred when the user clicked a link to play a movie trailer. The app downloaded the entire trailer in one HTTP request. The second example (ID 8) occurred when the user clicked a PDF file in their Dropbox folder and the app downloaded the whole file.

In summary, we can categorize app traffic patterns as *short-flow dominated* and *long-flow dominated*. *short-flow dominated* apps have only short connections or long-lived connections with little data transferred. *long-flow dominated* apps have one or multiple long-lasting flows transferring large amounts of data.

5. MOBILE APP REPLAY

We feed the app traffic patterns described in Section 4 into Mahimahi's ReplayShell for subsequent replay. To accurately emulate different network conditions, we use the recorded single-path TCP packet traces on both WiFi and LTE as a proxy for the true packet-



(a) CNN launch.

(b) CNN click.

(c) IMDB launch.

(d) IMDB click.

(e) Dropbox launch.

(f) Dropbox click.

Figure 17: Traffic patterns for app launching and user interacting. Figures 17d and 17f show the “long-flow dominated’ traffic pattern, the other figures show the “short-flow dominated” pattern.

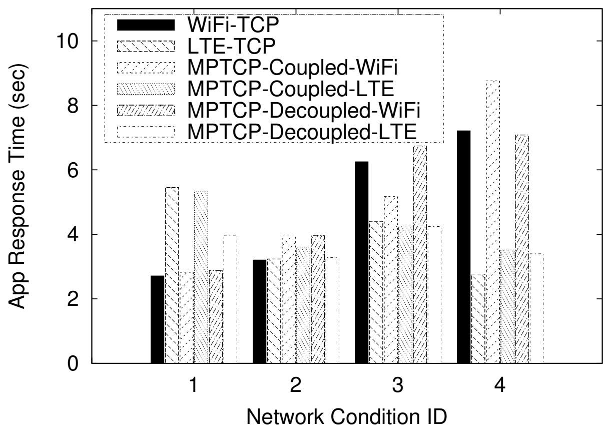


Figure 18: CNN app-response time under different network conditions.

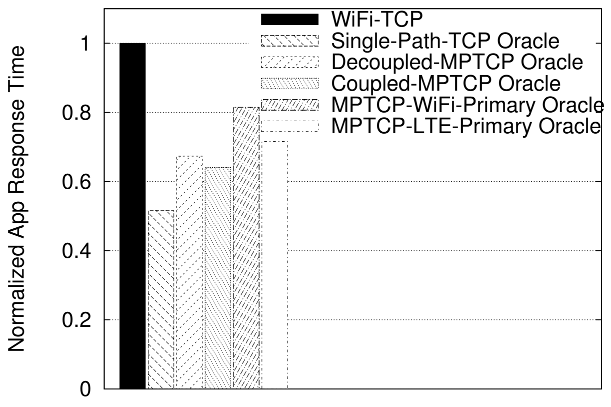


Figure 19: CNN normalized app-response reduction by different oracle schemes.

delivery trace for WiFi and LTE[6]. We use these TCP traces to emulate the WiFi and LTE links within MpShell. We emulate 20 distinct network conditions using the WiFi and LTE TCP data previously collected at 20 locations (Section 3.2).

We present results from replaying two traffic patterns. We refer to the first as the *short-flow dominated app* where, as shown in Figure 17a (CNN launch), an app initiates several connections but only transfers a small amount of data on each connection. We refer to the second as the *long-flow dominated app*, where, as shown in Figure 17f (Dropbox user click), an app initiates several connections and transfers a large amount of data for a few seconds over a small subset of the connections. We run each app pattern over the 20 different network conditions (we only show the results from 4 representative conditions due to space limitations). For each network condition, we emulate 6 configurations:

1. WiFi-TCP: Single-path TCP running on WiFi.
2. LTE-TCP: Single-path TCP running on LTE.
3. MPTCP-Coupled-WiFi: MPTCP with coupled congestion control using WiFi for the primary subflow.
4. MPTCP-Coupled-LTE: MPTCP with coupled congestion control using LTE for the primary subflow.
5. MPTCP-Decoupled-WiFi: MPTCP with decoupled congestion control using WiFi for the primary subflow.
6. MPTCP-Decoupled-LTE: MPTCP with decoupled congestion control using LTE for the primary subflow.

Using `tcpdump` during the emulation, we collect the timestamp at the start and end of each HTTP connection. Then we calculate the *app response time*: the time between the start of the first HTTP connection and the end of the last HTTP connection[7].

5.1 Short-Flow Dominated App Replay

Figure 18 shows the app-response time for the CNN app launching in different configurations under different network conditions. For clarity, we only show the emulation results for 4 representative network conditions out of the 20 we emulated; results for other conditions are similar.

[6]This approach does underestimate the true packet-delivery trace of the underlying network because TCP takes a finite duration to reach the link capacity due to Slow Start.

[7]This metric doesn't account for computation time that might be spent in the app itself after the last HTTP connection ends, but this is impossible to measure without instrumenting or rewriting existing applications to report these numbers.

Network Condition IDs 1 and 2 emulate locations where WiFi has a much higher bulk TCP throughput than LTE, and in Network Condition IDs 3 and 4, LTE outperforms WiFi. In Figure 18, we observe that:

1. Selecting the proper network to transmit for single-path TCP significantly affects app-response time. For example, in Network Condition 1, the app response time for WiFi-TCP is 2.7 seconds while LTE-TCP has an app response time of 5.5 seconds, implying that using the proper network for single-path TCP can reduce the app response time by about 2.0x. For a network condition in which LTE has better performance, such as Network Condition ID 4, the app-response times for TCP over WiFi (shown as "TCP-WiFi" in Figure 18) and TCP over LTE (shown as "TCP-LTE" in Figure 18) are 7.2 seconds and 2.8 seconds, respectively. In this case, using LTE can reduce the app response time by 2.6x.
2. Using MPTCP does not provide much improvement for the *short-flow dominated* app pattern. For instance, in Network Condition 1, MPTCP-Coupled-LTE and MPTCP-Decoupled-LTE have app response times of 5.3 and 4.0, respectively. Compared to TCP over LTE, these MPTCP schemes only reduce the app response time by 4% and 15%, much smaller improvements than the 2x improvement seen when using TCP over WiFi compared to TCP over LTE.

In summary, Figure 18 shows that the choice of network for the primary subflow has a strong impact on app response time. This result is consistent with the results we show in Section 3.5.

We also study the extent to which app response times can be reduced if we had access to an optimal network selection algorithm: an oracle that knew the right network to use, given a particular congestion control strategy (coupled vs decoupled) and another oracle that knew the right congestion control strategy to use given a particular choice for the network used by the primary subflow. Figure 19 shows the app-response time with different oracle schemes, averaged across all 20 network conditions and normalized by the app-response time with single-path TCP over WiFi (the default on Android today). The oracle schemes are:

1. Single-Path-TCP Oracle: Uses single-path TCP and knows which network minimizes app response time.
2. Decoupled-MPTCP Oracle: Uses MPTCP decoupled conges-

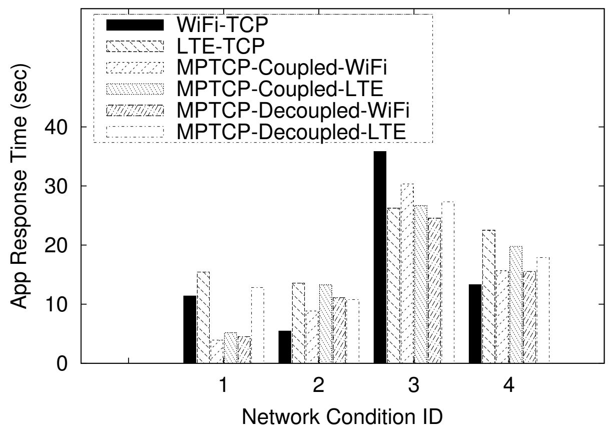


Figure 20: Dropbox app-response time under different network conditions.

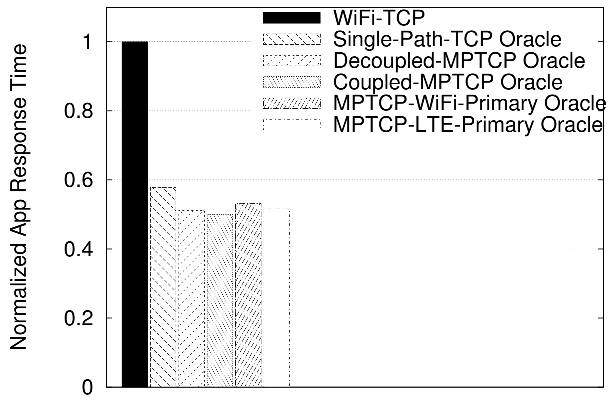


Figure 21: Dropbox normalized app-response reduction by different oracle schemes.

tion control and knows which network to use for the primary subflow to minimize app response time.

3. Coupled-MPTCP Oracle: Uses MPTCP coupled congestion control and knows which network to use for the primary subflow to minimize app response time.
4. MPTCP-WiFi-Primary Oracle: Uses MPTCP with WiFi for primary subflow and knows which congestion-control algorithm to use to minimize app response time.
5. MPTCP-LTE-Primary Oracle: Uses MPTCP with LTE for primary subflow and knows which congestion-control algorithm to use to minimize app response time.

We can see in Figure 19 that the 50% reduction in app response time with Single-Path-TCP Oracle is the most substantial, while the reductions with the MPTCP Oracles range from 15% to 35%. This suggests that MPTCP does not reduce app response time as significantly as selecting the right network for single-path TCP does.

5.2 Long-flow Dominated App Replay

Figures 20 and 21 show emulation results for the *long-flow dominated* traffic pattern, using the same data analysis methods and oracles as used in the previous subsection.

In Figure 20, Network Condition IDs 1 and 2 emulate places where WiFi has a much higher TCP throughput than LTE, and Network Condition IDs 3 and 4 represent places where LTE outperforms WiFi. We observe that:

1. Using MPTCP helps to reduce app response-time. For example, at Network Condition 1, when using single-path TCP, the app response time is 10 seconds for WiFi and 15 seconds for LTE. When using MPTCP, the app response time is 5 seconds.
2. Selecting the proper network is important: for example, at Network Condition ID 2, the app response time for MPTCP-Coupled-WiFi is 8 seconds, but if LTE is used for the primary flow, response time increases to 14 seconds.
3. Selecting the proper congestion control algorithm also affects app response time. For example, at Network Condition ID 1, when using LTE for the primary subflow, the app response time for coupled congestion control is 4 seconds, while the response time with decoupled congestion control is 13 seconds.

In Figure 21, we can see that:

1. MPTCP Oracles reduce the app response time by up to 50%, while the Single-Path-TCP Oracle only reduces app response time by 42%. So using MPTCP can help improve performance for "long-flow dominated" apps.
2. For MPTCP Oracles, both selecting the proper network for the primary flow and selecting the appropriate congestion control can reduce the normalized app response time by about 50%, implying that both mechanisms are almost equally beneficial to "long-flow dominated" apps.

6. RELATED WORK

We discuss related work under two headings: prior work comparing WiFi with cellular network performance and Multi-Path TCP.

6.1 WiFi-Cellular Comparison

Several prior papers compare cellular network performance with WiFi. Sommers et al. [20] analyzed crowd-sourced data from `SpeedTest.net`. Each data sample represents one run of a TCP upload/download test triggered by a mobile phone user, when the phone is connected to the Internet through either WiFi or a cellular network. We also collect our data in a crowd-sourced manner. However, our mobile app, *Cell vs WiFi*, measures both WiFi and cellular network performance on the same device at (almost) the same time. Thus, our dataset reflects the performance difference observed from a single device at almost the same time. Deshpande et al. [8] measured both 3G and WiFi performance simultaneously using a single device, but their measurement was focused on a vehicular setting and they only measured 3G, not LTE. Our dataset focuses on LTE measurements instead. In our app, we used an activity-recognition API provided by Google [1], which shows that most of our measurements happen when users are still. Moreover, our data is collected in a crowd-sourced manner, allowing it to capture a wide diversity of conditions.

6.2 Multi-Path TCP

Multipath TCP (MPTCP) [21], and its recent implementation in iOS 7 [13] allow a single TCP connection to use multiple paths. MPTCP provides TCP's reliable, in-order bytestream abstraction while taking advantage of multiple paths for increased throughput and fault tolerance. Previous work has looked at MPTCP in a mobile context: Raiciu et al. studied mobility with MPTCP [19]. Pluntke et al. designed a scheduler that picks radio interfaces with a view to reduce mobile energy consumption [18]. Paasch et al. proposed different MPTCP modes to be used by mobile devices for Mobile/WiFi handover [16]. Barlow-Bignell et al. [4] studied MPTCP performance in the presence of WiFi interference where multiple devices connected to the same AP could interfere with each other if they transmitted packets simultaneously. Closest to our work is the work of Chen et al., who measured MPTCP performance over cellular networks and WiFi [6]. Their measurement focuses on using different number of subflows, and fine-grained statistics, such as out-of-order delivery and round trip times. Instead, our focus is on studying the choice of networks for the primary subflow, the choice of congestion-control modes, MPTCP's energy consumption, and MPTCP's effect on higher-level metrics such as flow completion times and app-response times.

7. CONCLUSION AND FUTURE WORK

We presented a measurement study of single-path TCP and MPTCP over LTE and WiFi networks. For single-path TCP, we found that LTE outperforms WiFi 40% of the time – a higher fraction than one might expect at first sight. We also find that MPTCP offers no appreciable benefit over TCP for shorter flows, but it does improve performance for longer flows. For MPTCP, we found that, especially for short flows, it is crucial to select the correct network for the primary subflow. For long flows, it is equally important to select the proper congestion control algorithm. To understand how TCP and MPTCP over LTE and WiFi can affect mobile app performance, we analyzed mobile apps' traffic patterns and categorized apps as either *short-flow dominated* or *long-flow dominated*. For each category, we emulated app traffic patterns and the results we observed match our MPTCP measurement findings.

Our findings also bring up new research questions: how can we automatically decide when to use single path TCP and when to use MPTCP? How should we decide which network to use for TCP, or which network to use for a subflow with MPTCP? We think these are non-trivial questions due to the high mobility of devices and rapidly-changing network conditions. Also, with energy consumption being a major concern for mobile devices, how can we make the decisions when trying to minimize energy consumption? We plan to explore each of these future directions.

8. ACKNOWLEDGMENTS

We are grateful to the IMC reviewers, and our shepherd, Dr. Arnaud Legout in particular, for many helpful comments. We thank Katrina LaCurts, Lenin Ravindranath, and Amy Ousterhout for their thoughtful suggestions. This work was also supported in part by NSF grants 1407470 and 1161964. We also thank the members of the MIT Center for Wireless Networks and Mobile Computing (Wireless@MIT), including Amazon.com, Cisco, Google, Intel, Mediatek, Microsoft, ST Microelectronics, and Telefonica, for their support.

REFERENCES

[1] Recognizing the user's current activity. `http://developer.android.com/training/location/activity-recognition.html`.

[2] Android telephony manager api. `http://developer.android.com/reference/android/telephony/TelephonyManager.html`.

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *IMC*, 2009.

[4] J. Barlow-Bignell, C. da Silva, J. Gjengset, and P. Oliha. Wireless interference and multipath tcpreducing 3g energy consumption on mobile devices, 2013.

[5] S. Barré, C. Paasch, and O. Bonaventure. Multipath tcp: from theory to practice. In *NETWORKING 2011*, pages 444–457. Springer, 2011.

[6] Y.-C. Chen, Y. 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.

[7] S. Deng and H. Balakrishnan. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 181–192. ACM, 2012.

[8] P. Deshpande, X. Hou, and S. R. Das. Performance comparison of 3g and metro-scale wifi for vehicular network access. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 301–307. ACM, 2010.

[9] UE "Fast Dormancy" behavior, 2007. 3GPP discussion and decision notes R2-075251.

[10] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec. Mptcp is not pareto-optimal: performance issues and a possible solution. In *CoNEXT*, 2012.

[11] Mahimahi. `http://mahimahi.mit.edu`.

[12] Monsoon power monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[13] Apple ios 7 surprises as first with new multipath tcp connections. http://www.networkworld.com/news/2013/091913-ios7-multipath-273995.html.

[14] Multipath tcp - linux kernel implementation. http://www.multipath-tcp.org/.

[15] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic. MultiNets: Policy Oriented Real-Time Switching of Wireless Interfaces on Mobile Devices. In *RTAS*, 2012.

[16] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring mobile/wifi handover with multipath tcp. In *CellNet*, 2012.

[17] C. E. Perkins. Mobile ip. *Communications Magazine, IEEE*, 1997.

[18] C. Pluntke, L. Eggert, and N. Kiukkonen. Saving mobile device energy with multipath tcp. In *MobiArch*, 2011.

[19] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley. Opportunistic mobility with multipath tcp. In *MobiArch*, 2011.

[20] J. Sommers and P. Barford. Cell vs. wifi: on the performance of metro area mobile connections. In *IMC*, 2012.

[21] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, 2011.

[22] X. Zhao, C. Castelluccia, and M. Baker. Flexible network support for mobility. In *MobiCom*, 1998.