A Measurement-based Study of MultiPath TCP Performance over Wireless Networks

Yung-Chih Chen
School of Computer Science
University of Massachusetts
Amherst, MA USA
yungchih@cs.umass.edu

Yeon-sup Lim
School of Computer Science
University of Massachusetts
Amherst, MA USA
ylim@cs.umass.edu

Richard J. Gibbens
Computer Laboratory
University of Cambridge
Cambridge, UK
richard.gibbens@cl.cam.ac.uk

Erich M. Nahum
IBM Thomas J. Watson
Research Center
Yorktown Heights, NY USA
nahum@us.ibm.com

Ramin Khalili
T-Labs, Deutsche Telekom
Berlin, Germany
ramin@net.t-labs.tu-berlin.de

Don Towsley
School of Computer Science
University of Massachusetts
Amherst, MA USA
towsley@cs.umass.edu

ABSTRACT

With the popularity of mobile devices and the pervasive use of cellular technology, there is widespread interest in hybrid networks and on how to achieve robustness and good performance from them. As most smart phones and mobile devices are equipped with dual interfaces (WiFi and 3G/4G), a promising approach is through the use of multi-path TCP, which leverages path diversity to improve performance and provide robust data transfers. In this paper we explore the performance of multi-path TCP in the wild, focusing on simple 2-path multi-path TCP scenarios. We seek to answer the following questions: How much can a user benefit from using multi-path TCP over cellular and WiFi relative to using the either interface alone? What is the impact of flow size on average latency? What is the effect of the rate/route control algorithm on performance? We are especially interested in understanding how application level performance is affected when path characteristics (e.g., round trip times and loss rates) are diverse. We address these questions by conducting measurements using one commercial Internet service provider and three major cellular carriers in the US.

Categories and Subject Descriptors

C.2 [**Computer-communication Networks**]: Network Protocols; C.2.1 [**Network Architecture and Design**]: Wireless communication; C.4 [**Performance of System**]: Measurement techniques, Performance attributes

General Terms

Experimentation; Measurement; Performance

IMC'13, October 23–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2504730.2504751.

Keywords

Multi-Path TCP; MPTCP; Congestion Control; Measurements; Wireless; Cellular Networks; 4G; LTE; 3G; CDMA

1. INTRODUCTION

Many users with mobile devices can access the Internet through both WiFi and cellular networks. Typically, these users only utilize one technology at a time: WiFi when it is available, and cellular otherwise. Research has also focused on the development of mechanisms that switch between cellular and WiFi as the quality of the latter improves and degrades. This results in a quality of service that is quite variable over time. As data downloads (e.g., Web objects, video streaming, etc.) are dominant in the mobile environment, this can result in highly variable download latencies.

In this paper we explore the use of a promising recent development, multi-path rate/route control, as a mechanism for providing robustness by reducing the variability in download latencies. Multi-path rate/route control was first suggested by Kelly [17]. Key et al. [18] showed how multipath rate/route control provides load balancing in networks. Han et al. [10] and Kelly & Voice [16] developed theoretically grounded controllers that have since been adapted into Multipath TCP (MPTCP) [8], which is currently being standardized by the IETF.

Numerous studies, both theoretical and experimental, have focused on the benefits that MPTCP bring to long-lived flows. These studies have resulted in a number of changes in the controller [14,19,32], all in an attempt to provide better fairness and better throughput in the presence of fairness constraints. However, to date, these studies have ignored the effect of multi-path on finite duration flows. It is well known that most Web downloads are of objects no more than one MB in size, although the tail of the size distribution is large. Moreover, online video streaming to mobile devices is growing in popularity and, although it is typically thought of as a download of a single large object, usually consists of a sequence of smaller data downloads (500 KB - 4 MB) [27]. Thus it important to understand how the use of MPTCP might benefit such applications.

In this paper we evaluate how MPTCP performs in the wild with a common wireless environment, namely using

both WiFi and cellular simultaneously. We conduct a range of experiments varying over time, space, and download size. We utilize three different cellular providers (two 4G LTEs, one 3G CMDA) and one WiFi provider, covering a broad range of network characteristics in terms of bandwidth, packet loss, and round-trip time. To assess how effectively MPTCP behaves, we report not only multi-path results, but also single-path results using the WiFi and cellular networks in isolation. We report standard networking metrics (download time, RTT, loss) as well as MPTCP specific ones (e.g., share of traffic sent over one path, packet reordering delay). We also examine several potential optimizations to multi-path, such as simultaneous SYNs, different congestion controllers, and using larger numbers of paths.

This paper makes the following contributions:

- We find that MPTCP is robust in achieving performance at least close to the best single-path performance, across a wide range of network environments. For large transfers, performance is better than the best single path, except in cases with poor cellular networks.
- Download size is a key factor in how MPTCP performs, since it determines whether a subflow can get out of slow start. It also affects how quickly MPTCP can establish and utilize a second path. For short transfers (i.e., less than 64 KB), performance is determined by the round-trip time (RTT) of the best path, typically WiFi in our environment. In these cases, flows never leave slow start and are limited by the RTT. For larger transfers, in the case of LTE, as download size increases, MPTCP achieves significantly improved download times by leveraging both paths simultaneously, despite varying path characteristics.
- Round trip times over the cellular networks can be very large and exhibit large variability, which causes significant additional delay due to reordering out-of-order segments from different paths. This is particularly pronounced on the 3G network we tested. This impacts how well MPTCP can support multimedia applications such as video.
- Using multiple flows improves performance across download sizes. For small transfers, this is because more flows allow more opportunity to exploit slow start. For large transfers, this is due to utilizing the available network bandwidth in a more efficient way. Connecting multiple flows simultaneously, rather than serially, only improves performance for small transfers, which are most sensitive to RTT. Different congestion controllers do not appear to have a significant impact on performance for small file transfers. For larger file transfers, we observe that the default congestion controller of MPTCP (coupled [24]) does not perform as well as its alternative, olia [19].

The remainder of this paper is organized as follows: Section 2 provides some background on Cellular networks and MPTCP. We describe our experimental methodology in Section 3. Section 4 presents an overview of our results, and Section 5 looks at latency in detail. We discuss our some implications in Section 6, discuss about related work in Section 7, and conclude in Section 8.

2. BACKGROUND

This section provides background and basic characteristics of cellular data and WiFi networks, and MPTCP control mechanisms needed for the rest of the paper.

2.1 Cellular data and WiFi networks

With the emerging population of smart phones and mobile devices, to cope with the tremendous traffic growth, cellular operators have been upgrading their access technologies from the third generation (3G) to the fourth generation (4G) networks. 3G Services are required to satisfy the standards of providing a peak data rate of at least 200 K bits per second (bps). The specified peak speed for 4G services is 100 Mbps for high mobility communication, and 1 Gbps for low mobility communication. In western Massachusetts, where we perform our measurements, AT&T and Verizon networks have their 4G Long Term Evolution (LTE) widely deployed, while Sprint only has 3G Evolution-Data Optimized (EVDO) available.

Cellular data networks differ from WiFi networks in that they provide broader signal coverage and more reliable connectivity under mobility. Furthermore, since wireless link losses result in poor TCP throughput and are regarded as congestion by TCP, cellular carriers have augmented their systems with extensive local retransmission mechanisms [3], transparent to TCP, which mitigate TCP retransmissions and reduce the waste of precious resources in cellular networks. Although these mechanisms reduce the impact of losses dramatically and improves TCP throughput, they come at the cost of increased delay and rate variability.

On the other hand, WiFi networks provide shorter packet round trip times (RTTs) but higher loss rates. Throughout our measurements, we observe that the loss rates over 3G/4G networks are generally lower than 0.1%, while those of WiFi vary from 1% to 3%. From our observations, the average RTT for WiFi networks is about 30 ms, while that of 4G cellular carriers usually has base RTTs of 60 ms, and can increase by four to ten fold in a single 4G connection (depending on the carrier and the flow sizes, see Section 5), and 20-fold in 3G networks. We note that, although cellular networks in general have larger packet RTTs, in many of our measurements, WiFi is no longer faster than 4G LTE, and this provides greater incentive to use multi-path TCP for robust data transport and better throughput.

2.2 MPTCP

We discuss how the current MPTCP protocol establishes a connection and describe the different type of congestion controllers used by MPTCP.

2.2.1 Connection and Subflow Establishment

Once an MPTCP connection is initiated and the first flow is established, each end host knows one of its peer's IP addresses. When the client has an additional interface, for example, a 3G/4G interface, it will first notify the server its additional IP address with an *Add Address* option over the established subflow and send another SYN packet with a *JOIN* option to the server's known IP address. With this MPTCP-JOIN option, this subflow will be associated with a previously established MPTCP connection. As many of the mobile clients are behind Network Address Translations (NATs), when the server has an additional interface, it is difficult for the server to directly communicate with the mo-

bile client as the NATs usually filter out unidentified packets [26]. The server thus sends an *Add Address* option on the established subflow, notifying the client its additional interface. As soon as the client receives it, it sends out another SYN packet with *JOIN* option to the server's newly notified IP address, together with the exchanged hashed key for this MPTCP connection, and initiates a new subflow [8].

2.2.2 Congestion Controller

As each MPTCP subflow behaves as a legacy New Reno TCP flow except for the congestion control algorithms, after the 3-way handshake, each subflow maintains its own congestion window and retransmission scheme during data transfer, and begins with a slow-start phase that doubles the window per RTT [2] before entering the congestion avoidance phase.

We briefly describe the different congestion avoidance algorithms that have been proposed for MPTCP. Let us denote by w_i and rtt_i the congestion window size and round trip time of subflow i, and denote by w the total congestion window size over all the subflows. Also, let $\mathcal{R}$ be the set of all subflows.

Uncoupled TCP Reno (reno): The simplest algorithm that one can imagine is to use TCP New Reno over each of the subflows:

- For each ACK on flow i: $w_i = w_i + \frac{1}{w_i}$
- For each loss on flow i: $w_i = \frac{w_i}{2}$.

This does not satisfy the design goal of MPTCP [24], as it fails to provide congestion balancing in the network. We use this algorithm as the baseline and refer to it as reno.

Coupled: The coupled congestion control algorithm was introduced in [32] and is the default congestion controller of MPTCP [8, 24]. It couples the increases and uses the unmodified behavior of TCP in the case of a loss. The coupled congestion control algorithm works as follows:

- For each ACK on flow i: $w_i = w_i + \min(\frac{a}{w}, \frac{1}{w_i})$
- For each loss on flow i: $w_i = \frac{w_i}{2}$

a is a function of w_i and rtt_i for all $i \in \mathcal{R}$. As discussed in [19], this algorithm fails to fully satisfy the design goal of MPTCP but provides better congestion balancing than reno.

OLIA: An opportunistic link increase algorithm has been proposed by Khalili et al. [19] as an alternative to the coupled algorithm:

- For each ACK on flow i: $w_i = w_i + \frac{w_i/\text{rtt}_i^2}{(\sum_{p\in\mathcal{R}} w_p/\text{rtt}_p)^2} + \frac{\alpha_i}{w_i}$
- For each loss on flow i: $w_i = \frac{w_i}{2}$

where α_i is a function of w_i and rtt_i for all $i \in \mathcal{R}$. OLIA satisfies the design goals of MPTCP and provides a better congestion balancing than the coupled algorithm [19].

3. MEASUREMENT METHODOLOGY

In this section, we describe our experimental setup and discuss our methodology. Note that all measurements were performed during March 20 to May 7 in three different towns (Sunderland, Amherst, and Hadley) in western Massachusetts. These towns are approximately 10 miles away from each other.

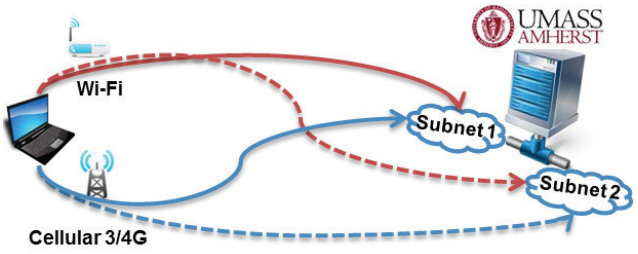

Figure 1: For 2-path MPTCP experiments, only solid-line paths are used. The additional dashed-line paths are included for the 4-path MPTCP experiments.

3.1 Experiment Setup

Figure 1 illustrates our testbed. It consists of a wired server, residing at the University of Massachusetts Amherst (UMass) and a mobile client. For most of the measurements, we focus on the 2-path scenarios (solid lines), where the client has two interfaces activated while the server has only interface in operation. A second interface is only active for performance comparisons between two flows and four flows.

Our server is configured as a multi-homed host, connecting via two Intel Gigabit Ethernet interfaces to two subnets (LANs) of the UMass network. Each Ethernet interface is assigned a public IP address and connected to the LAN via a 1 Gigabit Ethernet cable. The mobile client is a Lenovo X220 laptop and has a built-in 802.11 a/b/g/n WiFi interface. Here we consider two types of WiFi networks: private home WiFi networks and public WiFi hotspots. The home WiFi network is accessed by associating the WiFi interface to a D-Link WBR-1310 a/b/g wireless router connected to a private home network in a residential area. The home network traffic to the Internet is provided by Comcast network which serves users in the same residential community with a maximum download rate of up to 25 Mbps. Note that the actual WiFi download speed varies according to backhaul traffic load, types of home APs used, and users' wireless interfaces [29]. Unless otherwise stated, we refer to a private home network as a WiFi network. The mobile client has three additional cellular broadband data interfaces listed in Table 1, and only uses them one at a time.

Table 1: Cellular devices used for each carrier

Carrier	Device Name	Technology
AT&T	Elevate mobile hotspot	4G LTE
Verizon	LTE USB modem 551L	4G LTE
Sprint	OverdrivePro mobile hotspot	3G EVDO

Both the server and the client are running Ubuntu Linux 12.10 with Kernel version 3.5.7 using the stable release of the MPTCP Kernel implementation [21] version v0.86. The UMass server is configured as an HTTP server. It runs Apache2 on port 8080, as AT&T has a Web proxy running on port 80 which removes all the MPTCP option fields and thus does not allow MPTCP connections. The client uses *wget* to retrieve Web objects of different sizes via all the available paths.

To reduce potential WiFi interference to the working wireless interface, we disable the functionality of WiFi bandwidth sharing on both the AT&T and Sprint devices. Furthermore, though all devices run at different frequencies, to avoid possible interference between these electronic devices, we use USB cables to extend cellular dongles, and use the WiFi and only one cellular device at a time. Therefore, we assume interference among the devices is negligible. Throughout the measurements, cellular reception signals of different carriers (over different places) are in the range between -60dBm and -102dBm, which covers good and weak signals.

Connection parameters.

The default Linux TCP uses an initial slow start threshold value (*ssthresh*) of infinity, and caches parameters for per-destination TCP connections [28]. When losses occur, the *ssthresh* value will be reset and cached for initialization of future TCP connections to the same destination. However, this is shown to be harmful for short flows [13] if an earlier connection to a particular destination encounters a sequence of losses. This is because *ssthresh* will be set to a small value and all the subsequent newly open flows to that destination will have the same small *ssthresh*. Hence, we configure our server such that no parameters of previously closed TCP connections to any destination are cached. Moreover, as we are using cellular networks in nearly loss-free environments (as will be discussed later), a *ssthresh* value of infinity will lead to the case where the cellular path never leaves slow start. The congestion window of the cellular path could then become extremely large and hence suffer severe RTT inflation [3, 15], which can degrade the performance of MPTCP. Therefore, we set the default value of *ssthresh* to 64KB for fair comparisons among different configurations and file sizes, and to mitigate the impact of RTT inflation described above. We use Linux's default initial window size of 10 packets and apply TCP Selective Acknowledgement (SACK) [7].

Receive memory allocation.

As MPTCP requires a larger receive buffer than single-path TCP for out-of-order packets from different paths and uses a shared receive buffer, there is a potential performance degradation if the assigned buffer is too small [26, 32]. To avoid such events during our measurements, we set the maximum receive buffer to 8 MB.

No subflow penalty.

Throughout our experiments, we observe that the current MPTCP implementation by default monitors each flow's bandwidth delay product (BDP). If a particular flow has contributed too many out-of-order packets to the receive buffer, it penalizes that flow by reducing its congestion window by half [26], even though no loss has occurred. In our experiments, as the receive memory is always large enough, this penalization mechanism can only degrade the performance of MPTCP connections. To measure the true performance of MPTCP connections, we remove the penalization scheme from the implementation.

3.2 Experiment Methodology

As the UMass server has two physical interfaces, and the client has a built-in WiFi interface and broadband devices from three different cellular carriers, we conduct measurements of the following configurations:

- Single-path TCP: the UMass server activates its primary interface, and the client enables only one interface (WiFi or cellular). Thus, there are four configurations in this scenario: single path WiFi TCP or single path cellular TCP (through AT&T, Verizon, or Sprint).
- 2-path MPTCP: the UMass server activates its primary interface, while the client enables WiFi and a cellular device. For each configuration, we run back to back measurements of different congestion controllers described in Section 2.2. There are nine configurations in this scenario: client's three settings of two interfaces enabled (WiFi/AT&T, WiFi/Verizon, and WiFi/Sprint) to the server's primary interface with three congestion controller settings.
- 4-path MPTCP: for comparison purposes, we enable the server's secondary interface connected to a different subnet, as illustrated in Figure 1. There are also in total nine different configurations in this scenario.

As Web traffic can be short-lived or long-lived, for each configuration, the client downloads files of different sizes from the server via HTTP. As there is no clear distinction between short flows and long flows, in our measurements, we consider files of sizes 8 KB, 64 KB, 512 KB, and 4 MB as small flows. For large flows, we consider file of sizes 8 MB, 16 MB, and 32 MB. We also consider infinite backlog file transfers for performance purposes (see Section 4.2), and here file downloads are of size 512 MB.

Since network traffic might have dependencies and/or correlation from time to time, and from size to size, in each round of measurements, we *randomize* the sequence of configurations. That is, we randomize the order of file sizes, carriers, the choices of congestion controllers, single-path and multi-path TCP. To capture temporal effects, for each scenario, we conduct measurements for multiple days. To mitigate possible spatial factors, measurements were also performed at multiple locations in the same town, and at different towns in western Massachusetts. Note that we divide a day into four periods: night (0-6 AM), morning (6-12 AM), afternoon (12-6 PM), and evening (6-12 PM). For each period of time at each location, we perform 20 measurements for each configuration.

Furthermore, since cellular 3G/4G antennas have state machines for radio resources allocation and management of energy consumption, the state promotion delay (the time duration required to bring the antenna to ready state) is often longer than packet RTTs [11, 12] and might significantly impact our short flow measurements. Therefore, to avoid this impact, we send two ICMP ping packets to our server before each measurement, and start the measurements immediately after the ping responses are correctly received to ensure that the cellular antenna is in the ready state.

We collect packet traces from both the UMass server and the client using *tcpdump* [30], and use *tcptrace* [31] to analyze the collected traces at both sides.

3.3 Performance Metrics

We are interested in the following performance metrics related to MPTCP and single-path TCP:

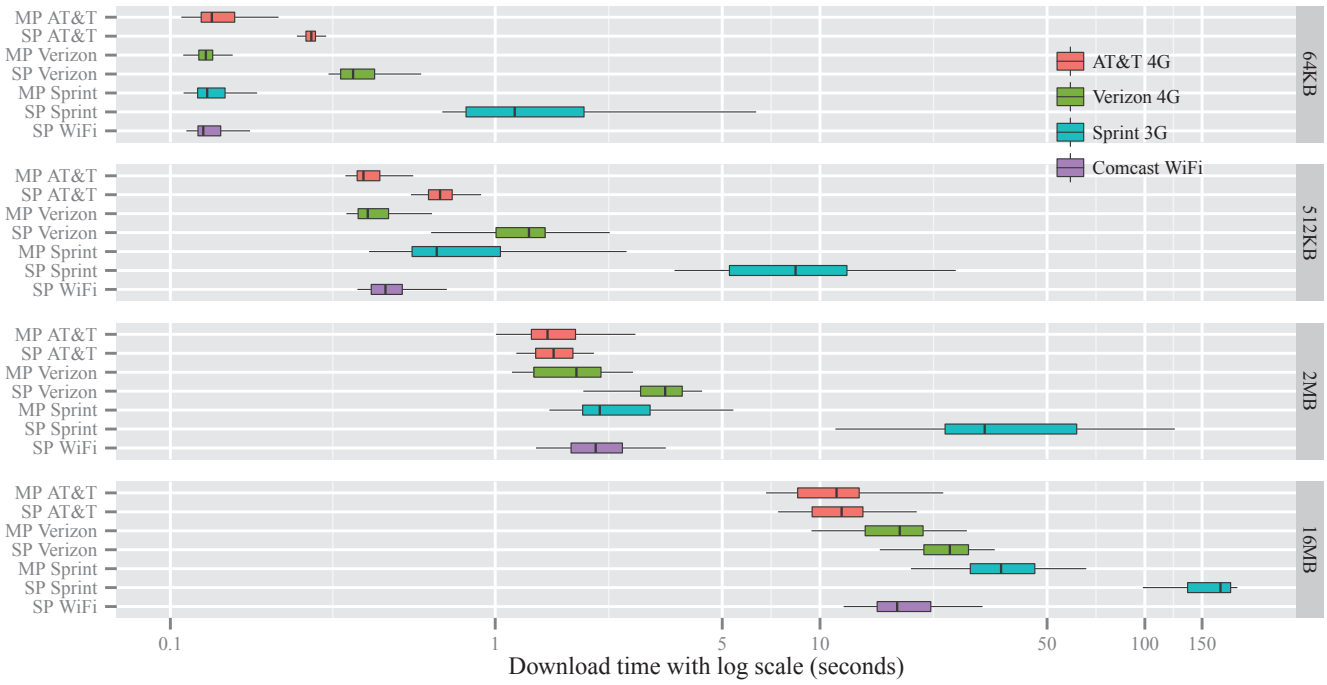

Figure 2: Baseline Download Time: MPTCP and single-path TCP connections for different carriers. The measurements were performed over the course of 24 hours for multiple days.

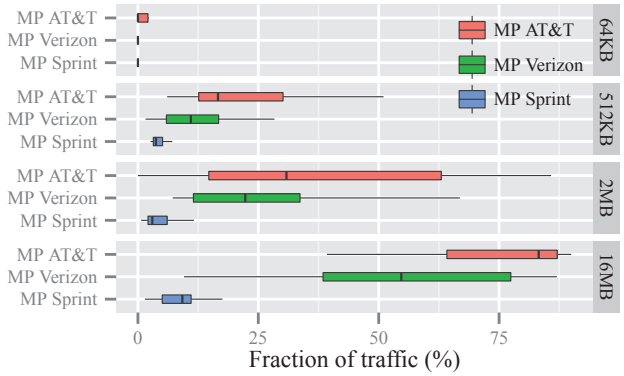

Figure 3: Baseline: faction of traffic carried by each cellular carrier in MPTCP connections.

Download time: As our goal is to understand how much gain mobile users obtain from using MPTCP, for both small flows and large flows, we focus on measuring the download time rather than the bandwidth and speed of each cellular technology. We define the download time as the duration from when the client sends out the *first SYN* to the server to the time it receives the *last data packet* from the server. We measure download time of a file using MPTCP and compare it with what we get if we use a single-path TCP over the available WiFi or 3G/4G paths.

Loss rate: The loss rate is measured on a per-subflow basis. It is calculated as the total number of retransmitted data packets divided by the total number of data packets sent by the server on the flow. We show the average loss rate by aggregating all the measurement results of the same configuration.

Round trip time (RTT): We measure RTTs on a per-subflow basis. Denote by T_r the server's receive time of an ACK packet for the previous packet sent from the server at time T_s over a subflow. RTT is measured as the difference between the time when a packet is sent by the server to the time the ACK for that packet is received (i.e., $RTT = T_r - T_s$), such that the ACK number is larger than the last sequence number of the packet and the packet is not a retransmission [31].

Out-of-order delay: MPTCP maintains two sequence numbers for each packet, a data (global) sequence number for the MPTCP connection and a subflow (local) sequence number for each TCP subflow. In-order packets arriving from the same subflow may wait in the receive buffer before their data sequence numbers become in-order. This could be due to late arrivals of packets from other paths. Therefore, a key performance metric of using MPTCP is to measure packet out-of-order delay at the receive buffer before packets are ready for delivery to the application layer. Out-of-order delay is defined to be the time difference between when a packet arrives at the receive buffer to when its data sequence number is in-order.

4. BASELINE MEASUREMENTS

Figure 2 presents the download times of different size files over different WiFi/cellular carriers using single-path or MPTCP. We show results for file sizes of 64 KB, 512 KB, 2 MB, and 16 MB. We perform our measurements over each of these four time periods in a day described in Section 3.2 and show the aggregate results in Figure 2. We use the default coupled congestion controller as the congestion control

Table 2: Baseline path characteristics: loss rates and RTTs (sample mean ± standard error) of single-path TCP on a per connection basis across file sizes. Note that Sprint has a particularly high loss rate on 512 KB downloads. Note that ∼ represents for negligible values (< 0.03%).

	File size			
	64 KB	512 KB	2 MB	16 MB
Loss (%)				
AT&T	0·03±0.03	0·04±0.01	0·06±0.03	0·31±0.12
Verizon	∼	∼	0·31±0.13	1·75±0.20
Sprint	0·37±0.16	8·76±4.8	3·93±0.34	1·64±0.01
Comcast	0·43±0.16	0·20±0.04	2·02±0.42	0·68±0.07
RTT(ms)				
AT&T	70·06±2.78	104·89±3.32	138·20±5.09	126·01±5.37
Verizon	92·41±13.23	204·65±20.45	422·56±28.34	624·66±54.55
Sprint	381·29±50.80	972·4 ±84.08	1209·81±178.68	703·81±81.96
Comcast	26·81±0.43	53·08±2.20	56·83±5.71	32·65±2.05

algorithm. We use box and whisker plots to summarize our measurement results. The line inside each box is the second quantile (median), the top and the bottom of each box are the first and the third quantiles (25% and 75%), and the ends of the whiskers are the minimum and maximum values.

MP-carrier refers to a 2-path MPTCP connection using a particular 3G/4G cellular network and a WiFi network. *SP-carrier* refers to a single-path TCP connection over a particular WiFi/3G/4G network.

For all file sizes, we observe that the download times for a file using MPTCP is almost the same as those using the best single-path TCP connection available to the user. Sometimes MPTCP outperforms the best path alone. MPTCP initiates the connection over using the WiFi network (i.e., the WiFi path is the default path).

For small flows, i.e., file sizes of 64 KB or smaller, single-path TCP over WiFi performs the best, and MPTCP does not provide much gain over using the cellular path. This is because WiFi connections have smaller RTTs (around 30 ms) than the 3G/4G cellular connections (60-80 ms for 4G, and 300 ms for 3G). Thus, in most small flow cases the file transfer is complete before the cellular paths can complete their 3-way handshakes. For slightly larger flows, we observe that single-path over WiFi is no longer guaranteed to be the best path (in terms of download times). Instead, single-path TCP over 4G LTE is the best choice in many instances. This is because, as can be seen in Table 2, the cellular networks (especially the 4G LTE networks) provide almost loss free paths, as opposed to WiFi's roughly 1.6% loss rate. Figure 3 shows the fraction of traffic offloaded to the cellular path from the data in Figure 2. We observe that MPTCP manages to offload traffic from the *fast but lossy* WiFi paths to the *not-so-fast but loss-free* cellular paths. Therefore, when the file size is not too small, MPTCP connections gain more by leveraging its cellular paths. Table 2 provides the loss rates and RTTs (averages and standard errors) for the measurements in Figure 2.

We observe that 3G networks tend to have slightly higher loss rates than 4G, much larger minimum RTTs (200 ms), and severe RTT variations (300-800 ms). Thus, for small flows, most packets in MPTCP-Sprint connections are delivered via WiFi. When the file sizes are large and a fraction of packets have initially been scheduled through the 3G path, it takes much longer for those packets to reach the client. In the case where the RTT variation is large over 3G paths (up to 8-10 times greater than its 3-way handshake RTT), and a packet is identified as lost and retransmitted, it can take a few seconds for a packet to be delivered and results in reduced performance. Section 5.2 analyzes this out-of-order delay in more detail.

In the rest of this section, we provide a more detailed analysis of the performance of MPTCP using different congestion controllers and different file sizes. For simplicity, we focus on one cellular carrier, AT&T 4G LTE, since it exhibits the lowest RTT variability and the most stable performance. We also utilize different WiFi networks at different locations.

4.1 Small Flow Measurements

We start by analyzing MPTCP measurements using small flows. We chose four different file sizes here (8 KB, 64 KB, 512 KB, and 4 MB) as representative of small flow measurements. For simplicity, we focus on one cellular carrier, AT&T 4G LTE, with Comcast WiFi as the default path. Our goal is 1) to understand how 2-path MPTCP performs in the wild, and 2) to understand the impact of different MPTCP congestion controllers on connection performance. For comparison purposes, we also seek 3) to understand how much more one can get when having 4-path MPTCP instead of 2-path MPTCP in the context of small file downloads.

Figure 4 shows our measurements of small flows. MP-4 and MP-2 represent MPTCP connections consisting of four and two subflows, while the congestion controller in parentheses indicates which congestion controller is used at the server. As an overview of baseline small flow measurements, a clear trend is, when file size increases, 4-path MPTCP performs better than 2-path MPTCP, which performs better than single-path TCP.

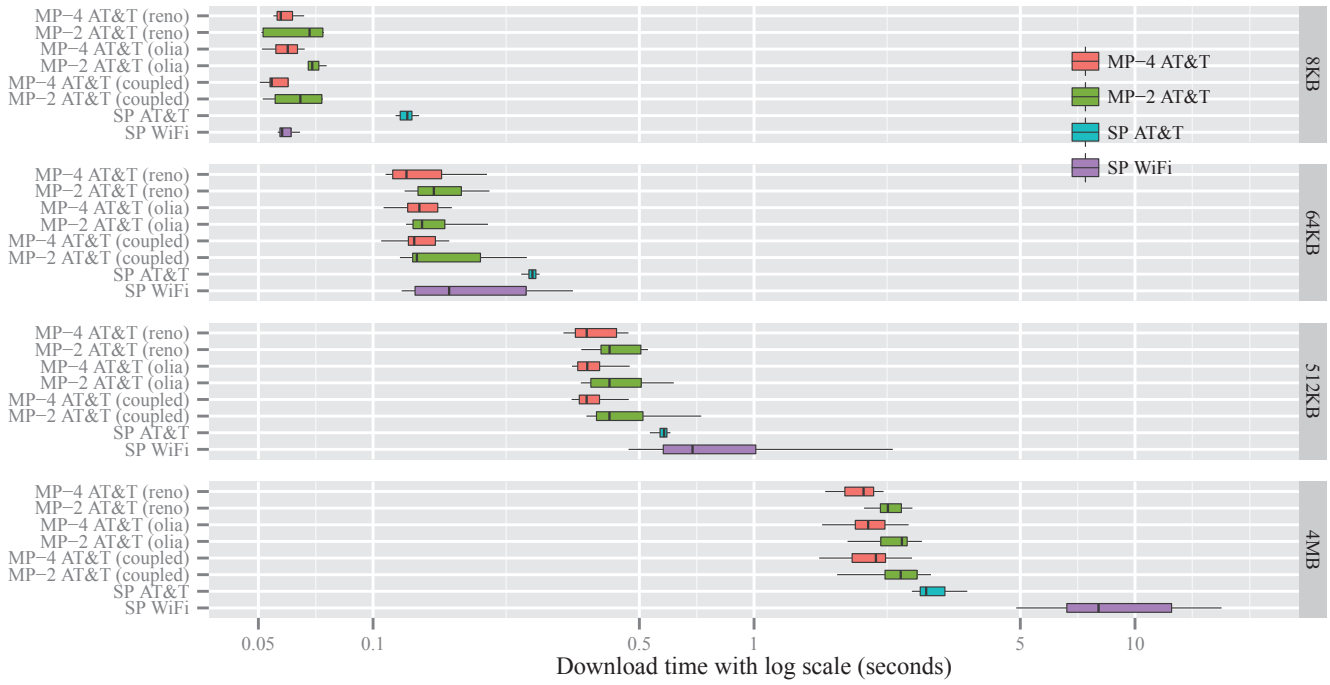

Figure 4: Small Flow Download Time: MP-4 and MP-2 represent for 4-path and 2-path MPTCP connections, and reno represents uncoupled New Reno multi-path TCP connections.

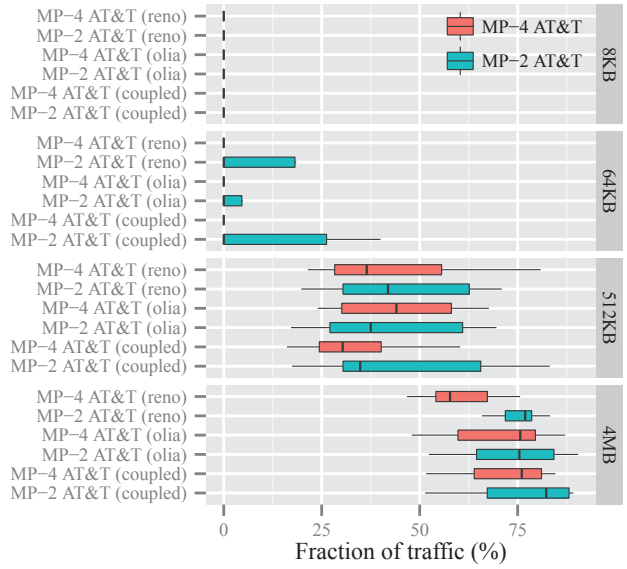

Figure 5: Small Flows: fraction of traffic carried by the cellular path for different file sizes.

4.1.1 Results at a glance

From our observations, in the case of single-path TCP, AT&T performs the worst when the file size is small (e.g., 8 KB). This is because the 4G network has a much larger minimum RTT, and the file download time over single-path WiFi is smaller than 4G's RTT of 60 ms (see Table 3). Hence, when the file sizes are as small as 8 KB, MPTCP can perform just as well as single-path TCP over WiFi (SP WiFi), regardless of the number of subflows - as most of the subflows are not utilized. Figure 5 presents the fraction of traffic carried by the cellular path in MPTCP connections over different file sizes. For file sizes smaller than 64 KB, 4-path MPTCP never utilizes the cellular path to deliver traffic, while 2-path MPTCP occasionally utilizes the cellular path.

For 4-path MPTCP, since both WiFi subflows have RTTs one half or one third of those of the cellular subflows, the two WiFi subflows can quickly complete the download of 64 KB within 2 RTTs (when no loss occurs), and the file transaction completes before the cellular paths are able to contribute. Given that the WiFi paths exhibit roughly 1.6% loss rates, in the 64 KB single-path TCP case, when a loss occurs, the cellular subflow of the 2-path MPTCP connection is able to carry some traffic.

When the flow size increases to 512 KB, we observe that WiFi is no longer the best path. Its download time is slightly larger than that of single-path TCP over AT&T LTE and has high variability. Although WiFi is characterized by small RTTs, it exhibits much larger loss rates compared to the cellular network, as shown in Table 3. When the download time spans several RTTs and the cellular path is able to contribute, the fraction of traffic carried by the cellular subflow(s) surpasses that of the WiFi flow(s). In Figure 5, we see a clear trend that the fraction of packets carried by the cellular flows reaches 50% and starts to dominate the packet delivery when the file size is 4MB. Note that by replacing the WiFi AP with a newer standard, such as 802.11n, the WiFi loss rates can be reduced because of more advanced technologies. In separate measurements, the flow loss rate of 802.11n WiFi home network is reduced but still much larger than that exhibited by cellular.

Effect of subflow number.

For each file size, we see a clear trend that 4-path MPTCP outperforms 2-path MPTCP. This result is more prominent as the file size increases. The main reason is that when a MPTCP connection starts four subflows for small file downloads (suppose all the subflows are utilized and no loss occurs), all subflows can still be in their slow-start phases before the download is complete. Therefore, the 4-path MPCP for small file transfers in principal leverages 4 slow-start phases simultaneously to fetch the one file. This may cause some fairness issues for other users sharing the same bottlenecks as MPTCP subflows.

Effect of congestion controllers.

In terms of different MPTCP congestion controllers, we do not see much difference between coupled, olia, and reno for small flows (except for 4 MB). This is likely due to the fact that most of time the connection terminates during the slow-start phase(s) if no loss occurs and the congestion controllers do not begin to operate.

Effect of background traffic.

Figure 6 shows the measurement results performed in a public WiFi hotspot offered by a coffee shop in downtown Amherst on a Friday afternoon, where the traffic load is high over the WiFi path, and we also used WiFi as the default path. During the measurements, there were on average 15 to 20 customers connecting to the WiFi hotspot with their laptops, iPads, and smart phones. For the sake of time, we did not measure the performance of olia. We observe from the results that (1): WiFi is unreliable and does not always provide the best path, (2): MPTCP performs close to the best available path. Figure 12 depicts the fraction of traffic carried over the cellular path in MPTCP connections for different file sizes. Compared to the previous results (Figure 5), we observe that more traffic is transmitted over the cellular network. This is because the WiFi path is very unreliable and lossy and, hence, MPTCP offloads the traffic to the more reliable cellular connection. These results show that MPTCP performs resonably well even in an extreme situation. Note that for 8 KB file size, we observe that MPTCP performs better than single-path TCP over WiFi even if MPTCP sends no traffic over cellular. This is because the WiFi path exhibits very large RTT variability and we did not have enough measurement samples to provide statistically meaningful results for the 8 KB case. Table 4 shows the average loss rates and RTTs over WiFi and AT&T connections.

4.1.2 Simultaneous SYNs

Current MPTCP implementations require a first flow to be established for information exchange (i.e., sender/client key and interface information) before adding a second flow. The approach of delaying the SYN packet for the second flow exhibits the following benefits: 1) it is safe and easier to fall back on legacy TCP if the other end does not speak MPTCP, and 2) it provides a higher level of connection security with key exchange. However, if the servers are known to be MPTCP-capable and the connections have been authorized, this delayed-SYN procedure postpones the usage of the second path and hence increases the download time, especially for small flows.

Table 3: Small flow path characteristics: loss rates and RTTs (sample mean ±standard error) for single-path TCP connections. Note that ∼ represents for negligible values (< 0.03%).

	File size			
	8 KB	64 KB	512 KB	4 MB
Loss(%)				
WiFi	1·0±0.5	1·6±0.4	1·4±0.2	2·1±0.2
AT&T	∼	∼	∼	∼
RTT(ms)				
WiFi	22·3±0.2	38·7±6.9	33·9±2.7	23·9±0.3
AT&T	60·8±0.5	64·9±0.5	73·2±2.1	140·9±1.1

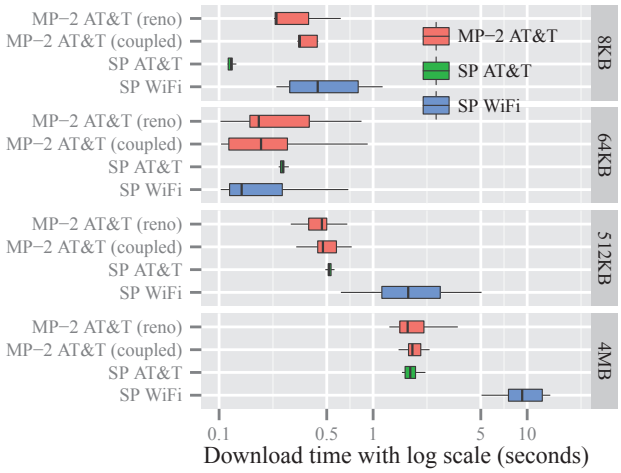

Figure 6: Amherst coffee shop: public free WiFi through Comcast business network.

Table 4: Path characteristics of Amherst coffee shop: cellular network and public WiFi hotspot. Note that ∼ represents for negligible values (< 0.03%).

	File size			
	8 KB	64 KB	512 KB	4 MB
Loss(%)				
WiFi	5·3±1.6	3·1±0.6	4·1±0.3	2·9±0.4
AT&T	∼	∼	∼	0·1±0.1
RTT (ms)				
WiFi	44·2±7.0	26·0±1.8	21·9±0.5	21·3±0.4
AT&T	62·4±0.6	63·4±0.4	61·4±0.4	80·8±1.8

For performance purposes, we modify the current MPTCP implementation to allow the client to send SYN packets simultaneously over each of its available paths to the server. In principle, this can allow the user to establish both its paths simultaneously and will reduce the download time of the file. This can also improve the performance of MPTCP

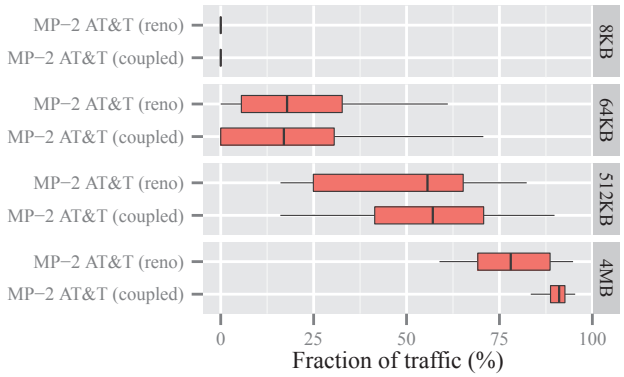

Figure 7: Amherst coffee shop: fraction of traffic carried by the cellular path. With MPTCP-coupled and uncoupled New Reno TCPs, where MPTCP is in favor of the cellular path when the file size increases.

in cases where the default path (WiFi in our case) is very lossy or has a large RTT.

Figure 8 shows that based on our measurements, even with large average RTT ratios, the simultaneous-SYN MPTCP on average reduces the download time by 14% for 512 KB files and 5% for 2 MB files, respectively. There could be even greater benefit if the RTTs of the paths are similar, especially for small downloads. Note that simultaneous SYN and delayed SYN might not differ much for very small size files since most of the packets can be delivered through the first path (as the initial congestion window is 10 packets).

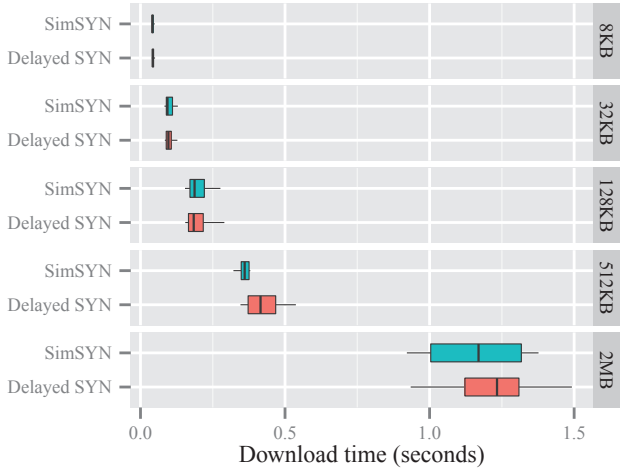

Figure 8: Small Flows: download time for simultaneous SYN and the default delayed SYN approach.

4.2 Large Flow Measurements

In this section, we present results for larger file sizes (e.g., 8 MB, 16 MB, and 32 MB). For comparison purposes, we also include 4 MB downloads with the other three large file sizes made during the day of our measurements. Our goal is to evaluate the behavior of MPTCP when subflows leave their slow start phases, and the MPTCP congestion controller takes over the connection and performs congestion control with load balancing. Our results show how current MPTCP congestion controllers (coupled and olia) perform in the wild, rather than in the environments where most of the traffic is well-controlled [19, 32]. We compare the results to a baseline where we use uncoupled New Reno (reno) as the congestion controller.

Figure 9 presents the results. We observe that: (1) WiFi is no longer the best path and MPTCP always outperforms the best single-path TCP, (2) 4-path MPTCP always outperforms its 2-path counterpart, (3) MPTCP-olia consistently performs slightly better than MPTCP-coupled. In particular, we observe that MPTCP-olia performs similarly to MPTCP-coupled for file size of 4 MB, and reduces the download latencies of files of sizes 8 MB, 16 MB, and 32 MB by 5%, 6%, and 10%, respectively, in both 2-path and 4-path scenarios). TCP New Reno performs better because it is more aggressive and not fair to other users. In contrast, olia's better performance (compared to coupled) is due to its better load balancing in the network [19].

Figure 10 shows that in all configurations, over 50% of traffic is now routed through the cellular path instead of WiFi. This is because, in large flow downloads, the cellular path's very low loss rate compensates for its much larger RTTs. Table 5 lists the RTTs and loss rates seen by the subflows on a per connection average. We see from this table that WiFi loss rates varies from 1.6% to 2.1%, while 4G LTE provides very consistent and low loss rate of 0.01%, and the per connection average RTTs are more stable (i.e., have much lower variability).

To exclude the possibility that the 4-path performance gain is due solely to the benefits of having multiple slow-start phases, we also performed measurements of transferring extremely large files of size 512 MB separately to approximate infinite backlog traffic. We performed the measurements for 2-path and 4-path MPTCP using coupled and uncoupled New Reno as congestion controller with 10 iterations each (results of olia are omitted for lack of space). Figure 11 shows that the download time is around 6-7 minutes, hence the effect of slow starts should be negligible. The results of 4-path MPTCP confirms the results in Figure 9 as we observe that 4-path MPTCP slightly outperforms 2-path MPTCP.

Table 5: Large flow path characteristics: loss rates and RTTs (sample mean ±standard error) of single-path TCP on per connection average. Note that ∼ represents for negligible values (< 0.03%).

	File size			
	4 MB	8 MB	16 MB	32 MB
Loss(%)				
WiFi	2·1±0.4	1·6±0.3	1·9±0.3	2·0±0.3
AT&T	0·1±0.1	∼	∼	∼
RTT(ms)				
WiFi	26·2±0.9	25·9±0.5	24·9±0.4	23·5±0.3
AT&T	133·1±4.4	154·5±2.7	144·5±4.1	146·4±4.3

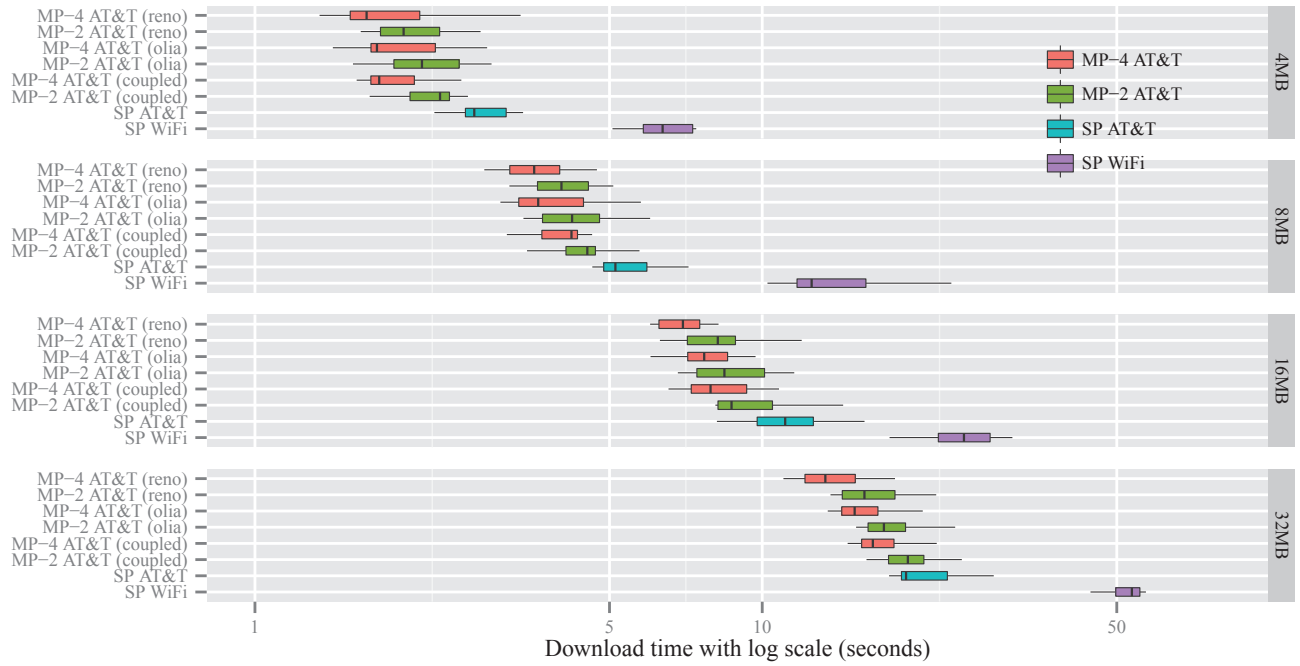

Figure 9: Large Flow Download Time: MP-4 and MP-2 represent for 4-path and 2-path MPTCP connections, and reno represents uncoupled New Reno multi-path TCP connections.

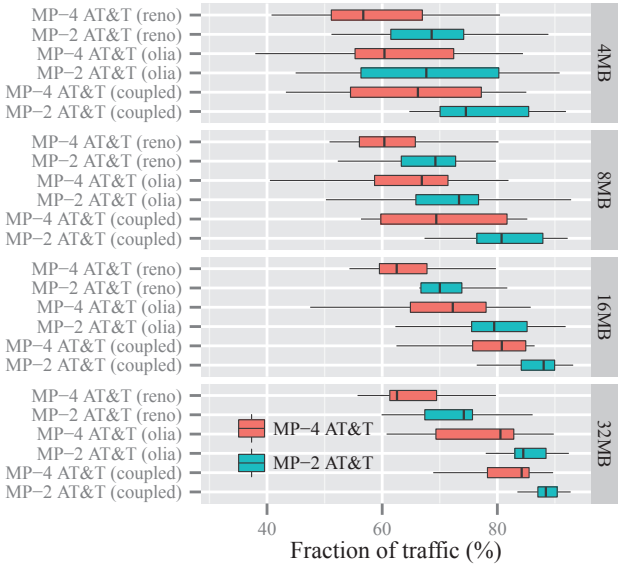

Figure 10: Large Flows: fraction of traffic carried by the cellular path for different file sizes.

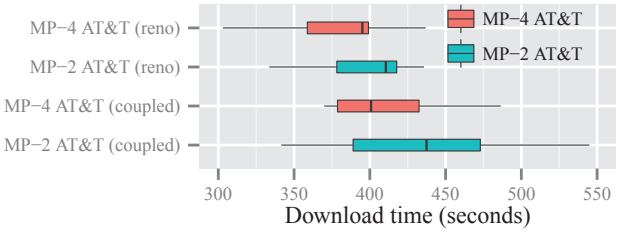

Figure 11: Large Flows: download time of infinite backlog (file size 512 MB) for uncoupled New Reno/coupled MPTCP connections with four/two flows.

5. LATENCY DISTRIBUTION

In previous sections, we focused mainly on the performance of MPTCP in terms of download times. For mobile users, however, low download time does not necessarily guarantee a high quality of experience. When using the Internet, users do more than simply fetching and viewing Web pages. Users often consume real-time applications, such as video streaming (e.g., Youtube, Netflix) or interactive services (e.g., Facetime, Skype). These applications require stable network service, i.e., low variability and jitter.

Although MPTCP provides robustness against time-varying path quality, the impact to MPTCP connection quality remains unclear when another cellular/WiFi path is being exploited for MPTCP. In the following sections, we first characterize path latency (in terms of packet round trip times) of each cellular carrier and the Internet service provider, and try to understand the impact of using heterogeneous networks. More importantly, we investigate how leveraging path diversity might introduce latency to application performance, which can directly affect user experience.

5.1 Packet Round Trip Times

We reported average RTTs (and their standard errors) of single-path TCP connections over cellular and WiFi paths as indications of path quality in previous sections. Here, we investigate RTTs at a finer granularity by focusing on the distributions of packet RTT for each file download size, and

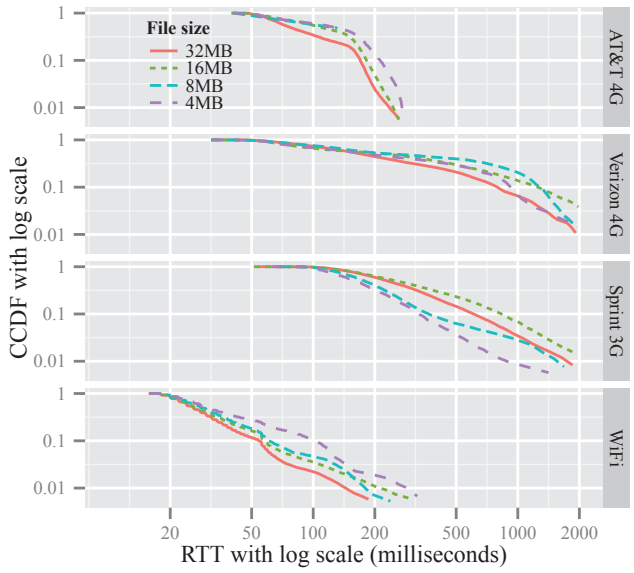

Figure 12: Packet RTT distributions of MPTCP connections using WiFi and one of the three cellular paths.

the RTT measurements are on a per-flow basis. The RTT is calculated as defined in Section 3.3. For each MPTCP connection, we record the RTT value of each packet if an ACK is received by the server for a particular packet, excluding retransmitted and timed out packets.

Note that the RTT traces are collected from the measurements described in Section 4, where the default coupled congestion controller is used. We then aggregate all the packet RTT traces over the course of 24 hours, and group them by interfaces (cellular and WiFi) and file sizes. In addition, we only report on flow sizes larger than 512KB, as some carriers have large RTTs and hence the cellular path does not carry any traffic when file sizes are smaller than 512KB.

Figure 12 presents the Complementary CDF (CCDF) plot of flow RTTs for different transfer sizes carried via different cellular/WiFi providers across all MPTCP connections. Note the the figure is in log-log scale to better visualize the tails.

Two clear behaviours are observed here. The WiFi path, on average, has lower and less variable RTTs than cellular paths. The minimum WiFi RTTs across different file sizes are about 15 ms, while 90% of packet RTTs are less than 50 ms for file sizes larger than 4 MB.

Cellular networks, on the other hand, have quite different RTT patterns than the WiFi network. The AT&T LTE path exhibits a minimum RTT of about 40 ms, and more than 70% of the RTT samples lie between 50 and 200 ms. The Sprint 3G network, on the other hand, has a minimum RTT of about 50 ms, but with more than 98% of the RTT samples larger than 100 ms, and the value could five-fold large when file sizes are 4-8 MB. If the transferred file size is as large as 16 or 32 MB, packet RTTs can be as large as 2 seconds.

Despite being based on LTE, the Verizon network, has an RTT distribution pattern that lies in between the patterns of both AT&T and Sprint. Its minimum RTT is 32 ms, which is smaller than AT&T's, but the RTT value can extend up to two seconds.

In all, packet RTTs over cellular networks have quite different patterns than conventional WiFi networks. Cellular networks exhibit larger minimum RTTs and higher RTT variability. The phenomenon of having inflated and varying RTTs over cellular networks is commonly termed as *bufferbloat* [9], and the root cause of this issue is the presence of huge buffers in the networks (routers at edge networks or in the cellular networks). Our measurements confirm results from previous studies by Allman [1] and Jiang et al. [15], which show that bufferbloat is less prominent in residential/non-residential networks (ex: private/public WiFi networks), and can be very severe in 3G/4G cellular networks.

When a MPTCP connection includes a path that has highly variable RTTs, that path can affect the overall MPTCP performance. This is mainly because for large RTT values, if the RTT values increases over time, it takes longer for the MPTCP congestion controller to update its estimated RTT and will delay the congestion controller's response to the large latency. The MPTCP congestion controller can hence underestimate the targeted throughput and lead to performance degradation. Since this issue is more related to path characteristics, we leave this for future work.

5.2 Out-of-order Delay

Our results in Section 4 show that MPTCP performs comparably to its best single-path TCP counterparts over any of the available paths, and sometimes performs slightly better. We measured the download time of a file and showed the results for different file sizes. However, in practice many applications are sensitive to the network quality (e.g., low RTT or jitter variation) rather than download time or throughput (as long as it satisfies the operational conditions). When the path characteristics (e.g., loss rate or RTT) are diverse, reordering delay becomes crucial as packets arriving early from one path need to wait for packets arriving late from another path. From our measurements, this happens very often when the paths have very different RTTs. In this case, the fraction of the traffic carried by the slow path (e.g., a 3G path) is very small, while the majority of packets arrive over the fast path, but are out-of-order in data sequence number. These packets arrive at the receive buffer as a burst, but will not be delivered to the application until the packets arrive from the slow path. In our testbed, the receive buffer is configured to be large enough so that there is no limitation due to the receive window, and thus we can measure the exact delay caused by reordering.

Figure 13 shows CCDFs of out-of-order delay using three different MPTCP configurations: AT&T/WiFi, Verizon/WiFi, and Sprint/WiFi. Note the time axis in the figure is in log scale so as to better visualize the tail. Table 6 shows the average and standard errors for RTTs and out-of-order delay of MPTCP connections.

MPTCP with AT&T 4G, and MPTCP with Verizon 4G in general do not suffer much from out-of-order packets. 75% of the packets are delivered in order (in terms of global data sequence numbers). However, transfers of of smaller files (4MB and 8MB) tend to exhibit larger out-of-order delay. This might be explained by their RTT distributions, where 4MB and 8MB flows tend to have higher RTTs. Thus, when a packet is out-of-order, it needs to wait for the later arriv-

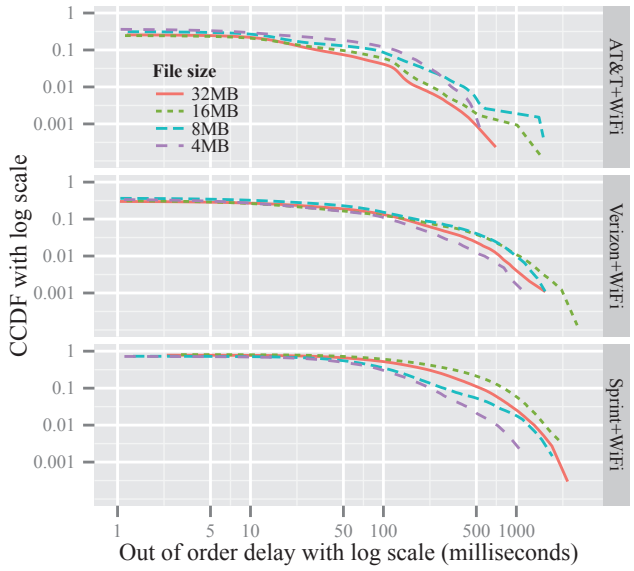

Figure 13: Out-of-order delay distributions of MPTCP connections using WiFi and one of the three cellular paths.

ing packets from the slow path (in this case, the cellular network).

MPTCP with Sprint 3G exhibits a different pattern. 75% of the packets are out-of-order when they arrive at the receive buffer. Note that the out-of-order delay might not be very important for user's Web browsing, but it is significant in the context of real-time traffic. For example, in Facetime or Skype, the maximum tolerable end-to-end latency is considered to be about 150 ms (one-way network delay plus the out-of-order delay). Here, we see more than 20% of the packets have out-of-order delay larger than 150 ms, even without including the one-way network delay. That is, given that Sprint 3G's average RTT is about 200 ms, if we consider the one-way delay to be half of the RTT, its overall end-to-end delay (prior to be available to associated application) is $(200/2)+100 = 200$ ms, which is much larger than the duration that most modern real time applications can tolerate.

6. DISCUSSION

As mobile devices and smart phones are now equipped with two interfaces (WiFi and 3G/4G), they provide natural platforms on which to use MPTCP. We have shown how applicable MPTCP is for mobile devices where multiple paths are available. We demonstrated the performance of MPTCP on file transfers of small and large flows, from 8 KB to 32 MB.

Web traffic contributes a large fraction of today's Internet traffic [4, 20], and cellular networks have also experienced tremendous HTTP traffic growth from mobile devices [5]. Although it has been reported that most Web traffic to mobile devices are flows smaller than 1 MB to 2 MB [6], online video streaming contributes the majority of the traffic to mobile devices [5], which has long been thought of as downloading a large single object from the server.

Table 6: Statistics on MPTCP RTT (flow mean ±standard errors) and out-of-order (OFO) delay (connection mean ±standard errors) over different carriers.

	File size			
	4 MB	8 MB	16 MB	32 MB
RTT(ms)				
AT&T	110.0±6.7	102.0±6.4	114.1±7.5	99.6±6.5
Verizon	228.0±26.9	399.2±46.1	360.4±44.3	296.1±31.7
Sprint	202.9±14.4	262.5±24.6	480.4±40.6	346.2±28.1
WiFi	56.2±6.7	43.4±6.4	29.4±7.5	30.0±6.5
OFO(ms)				
AT&T	30.9±3.2	26.8±4.0	16.7±1.4	13.1±1.6
Verizon	36.7±5.6	67.7±11.7	61.3±12.9	50.2±8.9
Sprint	91.3±12.6	126.9±29.9	301.7±44.3	204.5±29.8

A previous study [27] shows that, for modern online video streaming applications, such as Youtube or Netflix, transfers usually begin with a prefetching/buffering phase consisting of a large data download, followed by a sequence of periodic smaller data downloads. Table 7 summarizes the measurements we performed on two popular mobile devices when playing Netflix movies, whereas Youtube in general prefetches less aggressively by 10MB to 15MB and transfers blocks periodically of size 64 KB and 512 KB.

Table 7: Summary of Netflix video streaming

	Prefetch (MB)	Block (MB)	Period (sec)
Android	40·6±0.9	5·2±0.2	72·0±10.1
iPad	15·0±2.6	1·8±0.5	10·2±2.7

Our MPTCP measurements shed light on how MPTCP can be utilized not only for Web browsing, but also for online video streaming. We have demonstrated the utility of MPTCP for conventional Web object downloads by our small flow measurements. We show that small flows benefit from using MPTCP with multiple slow starts and by using multiple flows. When the file size is really small, say 8KB or 16KB, a fewer than a dozen of packets are required, which can be easily transmitted through the first flow within one or two RTTs. In this case, MPTCP behaves like single-path TCP and does not harm other TCP users.

In the future, when online video streaming servers are MPTCP-capable, our measurements provide some insights for understanding how well the long prefetching process and the short periodic transfers can be achieved. Furthermore, it can greatly reduce the download time without having the viewers waiting for too long and breaking the connection, even though they are mobile.

In the context of mobility, when using single-path TCP, users move from one access point to another, changing their IP address and forcing the on-going connections to be either stalled or reset. In addition, all the previously downloaded data in the stalled connections not yet delivered to the application would be wasted. In contrast, MPTCP not only

leverages multiple paths simultaneously and performs traffic offloading on the fly. It also provides robust data transport in a dynamically changing environment and can support mobility without wasting bandwidth in reset connections.

An alternative to MPTCP is to identify the best network among all available ones, and maintain a single flow over that network without worrying switching among them. We argue against this option because it could be very costly or almost impossible to decide which network is the real winning network as it depends on the loss rates and RTTs over each path, as well as the file sizes. Most of this information is not available a priori at the client, and the loss rates and RTTs can also vary over time. MPTCP, on the other hand, has been shown to be responsive to changes in the networks by performing congestion balancing across different paths/networks [19, 32] and can use the best path without any of this information in advance.

Finally, as one benefits from using MPTCP by utilizing an additional interface, a natural question is energy consumption. By adding another cellular path to an MPTCP connection, there will be an additional energy cost for activating and using the antenna. We have ported the current Linux MPTCP kernel to Android phones so as to better understand the relationship between the desired MPTCP performance gain and the additional energy cost. We leave this as future work.

7. RELATED WORK

MPTCP is a set of extensions to regular TCP, which allows users to spread their traffic across potentially disjoint paths [8]. The general design of MPTCP has been inspired by the early work of Han et al. [10] and Kelly & Voice [16] that developed theoretically grounded controllers for a multipath transport protocol. Numerous studies have recently been published that discuss performance issues with current MPTCP implementations. These studies have resulted in a number of changes in the congestion controller [14, 19, 32] in an attempt to provide better fairness and throughput.

Although MPTCP is being standardized by IETF, little is understood about how well it performs in dynamic environments such as wireless networks. Raiciu et al. [23, 32] showed that MPTCP outperforms standard TCP when path diversity is available in a data center network as well as in very simple wireless settings. Paasch et al. [22] studied mobile/WiFi handover performance with MPTCP. The authors investigated the impact of handover on MPTCP connections using different modes such as full-MPTCP mode (where all potential subflows are used to transmit packets) and backup mode (where only a subset of subflows are used). They showed that MPTCP can utilize other available subflows when WiFi is disconnected but did not explore how quickly MPTCP can re-use re-established WiFi. In [25], Raiciu et al. also studied mobility with MPTCP. They examined a mobile MPTCP architecture consisting of a mobile host, an optional MPTCP proxy, and a remote host. While it shows MPTCP outperforms standard TCP in a mobile scenario, it does not examine full end-to-end MPTCP or the delayed re-use problem.

All these studies have ignored the effect of multi-path on finite size flows. Moreover, they have studied the performance of MPTCP through analysis, by simulations, or by measurement in environments where all the traffic is well controlled. In contrast, we study the performance of MPTCP in the wild, with real wireless settings and background traffic, and focuses on finite size data objects that better represent real world traffic.

8. SUMMARY AND CONCLUSION

In this paper, we reported latency measurements made for different file sizes using multi-path over WiFi and one of three different cellular providers, and compared them to the latencies using only one of either the WiFi or cellular provider. Two of the providers use LTE, and for these we observed the latencies are smaller using them exclusively except for very small files. The third provider uses a CDMA-based 3G technology and we find that using WiFi significantly reduces download latency. However, in all cases, MPTCP generates latencies that are comparable to or nearly comparable to the smallest latency produced by either WiFi or cellular. We also studied how latencies are affected by load on the WiFi path, the congestion controller design in MPTCP, the number of paths, and whether data flows are started simultaneously or in a staggered manner (as stipulated by MPTCP). In all, we conclude from our results that MPTCP provides a robust data transport and reduces the variability in download latencies.

9. ACKNOWLEDGEMENTS

This research was sponsored by US Army Research laboratory and the UK Ministry of Defense under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This material is also based upon work supported by the National Science Foundation under Grant No. CNS-1040781 and was supported in part by the EU project CHANGE (FP7-ICT-257422).

10. REFERENCES

[1] M. Allman. Comments on bufferbloat. *ACM SIGCOMM Computer Communication Review*, 43(1):30–37, 2012.

[2] M. Allman, V. Paxson, and E. Blanton. RFC 5681: TCP congestion control, 2009.

[3] M. C. Chan and R. Ramjee. TCP/IP performance over 3G wireless links with rate and delay variation. *Wireless Networks*, 11(1-2):81–97, 2005.

[4] J. Erman, A. Gerber, M. T. Hajiaghayi, D. Pei, and O. Spatscheck. Network-aware forward caching. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, 2009.

[5] J. Erman, A. Gerber, K. Ramadrishnan, S. Sen, and O. Spatscheck. Over the top video: the gorilla in cellular networks. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet Measurement Conference (IMC)*, 2011.

[6] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM*

SIGCOMM conference on Internet Measurement Conference (IMC), 2010.

[7] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. RFC 2883: An extension to the selective acknowledgement (SACK) option for TCP, 2000.

[8] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. RFC 6824: TCP extensions for multipath operation with multiple addresses, 2013.

[9] J. Gettys and K. Nichols. Bufferbloat: Dark buffers in the Internet. *Communications of the ACM*, 55(1):57–65, 2012.

[10] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity in the Internet. *IEEE/ACM Transactions on Networking*, 14:1260–1271, December 2006.

[11] J. Huang, Q. Feng, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012.

[12] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl. Anatomizing application performance differences on smartphones. In *Proceedings of the 8th international Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2010.

[13] P. Hurtig and A. Brunstrom. Enhanced metric caching for short TCP flows. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2012.

[14] B. Jiang, Y. Cai, and D. Towsley. On the resource utilization and traffic distribution of multipath transmission control. *Perform. Eval.*, 68(11):1175–1192, Nov. 2011.

[15] H. Jiang, Y. Wang, K. Lee, and I. Rhee. Tackling bufferbloat in 3G/4G networks. In *Proceedings of the 2012 ACM SIGCOMM conference on Internet Measurement Conference (IMC)*, 2012.

[16] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. *SIGCOMM Computer Communications Review*, 35(2):5–12, Apr. 2005.

[17] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.

[18] P. Key, L. Massoulié, and D. Towsley. Combining multipath routing and congestion control for robustness. In *Proceedings of the 40th IEEE Conference on Information Sciences and Systems (CISS)*, 2006.

[19] 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 *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2012.

[20] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband Internet traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet Measurement Conference (IMC)*, 2009.

[21] MultiPath TCP Linux kernel implementation. http://mptcp.info.ucl.ac.be/.

[22] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring mobile/WiFi handover with multipath TCP. In *Proceedings of ACM SIGCOMM Workshop on Cellular Networks (CellNet)*, 2012.

[23] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath TCP. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 266–277, 2011.

[24] C. Raiciu, M. Handly, and D. Wischik. RFC 6356: Coupled congestion control for multipath transport protocols, 2011.

[25] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley. Opportunistic mobility with multipath TCP. In *Proceedings of the 6th ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*, 2011.

[26] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? Designing and implementing a deployable multipath TCP. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2012.

[27] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network characteristics of video streaming traffic. In *Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2011.

[28] P. Sarolahti and A. Kuznetsov. Congestion control in Linux TCP. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2002.

[29] V. Shrivastava, S. Rayanchu, J. Yoonj, and S. Banerjee. 802.11n under the microscope. In *Proceedings of the 8th ACM SIGCOMM conference on Internet Measurement Conference (IMC)*, 2008.

[30] tcpdump. http://www.tcpdump.org.

[31] tcptrace. http://www.tcptrace.org.

[32] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *Proceedings of the 8th USENIX conference on Networked Systems Design and Implementation (NSDI)*, 2011.