Characterizing 4G and 3G Networks: Supporting Mobility with Multi-Path TCP

UMass Amherst Technical Report: UM-CS-2012-022

Yung-Chih Chen
University of Massachusetts Amherst
yungchih@cs.umass.edu

Erich M. Nahum
IBM T.J. Watson Research Center
nahum@us.ibm.com

Richard J. Gibbens
University of Cambridge
richard.gibbens@cl.cam.ac.uk

Don Towsley
University of Massachusetts Amherst
towsley@cs.umass.edu

Yeon-sup Lim
University of Massachusetts Amherst
ylim@cs.umass.edu

***Abstract*—Advances in cellular technology have increased the demand of accessing the Internet dramatically. Cellular technology not only enables mobility, but also allows redundant connectivity using multiple wireless paths to improve availability, reliability, and performance. Multiple paths enable the potential to shift traffic from broken or congested paths to higher-quality ones as traffic characteristics dynamically change, particularly during movement. However, little work has been done to date studying cellular networks or their suitability.**

This paper characterizes the behavior of cellular networks, examining 3G, 4G, and Wi-Fi networks for both single-path and multi-path data transport. Our contribution is two-fold: First, we perform measurements of single path transport using TCP over major US cellular wireless networks (both 4G and 3G), and characterize them in terms of throughput, packet loss, and round-trip time. Second, we measure and evaluate transport using multi-path TCP in cellular environments and show that leveraging path diversity under changing environments is a promising solution for more reliable and efficient TCP transfer. We also identify potential issues in using multi-path TCP which can limit performance.

I. Introduction

With the ubiquitous deployment and rapid evolution of cellular data networks, the demand of accessing the Internet for mobile users has soared dramatically. Not long after the third generation (3G) standards were released, 3G services have attracted a tremendous number of users due to the convenience of mobile devices. Although traditional Wi-Fi has a much higher transfer rate than 3G, using Wi-Fi in moving vehicles is challenging since access points (APs) usually have short signal ranges. Furthermore, the association process takes up to $10-15$ seconds and the connection quality and durations are both affected by the moving speed [6]. As a hybrid approach, using Wi-Fi to augment 3G service [4] was a smart interim solution that offloads data from ubiquitous but slow 3G to intermittent yet fast Wi-Fi networks. However, this requires constantly switching between Wi-Fi and 3G technology and often breaks the existing connection into several short, possibly delayed sessions. Since each TCP connection setup requires additional time, due to three way handshaking, slow-start, and congestion window initialization/ramp-up, splitting the existing connection into several TCP connections will introduce additional overhead.

As most of major US cellular network carriers have recently launched their 4G services in commercial markets, the status quo of integrating the technologies of 3G and Wi-Fi for better performance has changed. Here, we seek a better solution for accessing the Internet under mobile scenarios by leveraging path diversity offered by cellular networks, where a mobile user can establish TCP connections using multiple paths simultaneously. When any of the paths becomes congested or breaks, this scheme can easily offload traffic from one path to another without breaking existing TCP connections.

In this paper, we first characterize the 4G/3G networks of three major US cellular network carriers (i.e., Verizon, AT&T, and Sprint). We observe that 4G outperforms Wi-Fi and provides several times greater performance than 3G, in terms of throughput and loss rates. We also show that, by leveraging path diversity (4G, 3G, and Wi-Fi), multi-path TCP can support mobility without breaking existing connections. When passing through areas where different technologies/carriers have different availability, this solution can dynamically offload traffic from congested/broken paths to better ones, and provides a more reliable and efficient TCP transfer with performance no worse than single-path TCP.

The remainder of this paper is structured as follows. Section II briefly describes the background of current cellular data networks and multi-path TCP. Measurement setup and methodology are presented in section III. Results of 4G/3G measurements with multi-path TCP of static and mobile scenarios and the performance comparison against single-path TCP are in section IV. Related works are discussed in section V, and section VI concludes this paper.

II. Background

This section provides background of the state of art cellular data network technologies and the multi-path TCP control mechanisms needed for the rest of the paper .

A. Cellular Data Networks

In this paper, we examine the performance of 3G and 4G (corresponding to the third and fourth generation) services of Verizon, AT&T and Sprint networks, as they are the three major carriers for which we have measured. We focus on the downlink measurements as mobile clients using multi-path TCP are usually the receivers.

1) 3G and 4G Technologies: 3G is a generation of standards defined by the International Mobile Telecommunication Union (ITU). 3G services are required to satisfy the standards of providing peak data rate of at least 200K bits per second (bps), and 3G services of Verizon and Sprint are based on the Evolution-Data Optimized (EV-DO), whereas AT&T is based on High Speed Packet Access (HSPA), respectively. Note that among all of our measurements, we did not observe any area where AT&T provides 3G-only service. All of AT&T's networks are now under the enhanced version named evolved HSPA (HSPA+). Unlike HSPA, there is no natural evolution from EV-DO system to 4G LTE (described below), hence Verizon currently uses the evolved High Rate Packet Data (eHRPD) as their stopgap from 3G to 4G (instead of replacing existing EV-DO system with HSPA and then migrating to 4G LTE). For simplicity, in this paper, we refer to AT&T HSPA+ as AT&T 3.5G, Verizon eHRPD as Verizon 3G, and Sprint EV-DO as Sprint 3G.

4G differs from technologies of previous generations in that it aims to support mobility and all-Internet Protocol (all-IP) based, packet-switched communications (as apposed to circuit-switched telephony service). There are two major 4G standards: Long Term Evolution (LTE) [1] and Worldwide Interoperability for Microwave Access (WiMax) [12]. The specified peak speed in the specifications for 4G services are 100 Mbps for high mobility communication (e.g., users in vehicles), and 1Gbps for low mobility communication (e.g., stationary users or pedestrians). Both Verizon and AT&T have launched their 4G LTE services recently in some areas. Sprint, as a counter part, started its 4G service much earlier with WiMax since 2008. Although Sprint recently announced that it is entering the LTE market in 2012, due to the low availability of Sprint LTE, we only focus on Sprint WiMax and refer to it as Sprint 4G.

2) Availability: As cellular technologies have entered to our daily life, it is much easier nowadays for us to access the Internet while walking across the streets, going to another place by car, or through public transportation. In most vehicular scenarios, mobile users will experience technology switches from certain carriers due to the availability of particular cellular technologies.

Previous measurements from UMass mobility testbed [4], a vehicular network consisting of 20 public transit vehicles, has shown that 3G is available 90% of the time and Wi-Fi is around 12% of the time in Amherst Massachusetts, a geographical area of 150 square miles. Further statistics show that 4G availability (from Verizon Wireless) in the same measurement area of Amherst is about 60% of the time. Since most carriers only offer their 4G services in limited, mostly metropolitan, cities[1], the presence of 4G technology, together with 3G and Wi-Fi, has attracted our interest in the robustness of data transport when using multiple paths during movement.

When a user moves across zones of different signal coverage provided by different carriers (e.g., from a metropolitan work place to a suburban residential area), it is often the case that the mobile user will suffer instability or stallness of the existing TCP connections. With multi-path TCP, these issues can be dealt with by offloading traffic from heavily loaded paths to higher quality ones during technology transition (e.g., 4G to 3G), and from the congested/broken paths to well-performed paths (e.g., weak signal coverage) without hurting users' experience.

B. Multi-Path TCP

Consider a scenario of two hosts where each host has multiple interfaces. Multi-path TCP establishes a connection, which utilizes the paths defined by all end-to-end interface pairs. The traffic transferred over each path is referred to as a *flow*, and each path contains one flow. Hence, in this paper, we refer to a multi-path TCP connection utilizing k paths simultaneously as having k flows in it. The benefits of leveraging multi-path TCP with cellular networks in mobile scenarios is three-fold:

- **Improve throughput**:
  The most basic goal of using multi-path TCP is to improve throughput. A multi-path TCP connection should perform at least as well as a single-path TCP connection on the best of all used paths.
- **Load Balancing**:
  A multi-path TCP connection should move traffic from its mostly congested path to other paths as much as possible. A more extreme case would be when a path breaks, a multi-path TCP connection should move all the traffic from the broken path to other working paths, and maintains the connection as stable as possible.
- **Robustness for technology availability**:
  When traveling from one place to another, mobile users might receive services of certain technologies from all carriers at one place (e.g., in big cities), but only from some of them in other places (e.g., in suburban areas). Since service can die or degrade from cell to cell, data transport with multi-path TCP can still offer a quality service regardless of the poor performance from certain carriers.

In summary, the aggregate throughput of a multi-path TCP with k flows should perform at least as well as the best single-path TCP running on any of the k paths. One should treat a multi-path TCP as a single-path TCP and compare its throughout to that of the *best single-path TCP connection*, rather than to that of the aggregation of k independent single-

[1]As of June 2012, Sprint, together with its collaborator Clearwire, has offered their 4G WiMax services in 77 cities [20] across the US since 2008. Verizon Wireless, although started slightly later, has extended it 4G coverage to 258 cities, while AT&T only covers 38 of them [24] [3].

Carrier	Device Name	Technology		Frequency (MHz)		Connection Type
		4G	3G	4G	3G	
AT&T	Elevate 3/4G mobile hotspot	LTE	HSPA(+)	704-746	850/1900	USB tethering
Verizon	LTE USB modem 551L	LTE	eHRPD	746-787	800/1900	Ethernet Cable
Sprint	OverdrivePro 3/4G mobile hotspot	WiMax	EV-DO	2495-2690	1850-1990	USB tethering

TABLE I
4G DEVICES USED FOR EACH CARRIER

path TCP connections for the purposes of fairness, as multi-path TCP is designed to be TCP friendly.

In this work, we use multi-path TCP Kernel implementation [15] under Ubuntu Linux 11.10 for measurements. As a standard procedure of running multi-path TCP [8], a TCP 3-way handshake is initiated by the mobile client, with multi-path capable information placed in the option field of the SYN packet. If the server also runs multi-path TCP, it then returns corresponding information in the option field of SYN/ACK and the first flow can be established. Information about other interfaces at both ends is then exchanged through this existing flow, and additional flows will be set up through additional 3-way handshakes for those interfaces.

Each flow behaves like a regular TCP connection, and maintains its own congestion window during data transfer. Denote by w_i the congestion widow size of flow i, and w the total congestion window size over all the flows. We briefly describe the congestion control schemes used in regular single-path TCP and multi-path TCP .

1) TCP Reno Congestion Control: The standard TCP congestion control increases and decreases the congestion window with the following manner:

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

As multi-path TCP cannot use the standard TCP congestion control algorithm over each path due to resource allocation and fairness problems, we hence explain the coupled congestion control schemes in the following.

2) Coupled Congestion Control:

- For each ACK on flow i: $w_i = w_i + \frac{1}{w}$
- For each loss on flow i: $w_i = \max(w_i - \frac{w}{2}, 0)$

The congestion controller listed above is referred to as *fully coupled* because it uses the window sizes of all paths, which couples all flows' congestion windows in either the increase phase (on each ACK received) and the decrease phase (on each packet loss identified). When the controller only couples the flows' congestion windows in the increase phase (and maintains the flow window decrease mechanism from TCP Reno), we refer to it as the *coupled increase* scheme. Similarly, when coupling the flows' congestion windows only occurs during the decrease phase (and maintains flow window increase mechanism from TCP Reno), we refer to it as *coupled decrease* scheme. Note that when there is only one flow available, all three coupled congestion control algorithms reduce to the standard Reno algorithm. A further analysis [13] has investigated issues related to these schemes, and suggested that the *coupled increase* scheme should be used.

An enhanced version of the coupled increase algorithm is proposed by Wischik et al. [26], which takes into account the properties of different RTTs over different paths. Its coupled increase phase is described as follows:

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

The additional parameter, α, controls the aggressiveness of the window's increase to compensate for situations where RTTs over different paths differ widely.

In this paper, we use the multi-path TCP Kernel implementation [15], and the enhanced *coupled increase* scheme defined in [26] is used as the default congestion controller of multi-path TCP.

III. MEASUREMENTS DESCRIPTIONS

We conduct measurements over three major commercial cellular network providers in the US: Verizon, AT&T and Sprint.

The setting of our measurements consists of a wired server, residing at the University of Massachusetts Amherst (UMass) and a mobile client. Our server is Dell Precision T1600n workstation using Intel Quad Core Xeon E3-1225 CPU (3.1 GHz processor, 8GB memory) and is configured as a multi-homed host [8], connecting via 2 Intel Gigabit Ethernet interfaces to two subnets (LANs) of the UMass network. Each Ethernet interface is assigned with a public IP address and connected to the LAN via a 1 Gigabit Ethernet cable. The mobile client is Lenovo X-60 (Intel Dual CPUs of 1.6 GHz processors with 2GB memory) and has a built-in 802.11 a/b/g wi-fi interface. The mobile host, under different configurations, is wired to 3 additional cellular broadband data devices listed in Table I. These devices have the functionality of detecting 4G LTE signal, and will switch to 3G/3.5G automatically when 4G LTE/WiMax is not available. Note that the AT&T and Sprint devices we use are mobile hotspots, which can serve as 802.11 Wi-Fi APs to allow multiple users to associate with it for bandwidth sharing, we disable the functionality of Wi-Fi bandwidth sharing, and wire it to our mobile client through a USB cable[2]

[2]As of April 2012, none of Verizon 4G dongles supports Linux as advertised. We connect the USB modem to a 4G mobile broadband router made by Cradlepoint [7], and wire it to the Ethernet port of the mobile client. Note that Verzion does carry mobile hotspots, called "Mi-Fi", which are similar to other mobile hotspots but do not support USB tethering (hence the mobile client is forced to use wifi to connect to Verizon mobile hotspot).

Carrier	Location	Technology	BW (Mbps)		RTT (ms)		Loss Rate (%)		3WHS RTT (ms)	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Verizon	Boston MA	4G LTE	7.15	(2.93)	141.62	(91.83)	0.20	(0.08)	67.72	(8.66)
AT&T		4G LTE	10.72	(1.10)	65.84	(14.94)	0.03	(0.03)	38.80	(0.42)
AT&T (adj)		4G LTE	15.63	(1.04)	124.64	(50.51)	0.05	(0.03)	40.57	(2.72)
Sprint		4G WiMax	2.72	(0.45)	166.22	(55.52)	0.26	(0.07)	158.16	(102.75)
RCN Wi-Fi		802.11	10.72	(0.15)	46.97	(22.80)	0.20	(0.01)	13.33	(1.66)
Verizon	Amherst MA	4G LTE	12.01	(2.78)	77.31	(21.43)	0.13	(0.09)	76.39	(13.43)
AT&T		3.5G HSPA+	6.49	(1.13)	115.32	(31.06)	0.14	(0.05)	170.18	(115.34)
Sprint		3G EVDO	1.19	(0.24)	348.02	(151.28)	0.27	(0.13)	107.03	(39.35)
UMass Wi-Fi		802.11	18.32	(0.87)	39.92	(16.82)	0.25	(0.08)	2.75	(1.10)
Verizon	Sunderland, MA	3G eHRPD	1.85	(0.30)	362.37	(135.28)	0.10	(0.32)	153.27	(217.08)

TABLE II
SINGLE-PATH TCP MEASUREMENTS AT BOSTON AND AMHERST AREAS (MA)

A. Methodology

In all experiments, we collect packet traces from both interfaces of the UMass server and all the interfaces used of the client using *tcpdump* [22], and use *tcptrace* [23] to analyze the collected traces. For each experiment, unless specifically mentioned, the mobile client downloads a 100MB file from the UMass server, and each result shown below is the average of (at least) 6 runs of any particular configuration in a day (morning/afternoon/evening).

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

- **Throughput**:
  The average throughput is computed as the number of bytes sent divided by the elapsed time (in Mbps).
- **Round trip time (RTT)**:
  We measure RTT on a per-flow basis. Denote T_r as the server's receive time of an ACK packet for a previous packet sent from the server at time T_s over a flow. Hence, the RTT is the time difference between the time when a packet is sent by the server and the time it receives the ACK for that packet (i.e., $RTT = T_r - T_s$), and only non-duplicate ACKs and non-retransmitted packets are considered. Note that the RTT here is different from that reported in [10], which uses only the RTT of the 3-way handshake (3WHS), which is usually lower than the RTT of data packets, as presented in Tables II, III, and IV.
- **Loss rate**:
  The average loss rate is calculated on a per-flow basis, and it is computed as the total number of retransmitted data packets divided by the total number of data packets sent from the server.

IV. RESULTS

A. Static Scenarios

1) Single-Path TCP: We have performed experiments for single-path TCP with New Reno algorithm at multiple places in Massachusetts USA, and summarize the average performance metrics in Table II for easy reference. In general, we observe very different properties for 4G networks than for 3G networks in terms of all the performance metrics.

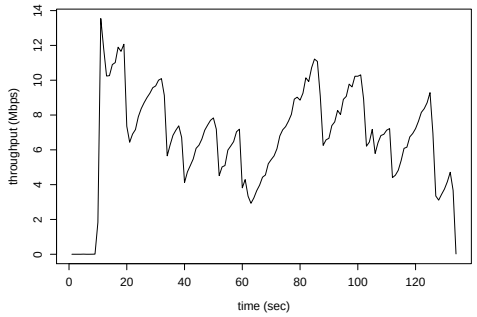

Fig. 1. Verizon 4G LTE throughput

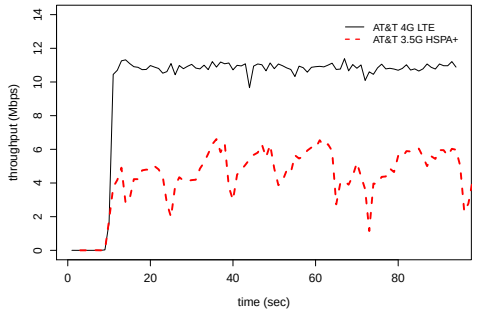

Fig. 2. AT&T 4G and 3.5G throughput

Figure 1 shows the throughput of Verizon LTE when downloading a 100 MB file from the UMass server over a single-path TCP connection as a function of time. A regular TCP Reno connection over Verizon LTE network behaves as follows: it increases its congestion window on each ACK received, and hence very quickly reaches the link capacity (the peak rate observed here is 13.54 Mbps). When this happens, packets start getting dropped, resulting in the congestion window size reducing by half, and the controller enters the congestion avoidance phase. This mechanism repeats until all packets are correctly received and ACKed, and hence the

typical TCP sawtooth curve is present.

With an even higher average download rate, surprisingly, we do not observe many packet losses from connections over the AT&T LTE network. For many rounds, the packet loss rate is even zero. Figure 2 shows the throughput of AT&T 4G LTE and 3.5G HSPA+ networks. It is interesting that the AT&T 4G throughput curve does not exhibit the typical TCP sawtooth pattern as Verizon 4G does in Figure 1, and seems to be capped at 11 Mbps.

When we inspect packet headers, we observe that the receiver window advertised in the AT&T LTE network differ substantially from that of Verizon LTE network. Right after the connection is established, the AT&T's receiver advertised window is set to a maximum value of 100,800 bytes and remains constant afterwards.

Since the receiver advertises its buffer size as 100,800 bytes, and the sender's congestion window size is restricted to the value of $\min(cwnd, rwnd)$, where $cwnd$ and $rwnd$ denote the updated congestion window size and the receiver advertised window size. We conjecture that the receiver advertised window size limits the number of packets that the sender can transmit over the AT&T TCP connections, since for Verizon LTE network, the receiver advertised window size is set to 676,304 bytes, which is 6.7 times larger than AT&T's advertised value.

Assume that the sender's congestion window is set to the receiver advertised window size of 100,800 bytes (equivalent to 0.77 Mbits). As our measurements show that the average RTT of the AT&T LTE network is 65.84 ms, the maximum data rate that the sender can send is

$$Rate = \frac{cwnd}{RTT} = \frac{0.77}{0.06584} \approx 11.68\ Mbps.$$

The calculation matches our measured average throughput of 10.72 Mbps and the instantaneous rate is stable as shown in Figure 2.

Discussion and Performance Comparison:

a) Small receiver advertised window: the maximum receiver advertised window is constrained by the bandwidth delay product ($BW \times RTT$), and the maximum socket receive buffer allocated by the Linux kernel. TCP maintains part of the buffer as the receive window advertised to the other end, and the rest of the space is used as the application buffer.

As the default maximum TCP receive socket buffer size for a 2GB RAM machine is 880KB, the AT&T USB tethering device tends to modify the assigned value, and hence gets less memory for its socket buffer (i.e., 100,800 Bytes), compared to other carriers' devices using the complete assigned memory size. This small advertised value restricts AT&T's window increase, and yields a low-loss and stable TCP connection. We also observe the same receiver advertised window in AT&T 3.5G network. Because the link capacity of AT&T 3.5G bandwidth is much lower than 11.68 Mbps, we observe the sawtooth behavior is prominent, as shown in Figure 2. Note that in the multi-path TCP kernel implementation [5], the Linux kernel maintains a connection-level receive buffer (over all sockets) as a shared resource, and uses this value for per-flow window advertisement. Therefore, for multi-path TCP connections, the small receiver advertised window from a particular carrier in the above scenario would not be problematic.

b) Rate control and loss reduction: To understand how the small receiver advertised window can affect the performance in AT&T networks, we first increase the maximum receive socket buffer size allocated to the AT&T device by eight times, and make sure that each AT&T TCP socket has enough receive buffer to advertise. As listed in Table II, the average throughput of the adjusted version, AT&T (adj), is approximately 15.63 Mbps, a roughly 46% performance gain. Furthermore, even after removing the constraint of small receiver advertised window, surprisingly, we still did not see prominent sawtooth patterns when inspecting the throughput as a function of time. Our observations show that when the sender starts transmitting at a high rate, the scheduler of the AT&T LTE base station, the evolved NodeB (eNB), queues packets sent from the server, and appears to drain the queue quickly or when a packet loss occurs, resulting in much larger packet RTTs. Furthermore, for non-retransmitted packets, they will be queued for different amount of time, but the maximum RTT is capped at 200 ms (will discuss the details in the following section). If we assume the 3-way handshakes packets do not suffer any additional queueing delay from by the AT&T network (since they are the first few packets of a connection), and any following data packets can be queued with a RTT up to 200 ms, then the average round trip time is

$$RTT \approx \frac{RTT_{3WHS} + RTT_{\max Delay}}{2} = \frac{40.57 + 200}{2} = 120.285$$

which matches our measurements of the average RTT of the adjusted AT&T LTE measurements, AT&T (adj), in Table II, which is 124 ms. If no additional queueing delay is introduced, the AT&T LTE should be able to achieve an ideal throughput roughly 3 times higher. We will see details in the following section.

2) Multi-Path TCP: In this section, we measure the throughput of multi-path TCP and compare it to that of single path TCP. As we have shown that single-path TCP's performance (i.e., TCP New Reno) over all three carriers' networks in Section IV-A, here we present measurement results of multi-path TCP in static scenarios.

a) Two client interfaces (4 flows):
In this configuration, we activate both the AT&T and Verizon 4G devices, and run multi-path TCP at both ends, downloading the same 100MB file from the UMass server. Note that in Section IV-A, we discussed the issue that the maximum throughput can be capped by assigning receiver advertised window size with a smaller number. In the multi-path TCP configuration, on the other hand, the receiver buffer is shared

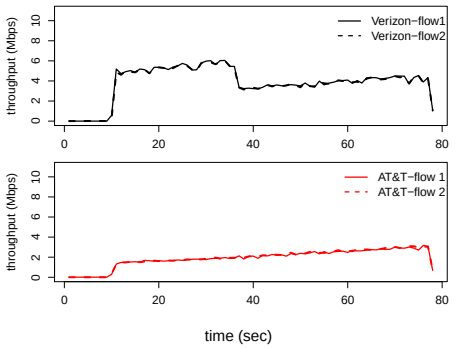

Fig. 3. Throughput of 4 flows from Verizon and AT&T LTE

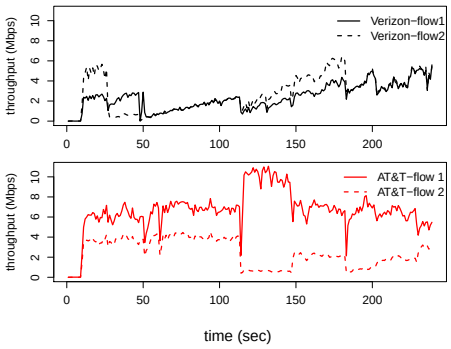

Fig. 4. The calibrated slow-start configuration of the 4-flow case

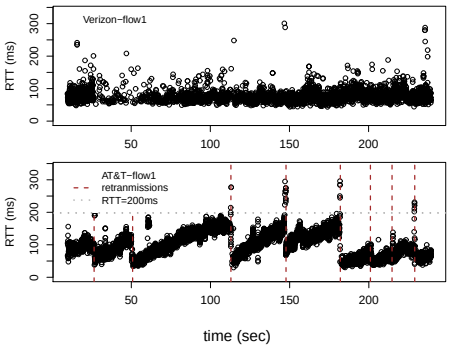

Fig. 5. Round trip time samples of Verizon and AT&T LTE flow-1

Carrier	Technology	BW (Mbps)		RTT (ms)		Loss Rate (%)		3WHS RTT (ms)		Aggregate
		Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Throughput
Amherst, MA										**19.59 Mbps**
Verizon-1		2.45	(0.63)	77.10	(17.54)	0.11	(0.05)	93.47	(27.95)	
Verizon-2	4G LTE	1.96	(0.53)	76.26	(17.06)	0.13	(0.08)	90.78	(27.53)	4.41
AT&T-1		0.95	(0.47)	117.51	(39.80)	0.05	(0.05)	295.87	(62.31)	
AT&T-2	3.5G HSPA+	1.16	(0.57)	115.27	(34.66)	0.06	(0.05)	321.78	(19.96)	2.11
Sprint-1		0.42	(0.12)	245.69	(105.29)	0.21	(0.22)	147.70	(110.08)	
Sprint-2	3G EVDO	0.46	(0.23)	253.26	(108.41)	0.18	(0.08)	164.40	(114.65)	0.88
UMass-1		6.43	(0.95)	17.20	(9.97)	0.09	(0.02)	3.43	(0.98)	
UMass-2	Wi-Fi	5.76	(1.12)	18.10	(10.40)	0.12	(0.03)	3.37	(0.85)	12.19
Boston, MA										**16.45 Mbps**
Verizon-1		2.53	(1.36)	79.95	(33.67)	0.09	(0.05)	63.77	(7.75)	
Verizon-2	4G LTE	2.06	(1.09)	84.23	(48.07)	0.17	(0.08)	71.77	(5.79)	4.59
AT&T-1		3.98	(2.49)	92.48	(32.08)	0.13	(0.06)	53.33	(13.08)	
AT&T-2	4G LTE	3.52	(2.29)	81.75	(43.12)	0.14	(0.12)	53.23	(16.32)	7.50
Sprint-1		0.80	(0.47)	129.68	(44.20)	0.24	(0.16)	98.47	(7.44)	
Sprint-2	4G WiMax	0.68	(0.50)	130.02	(40.88)	0.24	(0.13)	98.73	(7.19)	1.48
Boston RCN-1		1.45	(0.22)	23.12	(20.63)	0.48	(0.08)	14.00	(1.43)	
Boston RCN-2	Wi-Fi	1.43	(0.41)	24.33	(19.48)	0.44	(0.16)	14.18	(1.60)	2.88

TABLE III
STATIC 8-FLOW SCENARIOS IN AMHERST AND BOSTON, MA

by all flows and hence the receiver advertised window size of each flow is the same.

Figure 3 shows the throughput of 4 flows from the mobile client (2 interfaces: Verizon and AT&T) to the UMass server (2 interfaces, denote as flows 1 and 2). In this case, both AT&T and Verizon 4G networks have similar bandwidth and RTTs (around 100 ms). As in Figure 3, Verizon starts at around 5Mbps, and its two flows very quickly reach the link capacity 12 Mbps (6Mbps each). These flows then encounter packet losses at around 40 seconds. Both Verizon flows reduce their congestion windows by half, and increase their windows at a similar speed as the AT&T flows. Note that the AT&T flows start at much lower rates and do not suffer any loss. Both flows increase the throughput gradually and the aggregate throughput of the two flows (approximately 7Mbps) did not reach maximum link capacity even after the completion of downloading the file of 100MB at 80 seconds.

Discussion and adjustment:

As shown in Figure 3, there is a critical issue involved when running multi-path TCP in high link bandwidth environments. Our observations show that, in many cases, flows to the some carrier device seem to leave slow start at a very early manner, and the flows associated with the same carrier tend to leave the slow start simultaneously and very quickly (if no loss occurs). Since the flows' data rates increase much slower in the congestion avoidance phase than in the slow-start phase (linearly vs. exponentially), flows with early entrance to the congestion avoidance phase takes a very long time to reach a high bandwidth as the AT&T flows in Figure 3.

This issue is not a big concern in single-path TCP, as the single-path TCP congestion controller behaves N times more aggressive than any of the N flows in a multi-path TCP connection at the window increase phase (assume that all flows are of similar conditions). However, when one flow leaves the slow start phase at a very low data rate, followed by the $1/N$ slower window increase rate, it takes at least N times longer than single-path TCP to achieve the same data rate. As

shown in Figure 3, the two AT&T flows' data rates start at approximately 1.5 Mbps, and do not grow fast enough with reasonable high throughput even after one minute. This results in performance degradation of multi-path TCP connections.

The main cause of these issues is the default setting of Linux TCP sender's caching slow start threshold ($ssthresh$) for each IP-destination [18]. If an additional TCP connection is established to the same destination IP address, the cached value can be used for efficiently initializing the new TCP connection.

When a TCP connection encounters a loss (after three successive duplicate ACKs), as the standard procedure of fast retransmit, the sender resends the lost packet and sets its slow start threshold as follows:

$$ssthresh = \max\left\{\frac{FlightSize}{2}, 2 \times SMSS\right\}$$

$FlightSize$ is the amount of outstanding (un-ACKed) data in the network, and $SMSS$ is the size of the largest segment that the sender can transmit [2]. That is, when an established TCP connection suffers a sequence of losses, the $ssthresh$ value might be set to as low as 2 packets, and will be preserved to cutoff the slow-start for the next flows to the same IP address (i.e., the same cellular network carrier device).

For high bandwidth transmission links, we propose to modify this default Linux TCP sender setting to *not cache metrics* on closing connections to prevent possible performance degradation due to early congestion avoidance. Each newly opened flow/connection now uses the default slow-start threshold [21] (64KB, roughly 5Mbps for 100 ms RTT links) before encountering a packet loss.

Note that now each flow has its own slow-start threshold, and the throughput over time of the calibrated configurations of a 4-flow multi-path TCP is shown in Figure 4. Flows now leave the slow start phase at approximately $5 \sim 6$ Mbps when no loss occurs, as do Verizon flow-2 and AT&T flow-1. On the other hand, Verizon flow-1 and AT&T flow-2 in Figure 4 suffer losses during the slow start phase at the beginning, and hence the congestion controllers leave slow start and enter congestion avoidance.

With the calibrated TCP configurations, flows now leave the slow start phase with reasonable high data rates. As discussed in the previous section, when flows of AT&T LTE start sending packets at high data rates, data packets will be buffered in the network and results in a maximum RTT of 200 ms. To better characterize this behavior, we extended the download time to see if there are any prominent patterns. Figure 5 shows the RTT of each packet sent at a particular time from flow-1 of Verizon and AT&T. In Figure 4, the aggregate throughput of AT&T flows is 10 Mbps after leaving the slow start phase, but the throughput of both AT&T flows do not increase linearly as the Verizon flows do. This is mainly because the packets are queued in the network. Since packet RTTs of the AT&T flow-1 increases almost linearly during each RTT increase period, and the flow throughput remains stable, we can still infer that the AT&T flow-1's congestion window also increases during the same period (as flow throughput$= cwnd/RTT$).

An interesting observation is that at the end of each RTT increase period, the packet RTTs drop very quickly from 200 ms to roughly 40 ms. Our observations show that these drops occurs when a packet is dropped. In Figure 5, the vertical lines are the times where retransmissions over AT&T network take place at the server after packet losses.

Following by each packet loss/retransmission event, the scheduler of the eNB drains the queue very quickly with a sharp drop of RTT values. This is because after the lost event occurs, during fast recovery, the sender's congestion window size increases by one when an additional duplicate ACK is received after the lost packet is identified (by three duplicate ACKs), resulting in more unACKed packets. When the lost packet is correctly ACKed, the sender's congestion window will be set to $cwnd = ssthresh + 3$, where $ssthresh$ is half size of the congestion window before fast recovery [21]. When the congestion window is set to this new value (reduced by half), the sender stops transmitting packets for a period of $RTT/2$ until the number of unACKed packets is smaller than the number of new congestion window size. During this period of time, the scheduler drains the packets in the queue. Note that we also inspected the Sprint and Verizon's packet RTT samples, and did not see the same patterns in their networks.

b) Four client interfaces (8 flows):

As we have resolved all the issues which can potentially degrade the performance of running multi-path data transport, Table III lists the performance of static 8-flow multi-path TCP scenarios in Boston and Amherst. Note that in Boston, all three carriers (Verizon, AT&T, and Sprint) are of 4G services, and the provider of Wi-Fi network is RCN. When in UMass, we have a very diverse combination of wireless technologies. Our devices connect to Verizon via 4G LTE, to AT&T via 3.5G HSPA+, to Sprint via 3G EV-DO, and to Wi-Fi through UMass campus-wide free Wi-Fi. As the first goal of using multi-path TCP, one should have higher throughput when running multi-path TCP for data transport, and should compare the throughput of a multi-path TCP connection to that of the best of single-path TCP running on any of the utilized paths for the reason of fairness.

As listed in Table III, when running multi-path TCP for data transport in the Amherst configuration, the average aggregate throughput is 19.59 Mbps, which is higher than that of the best single-path TCP connection from UMass Wi-Fi at 18.32 Mbps (shown in Table II), a roughly 7% performance gain. Similarly, in the Boston configuration, the average aggregate throughput of multi-path TCP connections is 16.45 Mbps, which outperforms the best of the single-path TCP connection from AT&T's (adjusted) 4G LTE by 5.25%.

In the next section, we are going to present how one can benefit from leveraging path diversity of cellular networks in mobile scenarios. As in most of the mobile and vehicular

scenarios, wireless links might break from time to time; running multi-path TCP in such kind of scenarios provides a different aspect of robustness for data transport.

B. Mobile Scenarios

1) Single-Path TCP: In the following subsections, we will sketch how one can benefit from using multi-path TCP in mobile scenarios. We first performed experiments of single-path TCP in the mobile scenarios as a baseline. The experiments were mainly conducted at two rural towns in the western Massachusetts, Amherst and Sunderland, and at Boston city in eastern Massachusetts USA. For all the mobile experiments, unless specified, were conducted at an average speed of 30 MPH, with 100 MB data transferred for mobile single-path TCP scenarios, and 400 MB data transferred for multi-path TCP scenarios.

Figure 6 illustrates the throughput changes over time of mobile single-path TCP for AT&T 4G/3.5G, Verizon 4G/3G, and Sprint 4G/3G connections. Table IV summarizes the performance of all these technologies in mobile scenarios. Note that the sawtooth pattern of AT&T 4G throughput curve becomes prominent in mobile scenarios, and we conjecture that it is because now the huge chunk of packets is not only received and processed at a single eNB (base station) during our movement in Boston, and hence packets are not queued for long at each eNB. This is also reflected in the relatively short average RTT of 67.06 ms in Table IV (as opposed to an average of 124.64 ms in Table II).

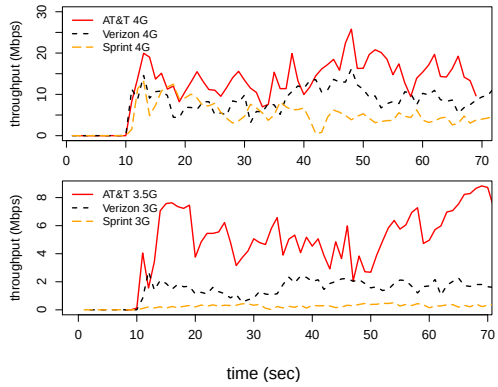

Fig. 6. Throughput of mobile single-path TCP

2) Multi-Path TCP: To better illustrate the robustness of using multi-path TCP in mobile scenarios where users might encounter technology switch and broken links due to the diverse cellular network availability, in each of the following scenarios, we make sure that there exists at least one working path in the multi-path data transport. As listed in Table II, we have AT&T 3.5G service in the entire western Massachusetts while Verizon offers 3G service in Sunderland and 4G LTE service in Amherst. To demonstrate how robust multi-path data transport is while moving across zones where carriers have different levels of coverage from various technologies, we will examine the following cases:

- Move from 4G to 3G.
- Move from 3G to 4G.
- Drive around UMass campus with intermittent Wi-Fi.
- Drive around Boston (all 4G) with intermittent Wi-Fi.

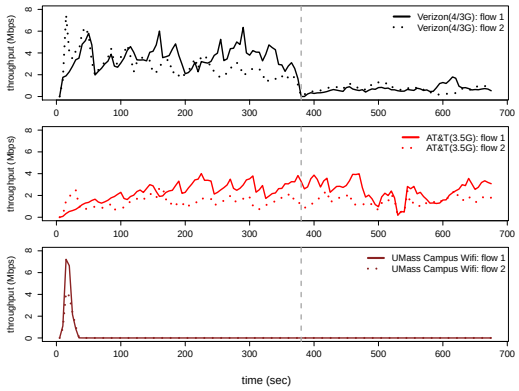

Fig. 7. Verizon 4G to 3G: individual throughput

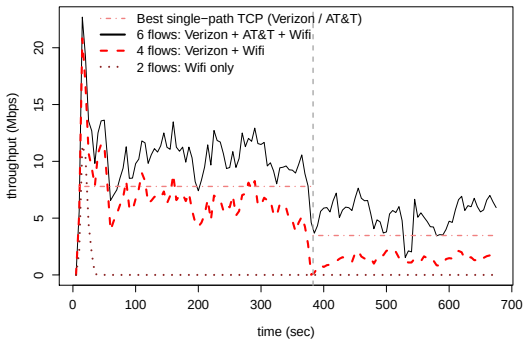

Fig. 8. Verizon 4G to 3G: aggregate throughput

a) Move from 4G to 3G:

Case 1: Verizon 4G LTE to 3G eHRPD

We start with the case of moving from Verizon 4G zone to 3G zone as Verizon is the only carrier that offers 4G services in Amherst, MA. In this scenario, we drove from Amherst to Sunderland, a town 7 miles away from the UMass campus, at an average speed of 40 MPH. Table V lists the schedule and the average throughput during the different periods while driving from 4G to 3G zone. Note that during the first and last minute, we were stationary for experiment setup/termination.

Figure 7 shows the throughput of each flow over time. The Wi-Fi paths broke right after our movement, and provided no throughput afterwards. We entered the 4G/3G handoff period during 375−385 seconds, where the throughput of both Verizon's flows dropped to 0 until the device successfully switched to 3G service, and the associated IPs (the external public IP address assigned by Verizon to the device, and the

Carrier	Location	Technology	BW (Mbps)		RTT (ms)		Loss Rate (%)		3WHS RTT (ms)	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Verizon	Boston, MA	4G LTE	9.43	(0.90)	107.80	(63.07)	0.30	(0.23)	74.95	(13.07)
AT&T		4G LTE	14.74	(1.32)	67.06	(26.10)	0.08	(0.06)	41.12	(3.44)
Sprint		4G WiMax	4.81	(0.45)	154.43	(92.37)	0.35	(0.22)	154.93	(83.10)
Verizon	Amherst, MA	4G LTE	7.79	(1.79)	104.90	(49.57)	0.13	(0.04)	69.58	(9.30)
AT&T	Western MA	3.5G HSPA+	3.47	(1.13)	158.46	(98.29)	0.11	(0.03)	244.31	(66.87)
Verizon	Sunderland, MA	3G eHRPD	0.85	(0.12)	695.10	(349.90)	0.21	(0.03)	189.10	(174.35)
Sprint	Amherst, MA	3G EV-DO	0.31	(0.07)	673.73	(333.07)	0.66	(0.27)	510.23	(79.58)

TABLE IV
MOBILE SINGLE-PATH TCP MEASUREMENTS: 4G/3.5G/3G IN MASSACHUSETTS

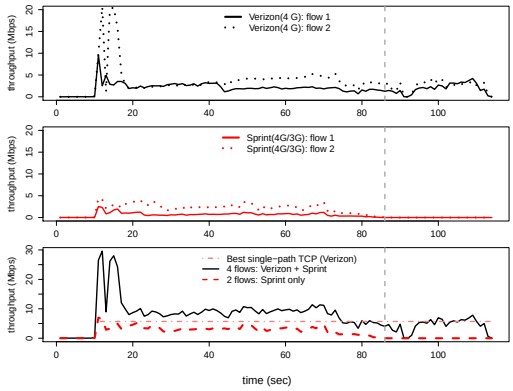

Fig. 9. Sprint 4G to 3G: individual/agg. throughput

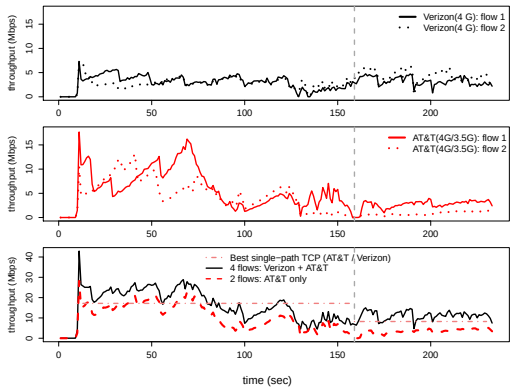

Fig. 10. AT&T 4G to 3G: individual/agg. throughput

Time (sec)	Description	Avg. Throughput
0−30	Stationary	12.29
30−60	Leaving UMass Wi-Fi	11.08
60−375	Within Verizon 4G zone	10.27
375−385	Transition: 4G to 3G	4.21
385− 590	Within Verizon 3G zone	5.12
590−675	Stationary	5.90
0−675	UMass to Sunderland	8.18 Mbps

TABLE V
MOBILE SCHEDULE: FROM 4G TO 3G ZONE

private IP address assigned by the device to the client) and port numbers remained the same after the switch[3]. Figure 8 illustrates the aggregate throughput of 2/4/6 flows over time, and 750MB data was transferred. When in the Verizon 4G zone, the multi-path TCP connection has an average throughput of 10.48 Mbps, which is better than that of the best single-path TCP (Verizon 4G LTE: 7.79 Mbps). Similarly, when in the Verizon 3G zone, the multi-path TCP connection has an average throughput of 5.30 Mbps, which outperforms the best mobile single-path TCP connection (AT&T 3.5G: 3.47 Mbps).

[3]As all the carriers do not directly assign public IP addresses to their customers, when a client is associating with a carrier, the carrier first assigns a private IP address to the client. Then an external public IP address will be shared by a group of customers, distinguished by the port number of that public IP address.

The following experiments on the effect of technology transitions on throughput of Sprint and AT&T were conducted when moving from Boston to the suburban areas. Along the way, we first encountered signal loss from Sprint 4G WiMax, followed by AT&T 4G LTE, and then Verizon 4G LTE (7 miles away from Boston). As Verizon's LTE coverage range is broader than the other two carriers, for simplicity, we examine AT&T and Sprint's backward compatibility with the following configurations: Verizon with AT&T, and Verizon with Sprint (i.e., use Verizon flows as working paths).

Case 2: Sprint 4G WiMax to 3G EV-DO

Figure 9 shows the throughput changes over time when traveling from Sprint 4G zone to 3G zone. The two flows from Sprint WiMax have an aggregate throughput around 3Mbps before entering the 3G zone. When detecting losing 4G signal at time 86 seconds, the Sprint's device, unlike those of other carriers, did not perform fast switch to 3G. Instead, it bootstrapped the device again and eventually reconnected to 3G. This results in getting a new public external IP address from Sprint's 3G network, but remaining the same private IP address assigned by the device, and leads to stalled Sprint flows in the multi-path TCP connections. This is because the client is not aware of the external public IP address change, and is still waiting for the server's retransmissions to the old public IP address (which may have been assigned to other customers).

However, although the Sprint's flows are not available after the transition, the data transport via multi-path TCP is still continuing. Traffic previously riding on the Sprint's flows is now offloading to the Verizon flows after time 86 seconds as shown in Figure 9. Note that during this experiment, 400 MB data was transferred, and the aggregate throughput of 2/4 flows is compared to that of the best mobile single-path TCP at the same place during that period, as shown at the bottom in Figure 9.

Case 3: AT&T 4G LTE to 3G HSPA+

Figure 10 shows the case when we drove 2 miles farther from Boston, we started losing AT&T's 4G signal. Similar to Verizon's transition from 4G to 3G, the two AT&T flows started switching to 3G service at time 157 seconds, and did not respond to the UMass server's retransmissions until five seconds later. After the switch, the mobile client kept the same private/external IP addresses. Note that in this experiment, 400MB file was transferred, and the aggregate throughput of multi-path TCP is also compared to the best of mobile single-path TCP at the same place during the same period of time at the bottom of Figure 10.

In all, the cellular network backward compatibility to switch from 4G to 3G results in connection break (or stalled) in all carriers for single-path TCP scenarios. When transmitting data with multi-path TCP, the fast switch to 3G results in an average of 10-second path break in Verizon network, an average of 5-second path break in AT&T network, and stalled paths in Sprint network due to the mobile client's unaware of external IP address change. During this period, the sender continues retransmitting packets on the broken paths, while traffic on those broken paths will be offloaded to other working paths. If the device successfully switches to 3G, then packet exchanges over associated paths will resume; otherwise, the existing working flows will continue the data transport and complete the file transfer.

b) Move from 3G to 4G:

When we inspect the effect of technology transitions from 3G to 4G, surprisingly, devices of all carriers behaved similarly. When we start a multi-path TCP connection with 3G service, the corresponding devices do not switch to 4G for the entire data transport, even though we have entered a 4G zone. Due to the space limit, we do not report the results here, as those devices do not switch from 3G to 4G.

c) UMass Campus with intermittent Wi-Fi:

In contrast to the first two cases, here we show the scenario when moving around UMass campus with intermittent free Wi-Fi connectivity. Figure 11 illustrates the throughput of each flow over time while moving on UMass campus. Wi-Fi flows resumed occasionally when passing through hotspots, but in general do not yield high throughput due to weak signal strength and the mobility (average on campus speed is 15 MPH). In this case, 300 MB data was trasferred, and the average aggregate throughput of the multi-path TCP connection is 8.96 Mbps, which outperforms the best of mobile single-path TCP (Verizon 4G: 7.79 Mbps) by roughly 15%.

Note that both Wi-Fi flows were created during the first minute, but only flow-1 continued transferring packets at time 130 and 200 seconds. Wi-Fi flow-2 still remained active, but the sender missed the first few chances to deliver packets due to the long retransmission timeouts (retransmissions occurred at times 83, 115, 178, and 306 seconds, respectively), spanning across the two hotspot-revisit periods at 130 and 200 seconds. As the establishment of Wi-Fi links takes 10−15 seconds [6], current TCP retransmission scheme makes it very challenging to re-establish broken Wi-Fi paths when the timeout timer goes beyond 10 seconds and increases exponentially.

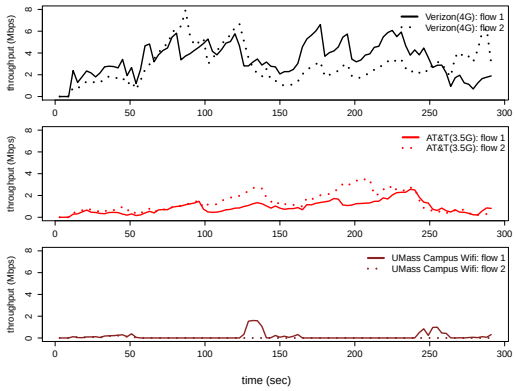

Fig. 11. Within UMass: individual throughput

d) Boston with all 4G and intermittent Wi-Fi:

Figure 12 presents the per-flow throughput of the scenario while driving in Boston with intermittent Wi-Fi connections. Note that in this case, we have all the devices from cellular networks of 4G services. The sawtooth pattern of the AT&T LTE throughput curve is prominent as in this case of mobile single-path TCP. During this experiment, 800MB data was transferred, and the average throughput of the 8-flow multi-path TCP connection is 16.82 Mpbs, with a 14% performance gain compared to the best single-path TCP from AT&T LTE (14.74 Mbps).

3) Performance comparison and discussions: In general, the performance of a single-path TCP connection in mobile scenarios does not degrade from the static cases with reasonable driving speed, as in Tables II and IV. By leveraging the path diversity of 4G/3G for multi-path TCP, mobile users do not need to worry about stalled connections even when entering an area where certain paths break momentarily or do not have signal coverage, as long as there is a working path still delivering packets properly. However, a more serious issue was described in mobile scenario (c), where intermittent

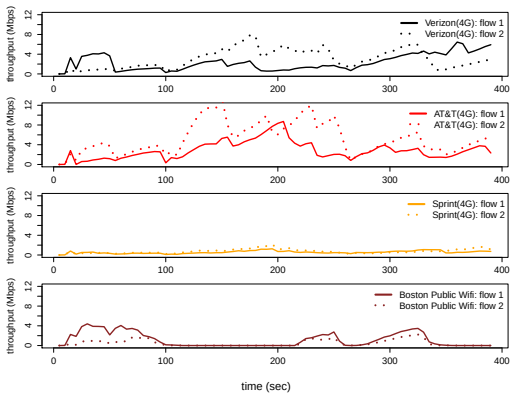

Fig. 12. Boston with 4G: individual throughput

Wi-Fi connectivity is available. In that case, retransmission timeout of a broken path increases exponentially and makes it challenging to re-establish broken Wi-Fi links as Wi-Fi link availability in mobile scenarios might be less than a minute. Although one can always remove the exponential increase of timeout in TCP, however, a better solution is to fast detect unstable Wi-Fi links. When the Wi-Fi link quality is below a certain level, then we stop transmitting packets through that path, and will only resume data transfer when the link quality is beyond certain level.

C. Other Issues

Further issues related to running multi-path TCP over cellular data networks are discussed in this subsection. It was reported in [25] that TCP split-connections were widely implemented in major US 3G networks. As split TCP connections may cause problems for the use of multi-path TCP, it is of our interest to understand the usage of split-connection in modern 3G/4G networks. Furthermore, some cellular network carriers tend to terminate least used or idle flows very quickly to save the resources in the cellular networks. This quick timeout forces a flow to be removed unexpected, introducing additional overhead for flow re-establishment.

1) Impact of split connection to multi-path TCP: Split-connection is used to improve TCP performance over wireless links by inserting a split point between the wireless and wired hosts, thus splitting the end-to-end TCP connection into two separate connections. As wireless links usually have worse performance than wired ones, the goal of splitting a connection is to separate wireless related issues from the wired hosts (the split point and the UMass servers).

To detect split-connections, we actively delay each ACK sent from the mobile client by a time interval of T before it is returned to the server [25]. Let $\{R_i\}_{i=1}^{n}$ denote the sequence of RTTs measured at the server, where n is the total number of unique data packets. If $\min_{1\leq i\leq n}\{R_i\} < T$, then the observed TCP connection is split. This delayed ACK approach does not require clock synchronization.

We examined most popular TCP applications and checked to see if the corresponding TCP connections are split. This included FTP (port 20, 21), SSH (port 22, including SCP), HTTP (port 80), and POP (port 110).

Fortunately, split-connection only occurs for the FTP control connections (port 21) of all carriers, and our further inspection shows that port 21 cannot be used for data transfer for all carriers. AT&T uses Web traffic proxies (port 80) for its 4G/3.5G networks. As we delayed 5,000 ms for each ACK sent from the client to the UMass server (at port 80) through AT&T 4G/3.5G networks, all the packet RTTs fall below 1,000 ms. When we download the same file multiple times consecutively, the UMass server only received the first request from the client. A detailed inspection is that AT&T does not support TCP option fields for Web traffic (both 3.5G and 4G). It only allows the mobile host to initiate a single-path TCP connection for web traffic. If any of the end points wishes to establish an additional flow for web traffic through AT&T 4G/3.5G networks, the option field will be wiped out, and hence the additional flow will be terminated immediately after the request is sent. Thereby one should consider other ports for web data transfer through AT&T cellular networks when using multi-path TCP.

As each flow in a multi-path TCP connection is designed to behave like a regular TCP connection. When a flow is established, it should not deal with issues more than that of a regular TCP connection [17]. From our measurements, these transparent behaviors done by the carriers (as users will not notice them unless to detect them actively) along the way of packets' traversing from one end to the other, have caused fundamental problems when running multi-path TCP.

2) Connection timeout policies: Another important factor of running multi-path TCP over cellular networks is to maintain each flow active when necessary. Table VI lists the time out thresholds of all carriers over different technologies. According to our observations, some of the carriers sets the connection timeout as short as 3 minutes. In order to maintain the existing flows in a multi-path TCP connection during mobile scenarios, one should send a keep-alive probing packet to the other end with a period of time smaller the listed timeout values.

Carrier	Type	Time Out	Type	Time Out
Verizon	LTE	30 min	eHRPD	60 min
AT&T	LTE	3 min	HSPA+	3 min
Sprint	WiMax	30 min	EVDO	30 min

TABLE VI
CONNECTION TIME OUT POLICY OF EACH CARRIER

V. RELATED WORK

Multi-path TCP has been proposed to exploit path diversity to improve TCP performance. Although it has been deployed and discussed for traditional Internet scenarios, little is understood about how well it will perform in mobile scenarios. Early work [9] performed side-by-side comparison of using open Wi-Fi and 3G only, and show that intermittent Wi-Fi connectivity in an urban area can yield equivalent or

greater throughput than what can be achieved using an always-connected, but slow 3G network. Based upon this idea, [4] proposed an interim scheme to tackle the traffic on congested and slow 3G networks by introducing a proxy to offload delay-tolerant data to the intermittent but fast Wi-Fi connections. This may break the connection into several short, possibly delayed connections, and will introduce additional overhead during the setup period of these newly opened TCP connections. On the other hand, Raiciu et al. [16] proposed a scheme to enhance the use of multi-path TCP when any of the end hosts is not multi-path TCP capable. The scheme requires a publicly available proxy with multi-path TCP capability sitting between both ends, and hence splits the TCP connection. If both ends are capable of running multi-path TCP, the public proxy would just serve as a packet relay.

In this paper, we characterize the state of art cellular data networks, 4G LTE/WiMax and 3.5/3G networks, and show that running TCP with path diversity can cope with mobility related issues. Although recent study has shown some properties of 4G/3G networks [10] [11], but their results were passively reported by users running mobile APPs. Here we have a very different emphasis on the performance of multi-path TCP in terms of throughput, loss rate, and round trip time. We focus more on measuring the latency of cellular networks when established connections have enough traffic riding on them, rather than looking at the round trip times of light weight probing packets, such as *ping* or 3-way handshake packets, which might underestimate the network congestion.

VI. Conclusion

As cellular technology evolves, taking advantage of multiple wireless radios provides a chance to improve the performance and availability of mobile Internet access. In this paper, we first characterize three major US 4G/3G networks in terms of throughput, round trip time, and loss rate. We then conduct experiments of single-path and multi-path TCP connections over these cellular data networks, and show that by leveraging path diversity of cellular networks, data transport using multi-path TCP is a promising solution for a more reliable and efficient data transfer scheme in changing environments.

We observe that 4G outperforms Wi-Fi and 3G in terms of the throughput and loss rate. In this high bandwidth environment, we observe that AT&T 4G network tends to control the rate at which data flows to the receivers, and results in queue buildups. Also, we suggest, in this high bandwidth environment, to remove the default Linux TCP's setting of caching sender's slow-start threshold for each destination IP address as that will cutoff the newly opened flow's slow-start value. We also show that, by properly assigning socket buffer to each flow in the AT&T LTE network, each flow can yield higher throughput. These issues would potentially degrade the performance of single- and multi-path TCP connections.

Furthermore, we have shown that multi-path TCP can support mobility without breaking existing connections, and can dynamically offload traffic to paths associated with different technologies of diverse availability during movement. Our experiments also show that the exponential timeout scheme is not proper for Wi-Fi links, and it should be adjusted since the revisits to Wi-Fi hotspots in the vehicular cases (or high speed movement) do not last longer than one or two minutes. We recommend a scheme where bad paths should be detected and removed quickly, and can resume packet exchanges when they become available.

In the meantime, we are currently investigating the cases where enabling multi-path TCP might not provide high throughput gain, especially when file size is small and a particular path has significantly higher quality. Further performance issues on separating the reliability from congestion control [19], and when to remove poorly performed paths [14] are our future work.

References

[1] 3GPP LTE. http://www.3gpp.org/lte.

[2] M. Allman, V. Paxson, and E. Blanton. RFC 5681 TCP Congestion Control, Sept. 2009.

[3] AT&T 4G LTE coverage. http://www.att.com/network/.

[4] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3g using wifi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 209–222. ACM, 2010.

[5] S. Barré, C. Paasch, and O. Bonaventure. Multipath TCP: from theory to practice. In *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, NETWORKING'11, pages 444–457. Springer-Verlag, 2011.

[6] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular internet access using in situ wi-fi networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, MobiCom '06, pages 50–61. ACM, 2006.

[7] cradlepoint. http://www.cradlepoint.com.

[8] A. Ford, C. Raiciu, H. M., and O. Bonaventure. TCP extensions for multipath operation with multiple addresses. draft ietf-mptcp-multiaddressed-07, 2012.

[9] R. Gass and C. Diot. An experimental performance comparison of 3g and wi-fi. In *Proceedings of the 11th international conference on Passive and active measurement*, PAM'10, pages 71–80. Springer-Verlag, 2010.

[10] 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 '12. ACM, 2012.

[11] 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 '10, pages 165–178. ACM, 2010.

[12] IEEE Standard 802.16. Broadband wireless metropolitan area networks (MANs) http://standards.ieee.org/about/get/802/802.16.html.

[13] 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.

[14] P. Key, L. Massoulié, and D. Towsley. Path selection and multipath congestion control. *Commun. ACM*, 54:109–116, Jan. 2011.

[15] MultiPath TCP Linux Kernel implementation. http://mptcp.info.ucl.ac.be/.

[16] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley. Opportunistic mobility with multipath TCP. In *Proceedings of the sixth international workshop on MobiArch*, MobiArch '11, pages 7–12. ACM, 2011.

[17] C. Raiciu, C. Paasch, S. Barré, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? designing and implementing a deployable multipath tcp. In *USENIX Symposium of Networked Systems Design and Implementation (NSDI'12), San Jose (CA)*, 2012.

[18] P. Sarolahti and A. Kuznetsov. Congestion control in linux TCP. In *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*, pages 49–62, Berkeley, CA, USA, 2002. USENIX Association.

[19] V. Sharma, K. Kar, K. K. Ramakrishnan, and K. S. A transport protocol to exploit multipath diversity in wireless networks. *Transactions on Networking (Infocom'08 extension)*, 2012.
[20] Sprint 4G WiMax coverage. http://www.clear.com/coverage.
[21] W. Stevens. RFC2581: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, Jan. 1997.
[22] tcpdump. http://www.tcpdump.org.
[23] tcptrace. http://www.tcptrace.org.
[24] Verizon Wireless 4G LTE coverage. http://news.verizonwireless.com/lte/markets.html.
[25] W. Wei, C. Zhang, H. Zang, J. Kurose, and D. Towsley. Inference and evaluation of split-connection approaches in cellular data networks. In *Proc. PAM (Passive and Active Measurement)*, 2006.
[26] 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'11. USENIX Association, 2011.