Mobile Accelerator: A New Approach to Improve TCP Performance in Mobile Data Networks

Ke Liu
The Chinese University of Hong Kong
lk008@ie.cuhk.edu.hk

Jack Y. B. Lee
The Chinese University of Hong Kong
yblee@ie.cuhk.edu.hk

***Abstract*—This paper investigates the performance of TCP in mobile data networks and proposes a novel approach to address the problem of optimizing transport protocol for these networks – a network-centric approach. We propose to realize protocol optimizations within the network by means of deploying a network-layer device — called a *mobile accelerator*, which performs protocol optimizations *on-the-fly* to improve transport protocol performance, without requiring any modification to the server or the client protocol implementations in the OS. Our extensive experiments conducted in production 3G/HSPA and LTE networks show that the proposed mobile accelerator can increase the throughput performance of TCP by over 200%. This mobile accelerator can be readily deployed in existing mobile data networks and can improve the network performance of *all* existing network applications running atop TCP.**

Index Terms*—Accelerator, Bandwidth Variation, Mobile Data Network, Transmission Control Protocol (TCP)

I. INTRODUCTION

Mobile Internet devices such as smartphones and netbooks will be the next major stage in the evolution of the Internet. Unlike conventional desktop computers these mobile devices are often connected to the Internet via wireless links. In particular, the growth of 3G [1] user population has been increasing rapidly in recent years and it is expected that the number of Internet users connected wirelessly will soon exceed their wired counterparts.

Despite the rapid developments in mobile Internet, the Internet's transport protocols, TCP [2-6] in particular, is still rooted in wired networks invented decades ago. While TCP works over mobile data networks, its performance in practice is often sub-optimal. Our experiments reveal that mobile data networks such as 3G/HSPA [1] and LTE [7], although is IP-based, have substantially different characteristics which prevent current TCP transports from performing optimally.

For example, in one experiment we downloaded a 15MB file from a wired Linux web server to a notebook computer connected to the Internet via a 3G/HSPA modem. The average download throughput turned out to be only ~1.5 Mbps, which is far lower than the maximum bandwidth of the 3G/HSPA network. By sending a UDP data flow over the same network to the notebook computer we were able to achieve a goodput of ~5.5 Mbps, which is much closer to the network's bandwidth limit. These results clearly show that the existing TCP implementation failed to fully utilize the bandwidth available in the mobile data network.

This paper tackles the performance problems of running TCP over mobile data networks. First we conducted extensive measurements in production mobile data networks including 3G/HSPA and LTE, and found that modern mobile data networks differ from wired networks in three important ways: (a) significantly longer and load-dependent round-trip-time (RTT); (b) non-congestion-related packet loss; and (c) rapid bandwidth fluctuations. These differences violate many of the assumptions in existing TCP implementations, and as a result, degrade their performance in mobile data networks significantly.

One approach to tackle these challenges is to modify or redesign TCP for use in mobile data networks (e.g., TCP Westwood [5] and TCP Veno [6]). While this approach is sound in principle it presents significant difficulties in practical deployment. Specifically, transport protocols are often implemented as part of the operating system (OS). Thus to deploy a new TCP variant one will need to modify the protocol implementation in the OS. This is a significant hurdle as there are many different OS implementations in use in Internet servers, ranging from open-source implementations (e.g., Linux and its many variants) to proprietary implementations (e.g., Microsoft Windows). Moreover, given the large number of servers already deployed across the Internet, such protocol upgrade will have to be carried out progressively and thus would take a long time to materialize.

A more subtle yet critical problem is that many of these servers will be serving users from both wired networks and mobile data networks. Thus any protocol updates will either need to work well for both wired and mobile data network users, or be able to distinguish the two types of users so that the proper protocol optimizations can be applied. Neither of these two problems is trivial and further research is required before the feasibility and performance of this approach can be ascertained.

At the other end of the connection is the user's mobile device. Unlike desktop computers, there are significantly more varieties of OS for mobile devices. Therefore deploying protocol upgrades will not be a straightforward exercise. Even if the protocol implementation in the client device can be modified, TCP's performance is still constrained by its congestion control algorithm, which is implemented at the sender-side (i.e., server for most applications).

Motivated by these challenges this work develops a novel alternative solution to the problem – a network-centric

978-1-4577-9538-2/11/$26.00 ©2011 IEEE

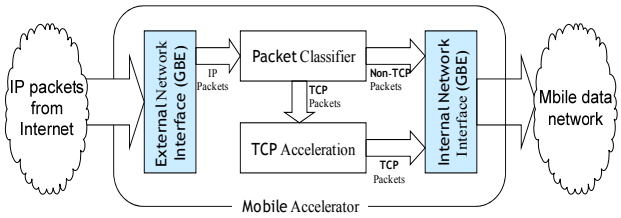

Fig. 1. Architecture of the network-centric mobile accelerator.

approach. Specifically, we propose to realize protocol optimizations in a network-layer device — called a *mobile accelerator*, which is to be situated in the mobile data network so that all TCP traffics to/from mobile clients pass through it as shown in Fig. 1. The mobile accelerator is compatible with existing TCP protocol implementations at both the server and the client, thus eliminating the need for server/client OS modification. It divides a TCP flow into two segments, i.e., from Internet server to accelerator, and from accelerator to mobile client, and then apply a novel opportunistic transmission algorithm to tackle the flow control bottleneck, and a novel rate-based congestion control algorithm to tackle random packet loss and rapid bandwidth fluctuation commonly found in mobile data networks.

Nevertheless this approach does suffer from two limitations. First, to maintain compatibility with existing transport protocols at both ends of the connection, the accelerator is limited to methods which do not require new TCP header fields and protocol handshakes. Second, the mobile accelerator generates and sends ACKs to the sender even before the real ACKs are received from the receiver. This changes TCP's end-to-end semantic which guarantees that data acknowledged by the receiver must have been correctly received. Nonetheless in practice TCP can only guarantee that the acknowledged data have been received into the receiver's transport buffer. Ultimately it is still possible for the receiver *application* to fail receiving the acknowledged data, e.g., the application may crash before it has a chance to read and process the received data. In mission-critical applications it is therefore common to perform integrity check at the application layer instead of relying on the transport and hence the issue is relatively insignificant in practice.

This work has five contributions: (a) we showed that mobile data networks exhibit substantially different characteristics which can seriously degrade the performance of TCP; (b) we developed a novel network-centric approach to protocol optimization which eliminates the problems of compatibility and deployment; (c) we developed a novel opportunistic transmission algorithm to overcome the bottleneck induced by large RTT and small receiver buffer size; (d) we developed a novel rate-based congestion control algorithm to tackle random packet loss and rapid bandwidth fluctuations common in mobile data networks; and (e) we evaluated the proposed mobile accelerator in production 3G/HSPA and LTE networks and reported detailed experimental results to verify the performance gains achievable in real network environments.

Table. 1. Average RTT and packet loss rate at different UDP data flow sending rates.

Sending rate (Mbps)	0.8	1.6	2.4	3.2	4.0	4.8	5.6	6.4
Average RTT (ms)	62	69	68	84	287	792	753	679
Loss (%)	0.02	0.01	0.32	0.14	1.9	27.1	26.5	35.4

The rest of the paper is organized as follows: Section II and III investigate the unique characteristics of mobile data networks, their impact to TCP performance, and our proposed protocol acceleration algorithms; Section IV presents experimental results obtained from production 3G/HSPA and LTE networks to compare the performance gains achievable using the mobile accelerator; and Section V summarizes the study.

II. The Flow Control Bottleneck

TCP's throughput is affected by a number of factors, most notably the link capacity, the round-trip-time (RTT), packet loss rate, and competing traffics. In this section we first investigate the characteristics of RTT in mobile data networks and their impact to TCP performance.

A. Round-trip-time

To characterize the mobile network's properties we developed a measurement tool to send UDP datagrams from a wired-network sender to a receiver connected via a 3G/HSPA modem at controlled data rates. A set of 8 sending rates ranging from 0.8 Mbps to 6.4 Mbps were tested and the corresponding average RTT and packet loss rate measured are summarized in Table 1.

The first observation is that the RTTs at the lower sending rates, e.g., below 4 Mbps, are quite short, e.g., within 100 ms. However as both the sender and the receiver are located in the same region (i.e., without traversing inter-city or international links) the RTT is still significantly longer than its wired counterpart, which typically measures below 10 ms.

The second observation is that the RTT increases significantly when the sending rate is approaching the link capacity. At 4 Mbps the mean RTT already increased to 287 ms although the network was able to sustain the UDP data flow at this data rate. The increases in the RTT are largely queuing time, thereby suggesting that the network bottleneck has a buffer size substantially larger than typical wired network routers. Such large RTT increases the network's bandwidth-delay-product (BDP) significantly. For example, at 4 Mbps with a RTT of 287 ms, the resultant BDP will become 143.5 KB. In a similar experiment conducted in the latest LTE network the BDP can become as large as 769.5KB (c.f. Section IV-B).

In networks with large BDP the performance of TCP will be limited in two ways. First, if the BDP is larger than TCP's transmission window size, which is the minimum of the current congestion window size and the receiver window size, then the link will be under-utilized due to idling periods in waiting for acknowledgement packets to return. This is a well-known problem in high-speed networks and a TCP extension called Large Window Scale (LWS) option [8] had been developed to

address this problem. However, as demonstrated by Wan and Lee [9] in their study of TCP performance over satellite links the LWS option is rarely activated in practice as there is no standard way to activate the option by the network application. Instead almost all network applications tested, ranging from FTP clients to web browsers, adopt the default window size setting.

Now the default receiver window size is operating system dependent and in Windows XP for example, it is a merely 17 KB. Compared to the previously measured BDPs at 143.5 KB for 3G/HSPA and 769.5KB for LTE, this receiver window size is clearly much too small, thereby limiting the transmission window size and consequently the attainable throughput. Although one can modify the existing operating system to use a sufficiently large receiver window size in TCP, the resultant memory requirement may become a problem for applications that make use of large number of sockets (e.g., P2P applications) or for mobile devices with very limited physical memory. With the proposed network-centric approach it is possible to *virtually* enlarge the receiver window size *without* the additional memory requirement.

B. Opportunistic Transmission

The receiver window is employed in TCP's flow control mechanism, which was designed to prevent fast senders from overwhelming slow receivers, leading to receiver buffer overflow. However the processing capacity of computers has progressed significantly so much so that even ordinary personal computers can easily process incoming data at rates measured in tens of Mbps.

In our extensive measurements of the receiver window size in TCP ACK packets when transferring data in a mobile data network or even in wired network, we found that the reported receiver window size of the mobile client stayed at the maximum level (e.g., at the default size of 17KB) nearly all the time. This implies that packets arriving at the receiver were quickly processed and passed to the application even before the next packet arrives, thus leaving the receiver buffer nearly empty at all times.

This motivates us to develop a new opportunistic transmission scheme to take advantage of the receiver's processing power to eliminate the need for a large receiver buffer. Specifically, consider a typical TCP flow where an Internet server (i.e., the sender) transmits data to a mobile client (i.e., the receiver) connected via a mobile data network. The mobile accelerator divides the TCP flow between the sender and the receiver into two segments: a *wired segment* linking the server to the mobile accelerator, and a *mobile segment* linking the mobile accelerator to the mobile client.

On the wired segment the mobile accelerator buffers incoming TCP segments from the server and returns TCP ACKs immediately. More importantly, the receiver window size reported in the ACKs is now determined by the buffer availability at the mobile accelerator rather than the mobile client. As the accelerator runs in computer with abundant memory the reported window size can be much larger (e.g., 2 MB) than the one reported by the actual mobile client.

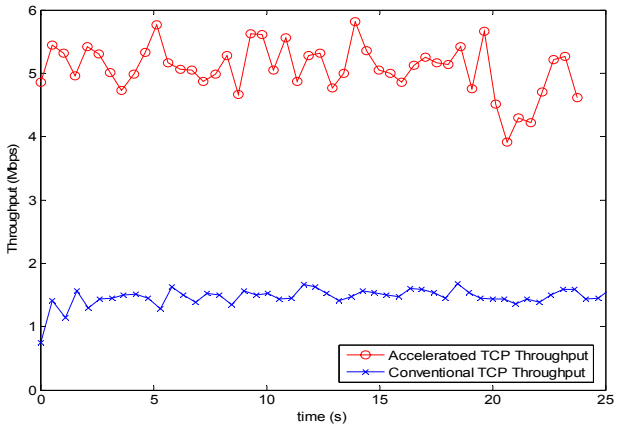

Fig. 2. Throughput comparison between accelerated TCP and conventional TCP.

On the mobile segment the mobile accelerator forwards the buffered TCP segments to the receiver. Normally a TCP sender will not send more data than the receiver's reported receiver window size but in the mobile accelerator packet forwarding is primarily controlled by a rate-based congestion control algorithm (c.f. Section III-B). Unless the reported receiver window size is zero - in which case the mobile accelerator will suspend packet forwarding, the mobile accelerator will forward the buffered TCP segments to the mobile receiver *irrespective* of the receiver window size – *opportunistic transmission*. In practice this means the accelerator will frequently transmit more packets than the receiver window size would have allowed.

The proposed mobile accelerator is similar to split-TCP [10] in splitting a TCP flow into two segments. However split-TCP employed conventional TCP for both segments and thus can only achieve performance gains from shortening the RTT of the two segments and suppressing loss events in the wireless networks. By contrast, the proposed mobile accelerator employs completely different flow and congestion control algorithms that incorporate the characteristics of modern computers and mobile data networks to achieve substantially higher performance gains in the mobile segment.

Fig. 2 compares the throughput of conventional TCP (TCP Cubic [4], the default TCP variant in Linux kernel 2.6.x) and the same TCP flow accelerated by opportunistic transmission. For sake of comparison the mobile accelerator transmits at a fixed maximum rate of 5.6 Mbps irrespective of the receiver window size as long as it is non-zero. The results clearly show that with opportunistic transmission the accelerated TCP flow can achieve significantly higher throughput (~5Mbps) than conventional TCP even though the maximum receiver window size is the same, 17KB in both cases.

Nevertheless, forwarding data at a fixed maximum rate may not work well all the time as the radio bandwidth can and does fluctuate from time to time. We develop a rate-based congestion control algorithm in the next section to tackle this problem.

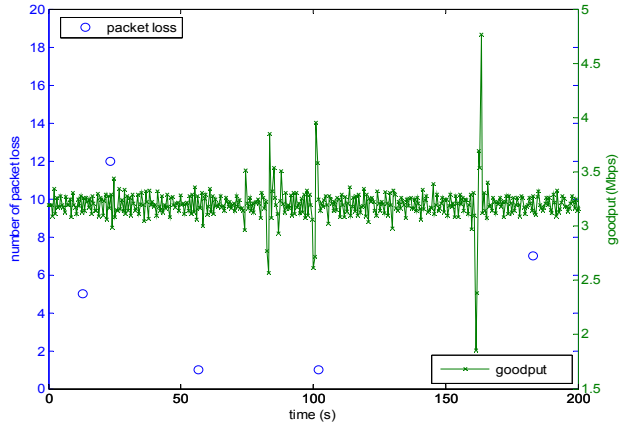

Fig. 3. The time series of (a) 500-ms averaged goodput; and (b) number of lost packets in a lost event, for a UDP data flow at a data rate of 3.2Mbps.

III. Congestion Control Bottleneck

In wired networks packet losses are considered as an indication of network congestion. However, this assumption may not hold in mobile data network as some packet losses are due to radio signal degradation rather than network congestion.

To illustrate this we conducted an experiment to transmit UDP datagrams at a fixed rate of 3.2 Mbps and plotted the 500-ms averaged goodput and packet loss events in Fig. 3. Note that for each loss event the number represents the number of consecutive packets lost in the burst.

One key observation from this result is that the loss events do not correlate with variations in goodput. This strongly suggests that some of the loss events are not due to network congestion. This characteristic of mobile data network may confuse TCP's congestion control algorithm, leading to unnecessary reduction of the congestion window size (CWnd), and consequently the achievable throughput.

For example, TCP Reno [2-3] reduces the CWnd by half after fast retransmit, leading to a substantial reduction in the transmission rate. Worst still, even after fast recovery is completed, TCP will only grow the CWnd additively, which will take considerable time before it can resume the transmission rate to normal. By contrast, the results in Fig. 3 clearly show that the available bandwidth in many cases did not degrade at all even after packet loss events, thus the existing TCP congestion control will likely underutilize the bandwidth available.

More recent TCP variants such as TCP Cubic [4] may perform better in these scenarios as it has a smaller CWnd reduction factor (0.2 versus 0.5), and it grows the CWnd more aggressively using a Cubic function. Nevertheless its congestion control algorithm will still be triggered by non-congestion losses and suffer from throughput degradations.

To quantify this, let r_i be the average throughput of the traffic flow in time interval i. Assume there are k packet loss events, and the j^{th} loss event occurs in time interval h_j, then we define the *throughput loss*, denoted by l_x, as

$$l_x = \left[r_{h_j-1} - r_{h_j} \right] / r_{h_j-1} \quad (1)$$

which measures the ratio of throughput loss after a packet loss event. We can also compute the average throughput loss, denoted by L, over the entire experiment from

$$L = \mathrm{E}\left[l_x \mid \forall x \in 0,1,\cdots k-1 \right] \quad (2)$$

using this metric, the computed average throughput loss for TCP Reno, TCP Cubic, and the UDP flow are 91%, 85.3% and 12.8% respectively. Thus it is clear that the loss events in TCP induce substantially more degradation in throughput than the actual bandwidth available. A number of previous works [5-6] [10] have investigated this problem and proposed various algorithms to differentiate random packet loss from congestion-induced packet loss. It is one of our ongoing works to investigate the effectiveness of these algorithms in modern mobile data networks and our preliminary findings revealed that the performance of these algorithms depends heavily on the network conditions and more work needs to be done to improve their robustness.

By contrast, in the proposed mobile accelerator the mobile segment no longer employs window-based flow and congestion control and as a result, we can decouple loss recovery altogether from transmission rate control to prevent packet losses from triggering false rate reductions.

A. Packet Loss Recovery

Packet loss recovery consists of two parts. The first part recovers packet loss occurring between the TCP sender and the accelerator, and the second part recovers packet loss between the accelerator and the receiver.

For the first part the loss recovery algorithm is similar to conventional TCP, i.e., via duplicate ACKs when packet losses occur. It operates independently from the second part where the accelerator performs retransmission locally. Specifically, the mobile accelerator maintains a list of unacknowledged TCP segments. When three duplicate ACKs are received, the accelerator retransmits the lost TCP segment and suppresses the duplicate ACK, i.e., not forwarding it back to the TCP sender, if the requested lost TCP segment is available in the accelerator.

Otherwise, the TCP segment was lost in the path from the TCP sender to the accelerator, and in this case the duplicate ACKs will be forwarded to the TCP sender for retransmission. In all cases, packet loss events do not affect the rate at which packets are forwarded by the accelerator to the receiver, thus decoupling packet loss recovery from congestion control. In the next section we present the rate-based congestion control algorithm adopted in the accelerator.

B. Rate-based Congestion Control Algorithm

Congestions in mobile data networks differ fundamentally from wired network as each mobile device is allocated a separate network channel that has no other competing traffics. However mobile devices sharing the same cell may still compete for radio resources, subject to the dynamic resource allocation algorithm implemented in the base station. In addition, bandwidth availability is also affected by the radio

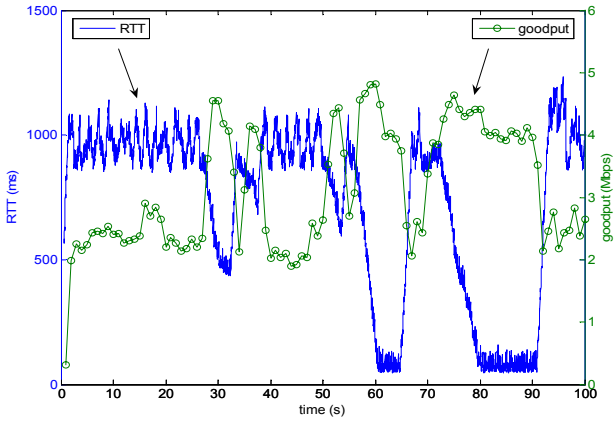

Fig. 4. Correlations between throughput and RTT.

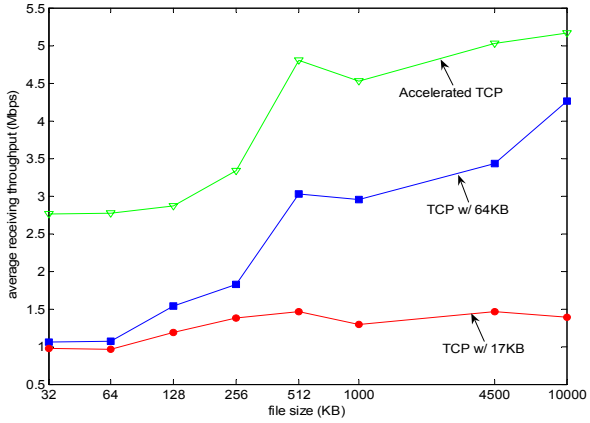

Fig. 5. Comparison of average TCP throughput for TCP Cubic and accelerated TCP.

signal quality and our experiments showed that the fluctuations can be very rapid and unpredictable.

The previous section showed that packet loss is no longer a reliable indicator of network congestion. Our investigations revealed that RTT does correlate with bandwidth availability and hence can be used in the rate-based congestion control algorithm. To illustrate the correlation we conducted an experiment to transmit UDP data over a 3G/HSPA network at a fixed data rate of 4 Mbps. The receiver returned an ACK for every UDP datagram received so that the sender can measure the RTT. Fig. 4 plots the UDP goodput and RTT measured by the receiver and sender respectively. It is clear that goodput and RTT are highly correlated. This correlation is consistent in our experiments, thus motivating us to employ RTT for congestion control.

The proposed rate-based congestion control algorithm has two components: bandwidth estimation and bandwidth adaptation. The accelerator performs an accelerator-to-receiver estimate of the available bandwidth by measuring the average rate of returning ACKs. Specifically, let t_i be the arrival time of ACK i with acknowledged sequence number ack_i. Then for a positive integer k, the estimated receiving bandwidth, denoted by R_i, is computed from

$$R_i = \frac{ack_{i+k} - ack_i}{t_{i+k} - t_i} \quad (3)$$

where the numerator is the amount of data received by the receiver during the time interval $(t_i, t_{i+k}]$. The parameter k controls the duration of the estimation interval (in number of ACKs) and can be adjusted to tradeoff between accuracy and timeliness of rate estimation. The computed R_i's are further smoothed by applying exponentially weighted moving averaging to obtain the estimated bandwidth R:

$$R = (1-\lambda)\times R + \lambda \times R_i \quad (4)$$

where λ is the smoothing factor.

Bandwidth adaptation is triggered by two RTT thresholds α and β, $\alpha<\beta$, and controlled by a transmission rate limit R_{max}. If the measured RTT exceeds β, then the accelerator will trigger congestion avoidance and set R_{max} to the estimated bandwidth R according to (4). If the measured RTT is less than α, then the network is not congested and the accelerator will increase R_{max} according to:

$$R_{\max} = \mu \times R_{\max} + (1-\mu)\times R_{cap} \quad (5)$$

where R_{cap} is the link capacity of the mobile data network. Note that (5) will not increase the transmission rate limit beyond the mobile data network's link capacity, which is known to the mobile accelerator. This can avoid the periodic congestions caused by conventional TCP's bandwidth probing actions.

The parameter μ ranges from 0 and 1, and is used to control the rate of transmission rate increase. Unlike conventional TCP, which increases the rate gradually to maintain fair bandwidth sharing with competing flows, the accelerator can increase the transmission rate more aggressively to improve bandwidth utilization. Fairness among TCP flows sharing the same radio network channel, i.e., TCP flows destined to the same mobile device, can be maintained by the accelerator using simple round-robin transmission scheduling.

In addition to network bandwidth constraint, in some cases the achievable throughput may also be limited by the mobile device's processing capacity. This is especially the case in mobile handsets. As the bandwidth estimation algorithm is based on ACKs returned by the receiver, it inherently accounts for the receiver's processing limit in case it is the bottleneck.

The above algorithm's processing complexity is similar to conventional TCP. More importantly the processing is done independently on a per flow basis and thus its complexity increases only linearly with the number of flows in the system.

IV. Performance Evaluation

We developed a software-based implementation of the mobile accelerator described in Section II and III. In this section we compare the throughput performance of TCP with and without the mobile accelerator in a production 3G/HSPA network and in a pre-production LTE network. Table 2 summarizes the parameters adopted in the experiments.

A. Performance over 3G/HSPA Network

The experiment setup consisted of a Linux server with

kernel 2.6 (with the default TCP Cubic congestion control module), connected via the mobile accelerator (running on Windows XP SP2) to the Internet via a wired network link running at 1Gbps. The receiver host ran Windows XP SP2 and was connected to the mobile data network via a USB 3G/HSPA modem supporting up to 7.2Mbps downlink bandwidth. When the mobile accelerator is deactived the accelerator will simply forward packets to-and-from the server without any processing. All experiments were conducted with the receiving host in a stationary position.

Fig.5 plots the average TCP throughput in downloading a file ranging from 32KB to 10MB from the apache web server running at the server to the browser running at the receiver. Three sets of experiments were conducted: (a) non-accelerated TCP with default window size of 17KB; (b) non-accelerated TCP with the window size manually increased to 64KB; and (c) accelerated TCP with default window size of 17KB.

The results show that the mobile accelerator can increase TCP throughput performance significantly. The improvement is most significant for smaller file sizes. This is because the rate-based congestion control algorithm does not suffer from slow ramp-up of the transmission rate, and is more effective in utilizing network bandwidth as discussed in Section III.

Moreover, even with the default window size of 17 KB, the accelerated TCP can still achieve throughput close to the network's capacity for large file sizes. This demonstrates the effectiveness of the proposed opportunistic transmission scheme in resolving the receiver window size limit.

We also evaluated and compared the performance of two newer TCP variants, namely TCP Westwood [5] and TCP Veno [6], which were also designed for wireless networks. In this experiment we developed a custom sender application to transmit data over TCP to the receiver at a controlled data rate from 0.8Mbps to 5.6Mbps. As summarized in Table 3, at the lower data rates of 0.8Mbps and 1.6Mbps the throughput performances are similar for all protocols and the accelerated TCP. The performance gap widens quickly at higher data rates, e.g., beginning at 2.4Mbps, even when the network is likely to be not congested. The improvement is largely due to the opportunistic transmission algorithm as the network's RTT is load-dependent and increases rapidly at higher data rates.

B. Performance over LTE Networks

Long Term Evolution (LTE) [7] is an emerging standard for mobile data networks that can offer downlink peak bandwidth in excess of 100Mbps. We conducted a series of experiments in a pre-production LTE network using the same experimental setup as described in Section A.

In the first set of experiments we aimed to measure the performance characteristics of the LTE network. Using the same custom measurement tool we send UDP datagrams at a fixed rate from the Linux server to a notebook equipped with a LTE USB modem. Table 4 summarizes the measured RTT and packet loss rate at sending rate ranging from 1.6 Mbps to 80 Mbps. There are two observations.

Table. 2. Parameters adopted in the experiments.

Symbol	k	λ	α	β	μ	R_{cap}
Value	250	0.2	140 ms	260 ms	0.8	5.6Mbps for 3G 100Mbps for LTE

Table 3. Comparison of TCP throughput performance under different source sending rates

Sending Rate (Mbps)	Receiving Throughput (Mbps)			
	Accelerated TCP	TCP Cubic	TCP Westwood	TCP Veno
0.8	0.80	0.80	0.8	0.79
1.6	1.60	1.56	1.55	1.60
2.4	2.39	1.50	1.45	1.60
3.2	3.19	1.48	1.49	1.55
4.0	3.94	1.56	1.45	1.60
4.8	4.50	1.58	1.49	1.48
5.6	5.19	1.63	1.65	1.58

Table. 4. RTT and loss rate over LTE for various data rates.

Sending rate (Mbps)	1.6	4.0	8	16	28	36	40	80
Average RTT (ms)	25	34	38	51	66	171	964	1466
Loss ratio (%)	0.01	0.01	0.8	3.7	5.5	10.7	11.8	34.8

Table. 5. Comparison of TCP throughput performance under different radio signal conditions over LTE.

{RSRP, SINR}	{-64, 27}	{-81, 27}	{-90, 27}	{-110, 15}
Accelerated TCP throughput (Mbps)	74	73	53	47
Normal TCP throughput (Mbps)	34	36	30	27
Maximum UDP goodput (Mbps)	81	80	55	50

First, while the RTT is very short at low data rates (e.g., 25 ms at 1.6 Mbps), it begins to increase significantly even at medium data rates (e.g., 171 ms at 36 Mbps). Consequently the resultant bandwidth-delay product will become very large (e.g., at 36 Mbps the BDP is 769.5KB) and the default TCP window size will become a severe bottleneck to throughput performance. Therefore in the following TCP throughput experiments we manually increased normal TCP's receiver window size to 1024KB to mitigate this bottleneck.

Second, the packet loss rate is not insignificant even at low data rates (e.g., 3.7% at 16 Mbps) and increases rapidly with higher data rates. This suggests that the performance of TCP could be severely degraded by the frequent packet losses which trigger congestion control.

This is confirmed by our TCP throughput measurements in Table 5. We conducted 4 sets of experiments by locating the mobile receiver in different physical locations such that different levels of radio signal quality were obtained. The radio signal quality is measured using two parameters: Reference Signal Receiving Power (*RSRP*) [7] and Signal to Interference plus Noise Ratio (*SINR*) [7]. Larger values represent better signal quality in both parameters. In each experiment the receiver downloaded a 68MB file via the LTE network from the Linux server.

The results in Table 4 clearly show the performance improvement achievable with the mobile accelerator under

different radio signal conditions. At the highest signal quality of {-64, 27} the accelerated TCP throughput reached 74 Mbps while normal TCP can only achieve 34 Mbps. Moreover, in each signal condition we also used UDP flow to measure the maximum goodput achievable. Comparing this to the accelerated TCP throughput we can see that the mobile accelerator can raise TCP's performance close to the network capacity limit, thus efficiently utilizing the large amount of bandwidth available in the emerging LTE networks.

V. CONCLUSION

This work shows that the existing TCP is far from optimal when used in the emerging high-speed mobile data networks. By applying different flow and congestion control algorithms to the wired and mobile network segments, the proposed mobile accelerator can effectively resolve both the flow and the congestion control bottlenecks to enable TCP to efficiently utilize the underlying network bandwidth. More importantly, this network-centric approach does not require modification to the existing server/client OS, nor the network applications, and thus can be readily deployed in today's mobile data networks to accelerate all traversing TCP traffics.

REFERENCES

[1] The 3rd Generation Partnership Project (3GPP), Available: http://www.3gpp.org.

[2] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control," *Request for Comments 2581*, April 1999.

[3] S. Floyd and T. Henderson. "The New Reno Modification to TCP's Fast Recovery Algorithm," *Request for Comments 2582,* April 1999.

[4] S. Ha, I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *International Workshop on Protocols for Fast and Long Distance Networks*, 2005.

[5] S. Mascolo, C. Casetti, M. Geria, M. Y. Sanadidi and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *in Proceedings of ACM SIGMOBILE*, July 2001.

[6] C. Fu, S. Liew, "TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks," in *IEEE Journal on Selected Areas in Communications*, Vol. 21, 2003.

[7] "Long Term Evolution (LTE): A Technical Overview," Motorola, July 2010.

[8] V. Jacobson, R. Braden and D. Borman, "TCP Extensions for High Performance," *Request for Comments 1323,* May 1992.

[9] Y. B. Lee, T. S. Yum and W. S. Wan, "TCP-SuperCharger: A New Approach to High-Throughput Satellite Data Transfer," in *Proceedings of 27th International Symposium on Space Technology and Science*, Epochal Tsukuba, Tokyo, Japan, July 2009.

[10] H. Balakrishnan, V. Padmanabhan, S. Seshan and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," in *Proceedings of IEEE/ACM Transactions on Networking*, Vol. 5, Dec 1997.