Mobile TCP Evaluation

EECS-395/495 - Networking Problems in Cloud Computing
Final Paper

Alexander Ayerdi, Muhammed Mas-ud Hussain, Kamalakar Kambhatla

1. Introduction

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite (IP) and resides at the transport layer. Web browsers use TCP when they connect to servers on the World Wide Web, deliver email, transfer files and perform many other jobs. However, TCP is not particularly designed for the mobile environment. For this reason, TCP encounters high packet loss rate and unstable throughput in this setting. The Number of broken connections also increases. The principle goal for this project is to evaluate TCP in mobile environments (WiFi and Cellular-Data). We aim to thoroughly analyze the obtained results and come up with the factors behind TCP's slow mobile performance. However, there are many factors as to why TCP performance in mobile environments is slow, so we focused on one main problem, namely TCP connections with Cellular-Data in a moving environment. We used a combination of a custom Ruby script and Wireshark to analyze the results.

The following questions are addressed in our analysis:

1) What is the comparative performance of TCP in wired, wi-fi and mobile networks?
2) What are the main reasons for slow TCP performance in mobile networks?
3) How TCP performance is affected by the mobility of cellular devices?
4) Can we suggest a plausible fix to the problem with moving Cellular-Data connections?

In order provide a proper analysis of the above questions, we decided to leverage real world data. We collected "pcap" files in varying experimental setups: PC-WiFi, PC-Wired, Mobile-WiFi, Mobile-Data, Mobile-Data on moving train. From these raw data files, we extracted structured data: packet loss, throughput, RTT, packets sent per second, packet retransmissions, out of order packets, and amount of connection resets. We then used these data to come up with valid answers to the above questions.

Our initial conclusions were that there is a significant decrease in performance in both WiFi and Cellular-Data for mobile phones. We found out that TCP was appearing to think that there was congestion in the connection, so RTT was increasing and connections were being reset often. In addition, when on Cellular-Data connections while moving, the connection is unbearably slow and barely transferred any data during the TCP connection. We propose that a possible solution or alleviation to this problem would be to include state transferring protocols inside of the cell towers just as if they were WiFi routers.

2. Motivation

Accessing mobile internet (3G/4G) is getting popular day-by-day. Smart phones, tablets and subsequently mobile internet is becoming available to people more easily with lesser cost. From [1], in Figure 1, it is projected that by 2014 world wide mobile internet users will be more than desktop internet users. This claim is backed up in Figure 2 (from [2]). By 2012 in China, number of mobile internet users has surpassed the number of desktop internet users.

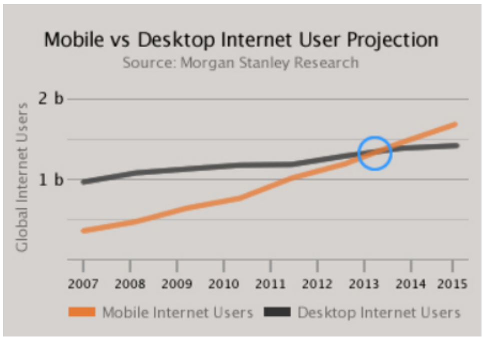


Figure 1: World-wide mobile vs desktop internet user projection

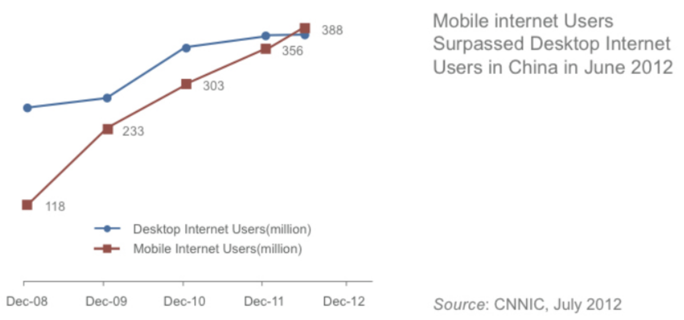


Figure 2: Mobile internet users surpassed desktop internet users in China in 2012

Thus, slow performance of TCP under mobile environment is becoming more and more a bottleneck for end users. As, TCP has performed well and most popular in other environments, we will definitely be benefitted if we can use TCP or similar solution in mobile environments as well. Also, HTTP, HTTPS, SMTP, POP3, IMAP, SSH, FTP, Telnet and a variety of other protocols are typically encapsulated in TCP. This too works as a motivation behind our goal for TCP mobile evaluation in this project.

We also focused on how TCP performs while we are moving and using cellular-data simultaneously. We show the data collected by US Census of Bureau in 2009 [12] in Figure 3. We can see average commute time to work for people in US is 25.2 minutes (one way) in 2009, which almost remained same as 2000. According to the Bureau's report [12], there are 600,000 mega-commuters (Who travels more than 90 minutes to work in one way) in America. If we decrease the travel time to 60 minutes (one way), the number jumps up to 10.8 million — a little over 8% of U.S. workers. Not only commuting to work, we have to be on vehicle for a lot of other purposes as well - going to the store, going to school, dropping off kids, etc. In Figure 4 (from [11]), it i showed that people typically spend 18.5 hours per week in moving vehicles, which is 11% of the whole available time (7 * 24 hours). Thus, people might have to use cellular internet for a significant amount of time while being in a moving vehicle. From an end user's perspective, performance of internet during this period can not be overlooked. This worked as our motivation behind deciding to evaluate and analyze TCP performance with cellular-data on a moving train.

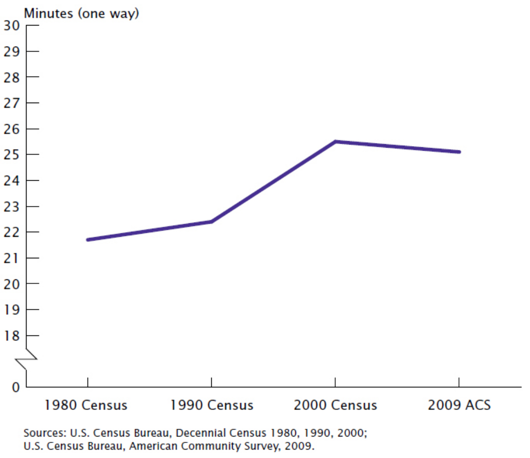


Figure 3: Avg. Commute to Work Time in US (2009)

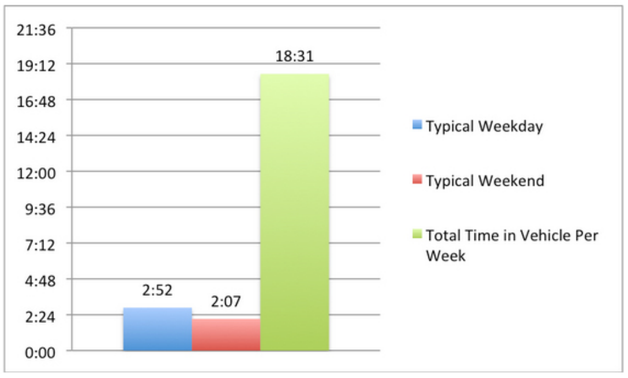


Figure 4: Time spent in vehicle per day/week for average people in US

3. Preliminaries

3.1 TCP

The Transmission Control Protocol (TCP) is the transport layer protocol of TCP/IP suite. TCP provides a reliable communication service at an intermediate level between an application program and the Internet Protocol (IP). Due to network congestion, load balancing, or other unpredictable network behavior, IP packets can be lost, duplicated, or delivered out of order. TCP detects these problems and solves them by requesting retransmission of lost data and rearranging out-of-order data. TCP even helps minimizing network congestion to reduce the occurrence of the other problems.

TCP leverages an end-to-end flow control protocol to avoid having the sender send data too fast for the receiver to receive and process it reliably. This kind of flow control mechanism is necessary in an environment where machines of diverse network speeds communicate. Another important aspect of TCP is its congestion control. TCP uses a number of mechanisms to achieve high performance and avoid congestion collapse, where network performance can degrade by several orders of magnitude. These mechanisms control the rate of data entering the network, keeping the data flow below a threshold that would trigger collapse. As an example, acknowledgments for data sent, or lack of acknowledgments, are used by senders to infer network conditions between the TCP sender and receiver, coupled with timers. Then, TCP senders and receivers can alter the behavior of the flow of data.

3.2 Mobile Internet (Cellular-Data)

'Mobile Internet' means accessing the Internet via a cellular telephone service provider. It is wireless access that can handoff to another radio tower while it's moving across the service area. We also can refer an immobile device that stays connected to one tower as mobile internet. Usually, cellular base stations are more expensive to provide than a wireless base station. A wireless base station connects directly to an internet service provider, whereas cellular base stations connect through the telephone system. Moreover, a laptop with a broadband modem and a cellular service provider subscription, that is traveling through the city is on mobile Internet. A mobile broadband modem "tethers" the smartphone to one or more computers or other end user devices to provide access to the Internet via the protocols that cellular telephone service providers may offer.

4. Experimental Setup & Method

The first step is to build tools for TCP analysis. We built a comprehensive Ruby script that was compiled into a Ruby Gem (similar to a C library). This was used to compute RTT and throughput data, as well as the CDF graphs. Any user has the ability to clone the tool from github and run the script on their own computer. In order to test on mobile environments, we have to create a pcap file from a TCPDump, transfer to the computer and then run the second half of the Ruby analysis on the already generated pcap file. Also, we used Wireshark after gathering the pcap file in order to generate the packet loss, packet retransmission rate, TCP congestion window recalculations, out of order packets, and packets sent per second information. It’s notable to mention that to gather our data we always used the same YouTube video. Also, the WiFi connections were all made from the same WiFi router connection. We also gathered at least 3 samples for each environments (cellular-data, cellular-WiFi, pc-WiFi, pc-wired, cellular-data on moving train) and then summed up the values from all the samples in our final results.

4.1 PC Method

The first step we had to do was to run the Ruby script by using the IRB Ruby interpreter. We use the TCPAnalysis.run method and you need to enter in the correct LAN port that you’ll want to listen from when prompted. In most cases this will be ‘wlan0’ or ‘eth0’.
Then we chose to do a ‘Packet Amount’ data gathering for 20,000 packets (the script will gather data for 20,000 packets). We started a Youtube clip and then started the script by hitting return. We had the option to do a timed data gathering as well (say, the script will gather data for 200 seconds). After the data was gathered, the script automatically sorts everything into specific directories. We were particularly interested in the RTT and Throughput data. The related graphs were generated automatically too via Gnuplot. We had to manually create the CDF graphs, simply by changing some settings on the GNUplots generated. The experiment was done using Wired and WiFi connections.

4.2 Mobile Method

The first step we had to do was to root an Android phone. Once that was completed then we had to install an internet hotspot that the computer could connect to via the mobile phone. We needed to be sure that the phone was not using WiFi to connect to the internet if we were testing

Cellular-Data connections. Then we ran the Ruby script but changed the listening port, which can be found by typing *ifconfig* on the rooted Android phone's terminal. We used TCPAnalysis.run and ran the YouTube clip, again until 20,000 packets were collected. We gathered the same data as for the PC Method. The experiment was done using WiFi and Cellular-Data connections.

4.3 Mobile on Train Method

This method is the same as the *Mobile Method* without loss of generality. Except the only difference is that the experiment is done in a constantly moving vehicle (CTA train). We did our experiment on the train, although we may have done it in a car or any other vehicle as well. We chose to not worry about the variable of a car that would be stopping frequently at traffic stops.

4.4 Wireshark

In order to get the rest of the analysis, we took the pcap files generated from these experiments and inserted them into the Wireshark tool. With wireshark we simply had to pull the specific graphs and tables that showed packet loss, out of order packets, packets sent per second, and all the other pieces explained above.

4.5 Ruby Script in Depth

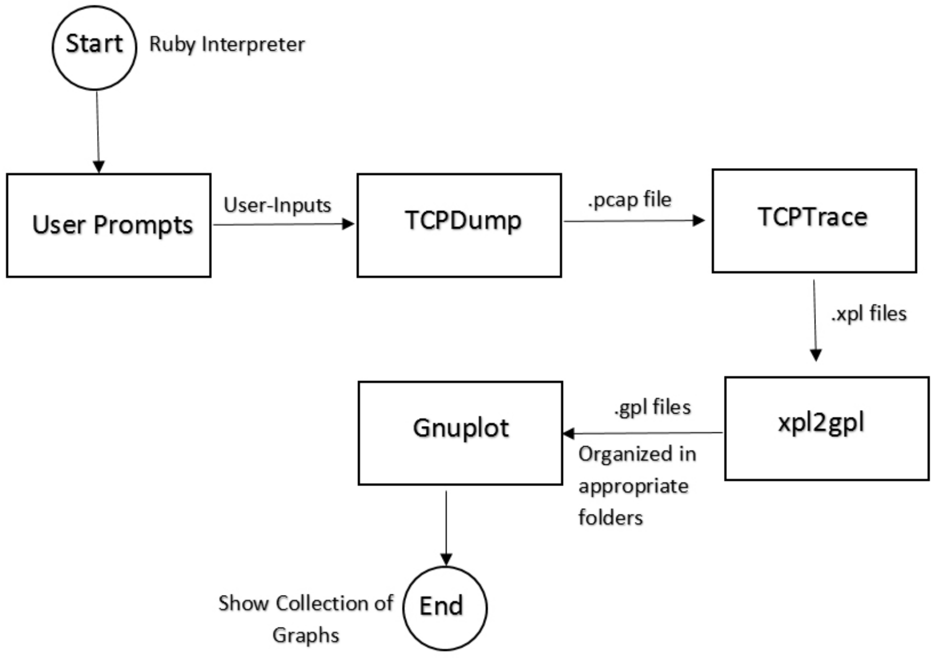


Figure 5: Flow of Ruby Script

Our Ruby script does the following things in order (as shown in Figure 5):

1. Prompts the user to enter user-desired inputs regarding when to terminate data collection and what level of data analysis to perform.
2. Runs TCPDump to gather packet capture data.

3. Runs TCPTrace to perform analysis on the collected pcap file and generate xplot graphs.
4. Uses a tool xpl2gpl to convert xplot files (.xpl) to gnuplot files (.gpl files).
5. Runs gnuplot to show the collection of generated gpl files.

TCPDump can perform two types of controlled analysis. The user can specify whether or not to do a 'Timed' analysis or a 'Packet Amount' analysis. With a 'Timed' analysis, TCPDump runs for a specified number of seconds and then returns the data gathered after the time has expired. The 'Packet Amount' analysis runs TCPDump until a specified number of packets are received and then saves the generated pcap file.

TCPTrace can be run with different parameters such as -T, -S, -G, etc. Users can specify which graphs they want by entering the necessary alphabet when prompted. Let, a user wants TCPTrace to plot only throughput graphs (-T option). Then, when prompted, he can enter 'T' and our ruby script will ensure that only throughput graphs are plotted.

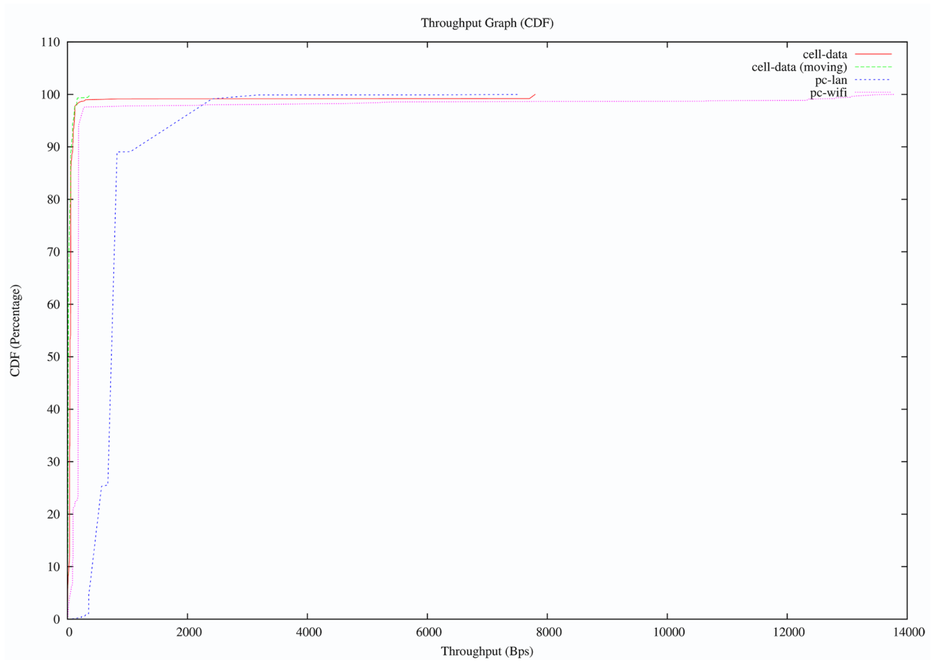
All commands are run in the Ruby Interpreter, which in turn runs bash commands to perform the TCPDump, TCPTrace, xpl2gpl and GNUPlot applications. The Ruby application is installed via its gem library and the user can run from any Ruby Interpreter, granted the host has TCPTrace, TCPDump, GNUPlot and Ruby 1.9.1 installed.

5 Experimental Results

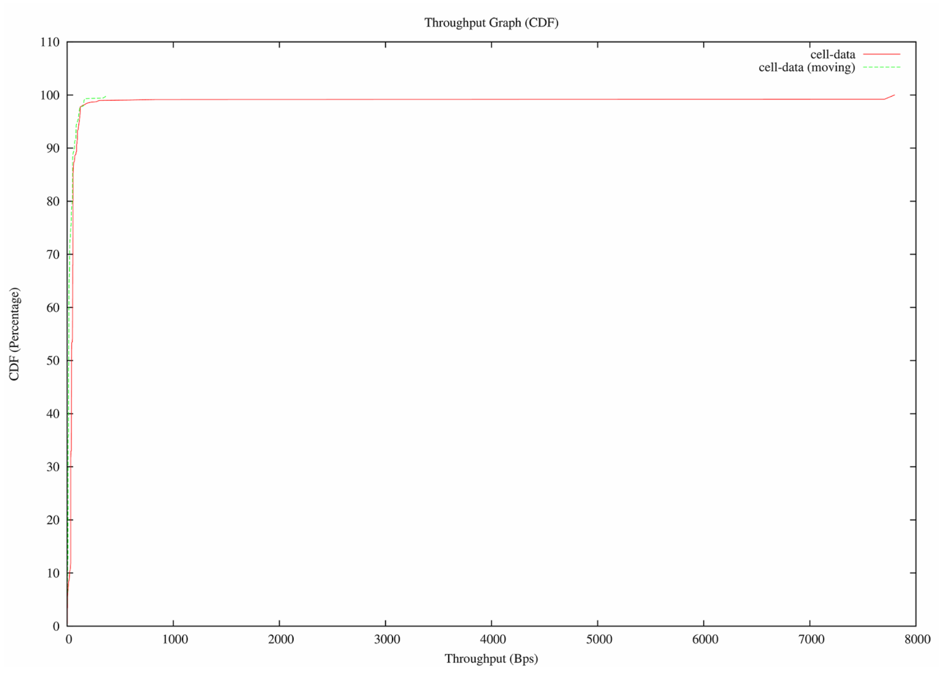
The following graphs and discussion are from our five experiment scenarios. Each graph shows a comparison of all five and then a related graph showing only mobile Cellular-Data versus mobile Cellular-Data-While-Moving.

5.1 Throughput Graph

A CDF graph for throughput is given in Figure 6 for cellular-data, cellular-data (moving), pc-lan (wired) and pc-WiFi environments. In Figure 6(a), we can see both of the pc environments outperform cellular-data. As expected, pc-lan has the highest throughput cdf, followed by pc-WiFi. In Figure 6(b), we observe that TCP performance for cellular-data degrades considerably when we're on a moving train. Throughput is near 0-10 Bps for around 70% of packets for cellular-data on a moving train. When not moving, this value comes down to below 10%.



(a)

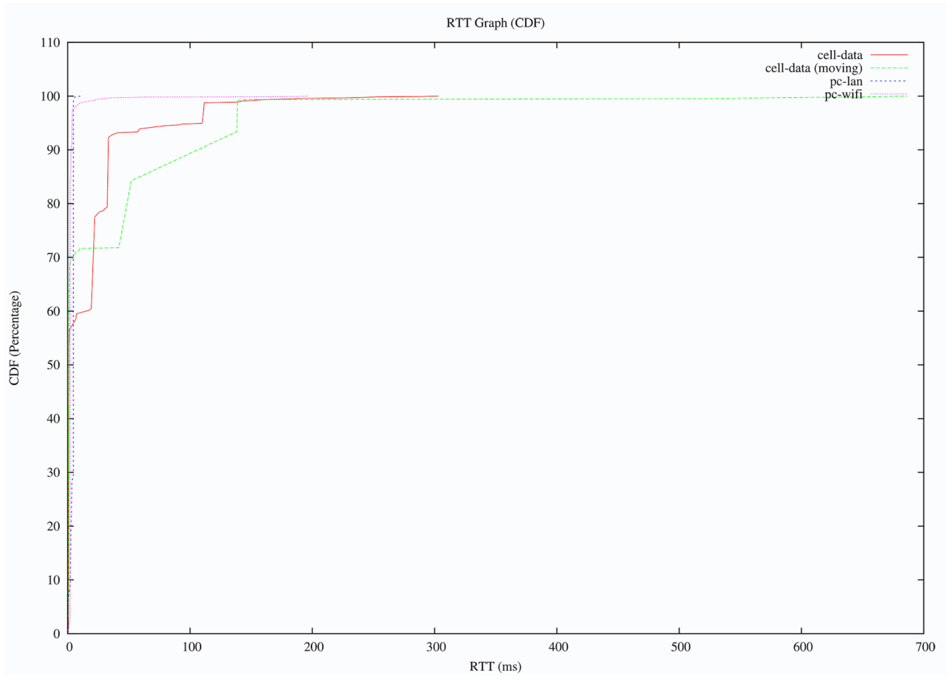


(b)

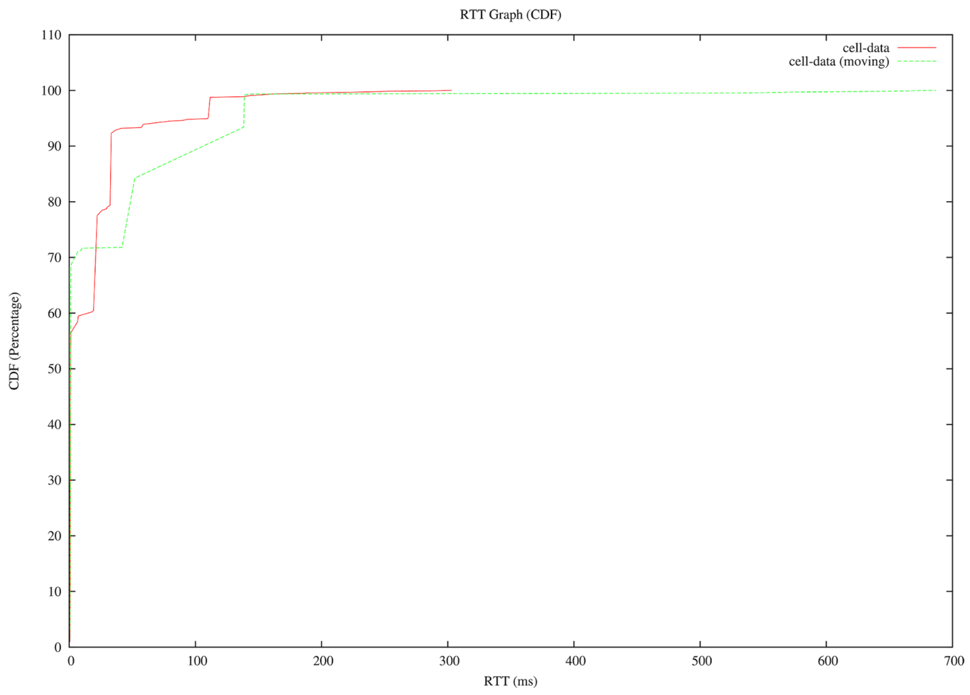
Figure 6: Throughput CDF graphs

5.2 RTT Graph

A CDF graph for RTT is given in Figure 7 for all the concerned environments. The results is similar to the throughput graphs. In Figure 7(a), pc environments have significantly lower RTT CDF values (between 0-10 ms) than cellular-data environments (meaning better performance). Figure 7(b) compares the RTT values for immobile cellular-data and moving cellular-data. Until around 60% mark they are relatively equal. Between 60-70% static cellular-data connection have higher RTT (less performance), but after that moving cellular-data connection has higher RTT values and thus performs worse.



(a)

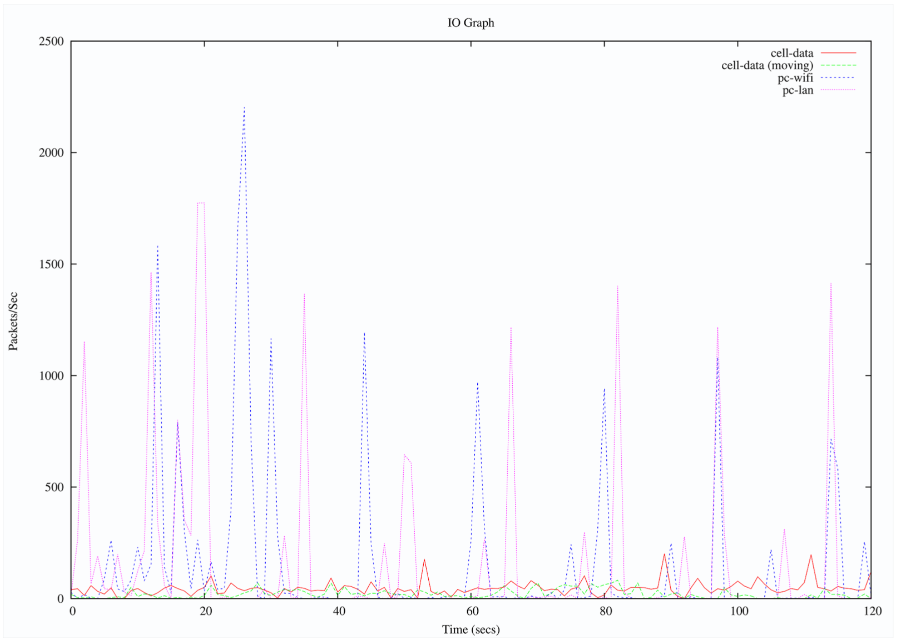


(b)

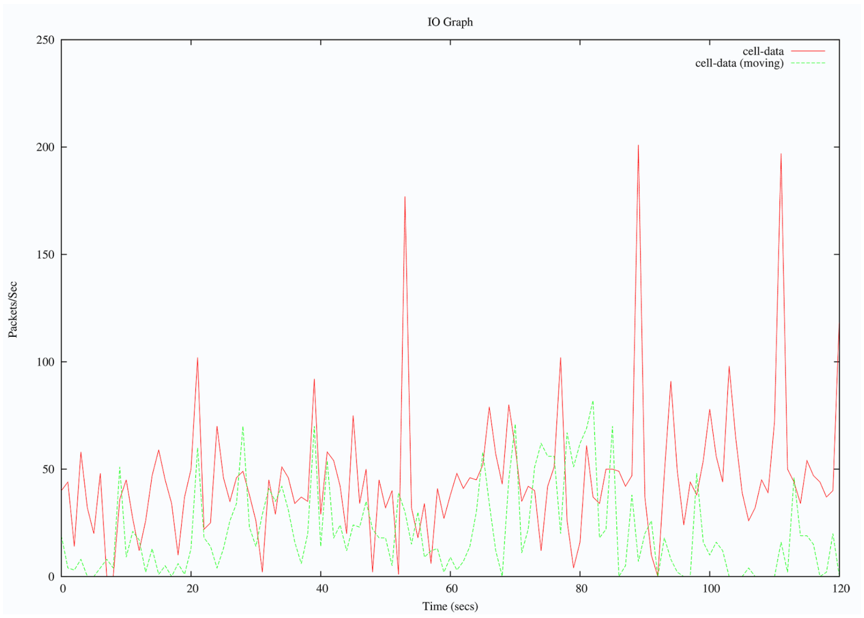
Figure 7: RTT CDF graphs

5.3 IO Graph

We obtained the IO graphs for the same environments using Wireshark. An IO graph plots packets/sec. (both sent and received) vs time data. As RTT is low and throughput is high for pc environments, they have the higher packets/sec values, reaching as much as ~2250 packets/sec. Meanwhile, non-mobile cellular-data connection has maximum value of only ~200 packets/sec. Moving cellular-data connection has the lowest value as they have high RTT and low throughput values. The maximum packets/sec. value recorded in this setting is 75.



(a)



(b)

Figure 8: IO Graphs

5.4 Retransmitted Packets

Table 1 shows the number of packet transmitted for all the samples of each environments. Cellar-data on a moving train shows the highest percentage of packet re-transmission (as high as 3.9% for one sample). Cellular-WiFi also has a small percent of packets retransmitted, but it is quite less than cellular-data samples. None of the pc environment samples show any sign of packet retransmission (0% packets retransmitted in all samples).

Environment	Sample	Packet Retransmitted (percentage)
Cellular-Data	1	**0.85%**
"	2	**0.7%**
"	3	**1.2%**
Cellular-Data (Moving)	1	**3.9%**
"	2	**3.1%**
"	3	**2.7%**
Cellular-WiFi	1	**0.4%**
"	2	**0.2%**
PC-WiFi	all	**0%**
PC-Lan	all	**0%**

Table 1: No. of retransmitted packets for all environments

5.5 Packet Losses, Out of Order Packets, Connection Resets

We observe similar trends in Table 2 as well. Higher percentage of packets are lost in cellular-data environments, while cellular-WiFi performs nearly as well as pc environments. PC environments have minimal out of order packets (0,1), while cellular-WiFi has the most. It is expected, as previous research works showed out of order packets are seen more in WiFi environments. Another important observation is that, TCP connection resets occur 5-10 times more in cellular-data environments, than other pc environments and cellular-WiFi environment. The connection reset percentage is most in case of moving cellular-data connections.

Environment	Packet Loss (Percentage)	Out of Order ckets (over all samples)	Connection Reset (Percentage)
Cellular-Data	0.387%	4	0.601%
Cellular-Data (Moving)	0.39%	8	0.806%
Cellular-WiFi	0.215%	17	0.111%
PC-WiFi	0.096%	1	0.128%
PC-Lan	0.195%	0	0.07%

Table 2: Various data for all environments

6 Evaluation and Analysis of Results

Packet retransmissions, connection resets, high RTT and all the other slowing effects seen in cellular-data environments above are mostly caused by TCP congestion control and flow-control protocols. Essentially, TCP is seeing the mobile environments as having "congestions" when that is not necessarily the case. Where we see high RTT times leads to the conclusion that actually the mobile connections are just traversing much more complex terrain since they need to connect through a cell tower, as well as handle radio connection protocols, and switch to a different tower often based on cell tower load balancing. Also, underlying radio connection for cell phones is flaky. This complex traversal and flaky radio connection is seen as congestion to TCP. Therefore TCP reacts with a higher wait time to retransmit, and slows the connection handshake exponentially as time moves on as well as having to recalculate the congestion window each time a new cell tower is encountered.

A possible mitigation to this problem is state transfer and using indirect TCP sockets [8]. This is something that would be able to at least make the connections while moving no different than a normal cellular connection or even a mobile WiFi connection. This protocol has already been tested on WiFi connections, so it has a high chance of improvement [8]. Surely the TCP state would be able to transfer to the adjacent cell tower that you need to connect to while moving, but the packets still need to be transferred again anyway. This is not something that is even really solved in WiFi connections yet, so we leave that for further research in the

WiFi field.

7 Related Work

Very little research has been conducted in this area and any solutions identified would be paving new ways. In [3], it is discussed that how TCP's congestion control algorithm can backfire in mobile environments, especially considering varying signal power inherent with mobility and handover between base-stations. Authors proposed a TCP modification namely TCP-L, which allows an application to increase its performance by making a reliability trade-off. Although, the work lacks proper evaluation and validation of proposed TCP-L.

Performance evaluation of TCP over wireless networks is done in [4]. In wireless networks, similar to mobile internet, performance of TCP degrades due to faulty congestion control scheme. Authors measure the protocol performance based on FTP transmission delay experienced by the client, the FTP throughput, and the packet discard ratio with changes in the TCP segment size and burstiness of the channel. [5] discusses the effect of high RTT times and mobility on the poor performance of TCP in terms of network intensive mobile apps.

In recent times, [6] uses a novel loss recovery mechanism to improve TCP performance, which may be used under mobile environment as well. Meanwhile [7] explores and evaluates the possibility of extending TCP. They conclude that TCP can still be extended, but extensions' design is very constrained. In [10], the author briefly discuss why TCP performance is not up to the mark in mobile apps and devices. The author suggests that conflict between TCP's congestion control and 3G data connection's own attempt for reliable delivery is the reason behind this. [13] states that TCP's flow control features perform less than optimally over wireless interfaces. A number of TCP optimization techniques have been studied to enhance the TCP performance for various wireless environments. The authors of [13] propose a profile of such techniques, particularly effective for to 2.5G/3G wireless networks

8 Conclusion

Overall, our aim in this work was to analyze TCP performance in various settings, specially focusing on cellular data connections. We showed that TCP performs worse in cellular data connections than any other pc environments. We also demonstrated that moving mobile cellular-data connections are significantly worse than any other connection type and thus it is an area that is in vast need of improvement. We propose one possible solution but ideally we want to encourage more work to be done in the area of state transfer and indirect TCP sockets in the mobile cellular data environment.

8.1 Problems/Challenges

- We faced problems to obtain a rooted cellphone initially, as a rooted cellphone is void of any warranty from the makers. In the end, one of us agreed to root an android phone and do the experiments on that phone.
- Collecting a good amount of TCP packets (20,000 for our experiments) was tough for cellular-data on a moving train, as the youtube video barely played. It took around a 45 minute train journey to capture desired amount of TCP packets in this setting.
- We could not do a comparative analysis of moving WiFi and moving cellular-data, as we did not have the devices and equipments to access WiFi internet while moving.

8. 2 Future Work

- Test out the mobile moving cellular connections more extensively. If possible, compare its performance with moving cellular WiFi connections.
- Test out moving WiFi environment for both PC and Mobile.
- Do experiments to find out if our suggested fix improves performance in moving cellular data environment.
- Research more possible solutions to solve the moving mobile connection problems.

References:

[1] http://www.cambiolabs.com/_lp/palm_your_site.php
[2] http://www.techinasia.com/china-mobile-internet-2012/
[3] "TCP in Wireless Networks: Challenges, Optimizations and Evaluations" - Karlstad University Studies
[4] "Performance Evaluation of TCP Over Wireless Links" - Praveen Sheethalnath, Sonal Ramnani, Shalaka Tendulkar, Laurent Rival
[5] http://blog.davidsingleton.org/mobiletcp/
[6] Reducing Web Latency: The Virtue of Gentle Aggression, Tobias Flach et. al., Sigcomm. '13
[7] "Is it Still Possible to Extend TCP?", Michio Honda et. al., Sigcomm. '11
[8] http://www.ccs.neu.edu/home/rraj/Courses/6710/S10/Lectures/TCPForWireless.pdf
[9] "TCP over 2.5G and 3G wireless networks" - Dongmei Zhang , Runtong Zhang, Zhigang Kan, Renaud Cuny, Jussi Ruutu, Jian Ma
[10] Why mobile apps suck when you're mobile
[11] http://giantpromotions.com/sale-in-a-box-advertising/
[12] Commuting in the United States : 2009 - United States Census Bureau