

Abstract

This project investigated the viability of data flows over the Worcester, MA. Sprint PCS EvDO network by building an experimental test bed between the WPI campus and the Sprint network. A custom-built bandwidth estimation tool yielded consistent estimates of available capacity of approximately 500 kbps over the EvDO network. Measured TCP throughput was slightly higher on weekdays than weekends, ninety percent of measured round trip times were below 210 ms, and performance analysis produced no significant time of day correlation.

Acknowledgments

The authors would like to thank Professors Mark Claypool and Robert Kinicki for their guidance and advice throughout the project, as well as Mingzhe Li for his helpful comments and assistance. The authors would also like to acknowledge Constantinos Dovrolis, whose program *Pathrate* was adapted for this project.

Contents

1	Introduction	1
2	Background	3
2.1	CDMA Overview	3
2.1.1	EvDO Networks	3
2.2	Related Work	4
2.2.1	Bandwidth Estimation Techniques	4
3	Methodology	6
3.1	Equipment Testbed	6
3.2	Tools	7
3.2.1	<i>TCPDump</i> , <i>TCPTrace</i> , and GNU <i>wget</i>	7
3.2.2	ICMP <i>ping</i>	7
3.2.3	<i>GNUplot</i>	8
3.2.4	<i>npath Experimental</i>	8
3.3	Experimental Design	8
3.4	TCP and ICMP Traffic Performance Analysis	9
3.5	Bandwidth Estimation	9
4	Results and Analysis	11
4.1	EvDO Network Characterization	11
4.1.1	TCP Throughput	11
4.1.2	ICMP Round Trip Times	14
4.2	Bandwidth Estimation and Summary	17
4.2.1	Bandwidth Estimation	18

5	Conclusions	27
5.1	Future Work	28
6	Appendix A - Setting up the Aircard 580 under a Linux Environment	30

List of Figures

1	Experimental Network	6
2	CDF of Weekday vs. Weekend Average Throughput	12
3	Weekday vs. Weekend Average Throughput	13
4	Round Trip Times for Wednesday, November 30, 2005	14
5	Round Trip Times for Saturday, December 3, 2005	15
6	CCDF for Round Trip Times	16
7	CDF of Bandwidth Estimation for 1200 Byte Trains for 5:30 PM	17
8	Weekday 800-byte Packets	18
9	Weekend 800-byte Packets	20
10	Weekday 1200-byte Packets	21
11	Weekend 1200-byte Packets	22
12	Weekday 1472-byte Packets	23
13	Weekend 1472-byte Packets	24
14	Summary for Weekday vs. Weekend	25
15	Bandwidth Estimation for November 30, 2005	25

1 Introduction

Cellular networks are widely used today for voice communications, and the number of cellular network users is continually growing. Apart from voice, however, cellular networks historically did not have the capacity for more bandwidth intensive services, such as multimedia streaming. With the advent of third generation (3G) wireless technologies CDMA (Code Division Multiple Access) cellular network providers such as Sprint PCS Wireless and Verizon are implementing newer protocols for faster data transfer on their networks.

One such protocol is EvDO (Evolution Data Optimized). Part of the CDMA2000 standards, the EvDO protocol allows for maximum data rates higher than previous protocols such as 1xRTT (Radio Transmission Technology) or GPRS (General Packet Radio Service), the latter of which is used in GSM (Global System for Communications) cellular networks. With purported bandwidths ranging from 500 kbps up to 2.4 Mbps, EvDO networks would seem suitable for high-bandwidth applications. However, since EvDO is a relatively new cellular protocol, published research evaluating its performance is limited.

This project, therefore, was created with two major goals in mind, which combine two areas of interest among network researchers, evaluating cellular network performance and applying bandwidth estimation methodologies to EvDO networks. One goal was to determine the suitability of cellular networks, mainly EvDO networks, for data applications such as file transfers. This goal also involved evaluating whether such networks are suitable for streaming media, an application largely dependent on factors such as latency, bandwidth, and packet loss. The other objective of this project was to apply bandwidth estimation techniques on cellular networks to estimate channel capacity of the forward link.

The first goal, then, was carried out by characterizing the EvDO network based on several protocols. Network evaluation was done through experiments involving TCP and ICMP traffic. UDP-based tests were part of the original experimental design, but the lack of a suitable UDP testing utility and the lack of time to port a pre-existing tool to the platform utilized resulted in

UDP tests being removed from the project. Thus TCP throughput is analyzed in this project, whereas a more favorable situation would involve analyzing TCP and UDP throughput.

The latter goal, performing bandwidth estimation of the EvDO network, was done with a custom application that generated packet trains sent through UDP. Bandwidth estimation is important in applications such as streaming video, where apriori bandwidth estimation is desirable to possibly improve the quality of the media stream.

This report is divided into four main parts. Chapter two details background information and related work. Chapter three describes the methodology and the tools used in experimentation. Chapter four presents the data from the experimental runs and their analysis. Chapter five details conclusions extrapolated from the data and recommendations for future work.

2 Background

This chapter provides an overview of Code Division Multiple Access (CDMA), EvDO networks, a discussion of bandwidth estimation techniques and related work.

2.1 CDMA Overview

The term CDMA, in the scope of this project, describes a digital cellular network interface in which a channel is multiplexed by assigning a unique code to each voice conversation or data transfer. This scheme allows multiple users to utilize the same radio frequency[YEE91].

Originally, CDMA for cellular networks was developed by Qualcomm, and was known as IS-95 or 2G (second generation cellular)[THE05]. The IS-95 standard was superseded by the IS-2000 standard, which led to the development of the 1xRTT (radio transmission technology) protocol, the standard used by CDMA providers today. The 1xRTT protocol specifies maximum theoretical channel capacity of 153 kbps on both the forward and reverse link, making it more suitable for data transfers than IS-95 based protocols, which have a maximum channel capacity of 64 kbps on the forward and reverse links[THE05].

2.1.1 EvDO Networks

The newest CDMA data protocol, EvDO, is commonly referred to as DO (Data Only). It allows for maximum forward link capacity ranging from 500 kbps up to 2.4 Mbps, and a maximum reverse link capacity of 153 kbps[THE05].

EvDO networks are optimized for data transfer on the forward link through the use of packet-based TDM (time division multiplexing) and by scheduling packets based on a users channel quality, rather than utilizing traditional round-robin scheduling. In contrast, CDMA is used on the reverse link, but with adaptive rate control, which allows an increase in reverse link throughput that can be higher than 1xRTT networks[AIR04].

A revision to the EvDO standard, EvDO Revision A, currently available in Korea and Japan, is expected to be deployed by CDMA carriers in the United States in 2006. This new standard is designed to decrease network latency and increase bandwidth beyond the current EvDO maximum, with theoretical maximum forward link and reverse link capacities of 3.1 Mbps and 1.8 Mbps, respectively[THE05]. One of the intended purposes of the next EvDO revision is to support low latency applications such as video telephony and VoIP (Voice over IP), a technology which allows voice communications over the internet[AIR04].

2.2 Related Work

There have been several studies aimed at bandwidth estimation of networks in general as well as analyzing the performance of cellular networks. Such studies and their relation to our work are detailed in this section.

2.2.1 Bandwidth Estimation Techniques

In [DOV04], several methodologies on bandwidth estimation are presented. Since the goal of this project was to study the capacity of EvDO channel, packet dispersion techniques were the most suitable bandwidth estimation technique.

For bottleneck bandwidth estimation, fixed-size packets, in what are known as packet trains, are sent in a burst from a high speed network to a lower speed link whose bottleneck capacity is of interest. The number of packets in each burst, or the length of the packet train, can affect the resulting bandwidth estimations. As the packets arrive at the destination the spacing in time between the arrival of the first packet and the second packet in each pair is called packet pair dispersion. For a packet train, packet train dispersion is defined as the difference in time between the arrival of the first packet and the arrival of the final packet, provided the packets of the train arrive in the same order that they were sent.

From the dispersion time between two packets, channel bandwidth can be estimated. In the

absence of cross-traffic on a channel, packet pair dispersion will estimate the end-to-end capacity of a channel, regardless of the packet size. In a non-ideal environment where cross-traffic is unavoidable, the packet size used can have a significant effect on the resulting bandwidth estimation.

For the dispersion between packets, then, the channel capacity of the narrow link can be determined by $C = \frac{L}{\delta_T}$, where C is the capacity of the narrow link, L is packet size, and δ_T is the dispersion time between packets.

Similarly, for a packet train, a singular estimate of the narrow link capacity can be calculated through the formula $b(N) = \frac{(N-1)L}{\Delta(N)}$, where $b(N)$ is the bandwidth estimate, N is the length of the packet train, L is the size of the packets, and $\Delta(N)$ is the dispersion between the first packet and the last packet of the train. Alternatively, packet train dispersion can be calculated from the formula $b(N) = \frac{\sum_{k=1}^{N-1} \delta^k}{N-1}$, where $b(N)$ and N are the same as in the first packet train dispersion formula, and δ^k is the k th packet dispersion. This formula indicates that packet train dispersion is essentially the average of $N-1$ packet pair dispersions.

As train length increases, its packets are more prone to cross-traffic and interference, which can delay packet arrival and increase dispersion times. The result is an underestimate of the channels capacity that the train is transmitted through. Shorter trains, on the other hand, can produce inaccurate estimates in the presence of outlier packets[DOV04].

As mentioned, packet size can affect the resulting bandwidth estimation. Larger packets result in a wider dispersion that is more sensitive to queuing delay in intermediate hops and produce a more accurate estimation of end-to-end capacity. However, if packets are too large, such as packets larger than the MTU (maximum transmission unit) of a network, additional latency can be introduced as packets are fragmented and reassembled from server to client. Small packets are not ideal either, since they result in proportionally shorter dispersion times which can lead to overestimation of end-to-end capacity[DOV04].

3 Methodology

This project concentrated on characterizing the EvDO channel and performing bandwidth estimation on the network. Figure 1 shows a high-level overview of the network setup used in the experimentation.

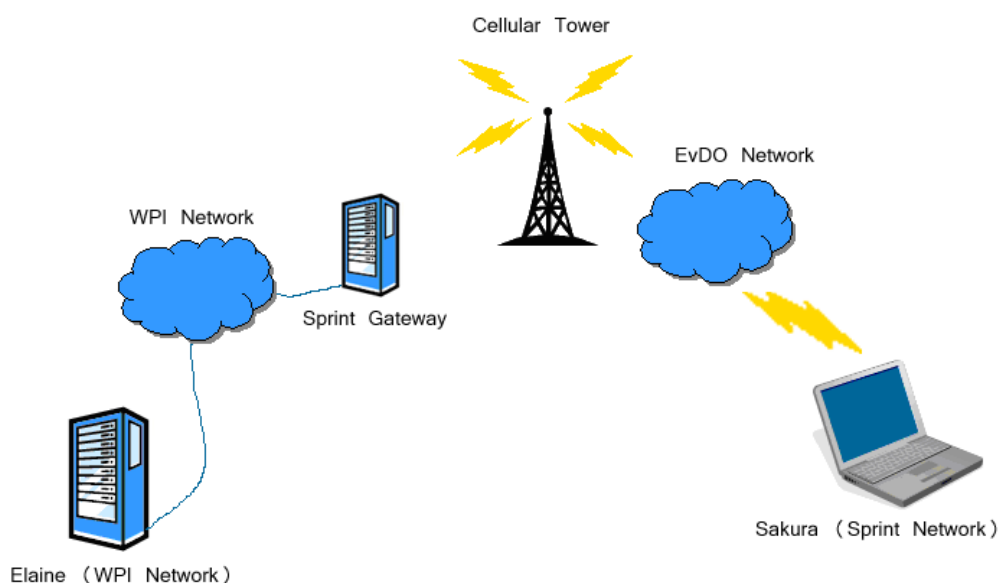


Figure 1: Experimental Network

3.1 Equipment Testbed

The experimental traffic for this study was sent between two computers. One computer, Elaine in Figure 1, is the server for bandwidth estimation experiments. It is connected to a high speed 100 Mbps uplink on the WPI (Worcester Polytechnic Institute) LAN. It runs a Linux-based operating system with the 2.6.11 kernel and a minimal amount of services necessary for basic network operation.

The destination computer, Sakura in Figure 1, is a notebook with a Sierra Wireless Aircard 580 PCMCIA EvDO network card connected to Sprint PCS's EvDO network in Worcester, MA. The notebook is equipped with a Pentium M 1.8 Ghz CPU, 1 GB RAM, and operates on the Linux

2.6.11 kernel. All services requiring network connection were shut off during the experimentation.

3.2 Tools

This section details the various tools utilized in the experiments. These tools range from open-source networking tools to a customized bandwidth estimation program, *npath Experimental*.

3.2.1 *TCPDump*, *TCPTrace*, and GNU *wget*

TCPDump is a program that captures all traffic information on a specific network interface device. When TCP file transfers are initiated in our experiments, *TCPDump* captures all activity of the EvDO network card into a binary file on Sakura. This file is later analyzed to extract required information about a specific transfer.

TCPTrace is a tool from Ohio University which can parse and process *TCPDump* files. Using this tool required information about duration of a transfer and the number of bytes transferred was extracted. This data was used to calculate average TCP throughput as is discussed in the "Results and Analysis" section.

GNU *wget* is a tool that is capable of retrieving content from servers using various TCP/IP-based protocols, such as HTTP (hypertext transport protocol). It was used to transfer a 20 MB file from Elaine to Sakura.

3.2.2 ICMP *ping*

The standard Linux/Unix *ping* utility was used to estimate round trip times and packet loss rates between hosts by sending ICMP "echo request" packets every second to the target host on the Sprint network and listening for replies. If a response is not received from the target within one second, a packet loss is recorded. The standard Linux *ping* command is used to generate 100 pings during each test run.

3.2.3 *GNUplot*

GNUPlot is a graphing tool capable of generating two- and three-dimensional plots of functions and data. Given a set of data and additional parameters, such as scale and axes labels, it can generate a variety of graphs of various formats. *GNUPlot* was used to generate all the graphs seen in the "Results and Analysis" chapter.

3.2.4 *npath Experimental*

Loosely based on *Pathrate*, which is a bandwidth estimation tool as discussed in [DOV04], *npath Experimental* is a bandwidth estimation tool created specifically for this project.

npath Experimental exists as a server and a client. The server program was run on Elaine and waited for a connection from the client program, which was run on Sakura. The client would initiate a TCP connection with the server, signaling the server to begin the bandwidth estimation experiment.

The server then generates packet pair trains and sends them out in a burst to the client through a UDP transmission. Each packet contains two pieces of information, the train number and the packet number. As the client receives each packet, it checks the train number and packet number. If either are not what is expected, the client discards the train and sends a NACK (Negative Acknowledgment) packet to the server to begin retransmission of the train. If the train is received without error, the experiment continues. The server pauses for one second between each packet train, and ten seconds between every ten trains. After thirty packet pair trains, the server changes packet sizes and begins anew. Packet dispersion information is recorded on the client side.

3.3 Experimental Design

For this project a series of BASH shell and Perl scripts were designed. These scripts utilized open-source network tools for Linux and a custom tool *npath Experimental*. Tests were scheduled

to run every thirty minutes for seven days and outputs were logged on Sakura for later analyses. Each test was approximately twelve minutes long and consisted of 20 MB TCP transfer, 100 ICMP pings within the Sprint PCS network, and four bandwidth estimation measurements (each with a different packet size) done in series. Signal strength of the EvDO card was measured prior to each run by sending a special serial command to the card. This was done to insure that signal strength would not have an adverse effect on network performance. For each one of the runs the signal strength ranged from -90 to -83 dBm which is equivalent to excellent signal strength relative to cell phone signal strength metrics.

3.4 TCP and ICMP Traffic Performance Analysis

To analyze the performance of TCP traffic, a 20 MB file was transferred using *wget* via a TCP connection from Elaine to Sakura. *TCPdump* was used to generate and log an in-depth trace of all activity on the EvDO network interface device during the file transfer. After the TCP transfer was completed, 100 ICMP pings were sent from the notebook to a node within the Sprints network to gauge ICMP round trip time performance. Since WPI's network infrastructure drops all incoming ICMP traffic, it was necessary to ping within Sprint PCS's cloud. Pinging within the Sprint network also eliminated possible overhead from sending pings from one network to another. Round trip time and packet loss information was time stamped and logged for each ping on Sakura for detailed analysis.

3.5 Bandwidth Estimation

To estimate the available bandwidth of the EvDO network, *npath experimental* was developed to send packet trains from a server on a high speed network to a client computer on the EvDO network. The tool sent out 30 trains with 25 packets each of size 1472, 1200, 1000, and 800 bytes. The 1472 byte packet was chosen because with IP layer overhead, it is the maximum allowable size of an Ethernet packet (1500 bytes). Since the actual MTU (Maximum Transmission Unit) of

the EvDO network was unknown to us, packets of 1200, 1000, and 800 bytes we also used.

The length of the trains was experimentally determined using an estimation tool that would send bursts of packet trains with 1472 byte payload and incremented the number of packets in a train until packet loss was seen, meaning packets would not arrive at the client after being sent by the server. Using this technique the longest train that could be sent without error was determined to be 25 packets, and this length was used for each packet size for consistency.

4 Results and Analysis

This chapter presents results and analysis for the experiments for this study. The data are from a week of experimental runs, collected from November 28, 2005 to December 4, 2005. From the data, performance was seen to be consistent from weekday to weekend, and thus this chapter will present a small subset of the data. The follow analysis concentrates on characterizing the EvDO connection as well as bandwidth estimation of the EvDO network on a typical weekday, Wednesday November 30, 2005, and a typical weekend, Saturday December 3, 2005.

4.1 EvDO Network Characterization

4.1.1 TCP Throughput

While it would have been favorable to conduct experiments with UDP traffic to measure EvDO network performance and compare the results to the bandwidth estimations obtained from *npath Experimental*, TCP traffic was used instead. This decision was made due to the lack of a UDP network testing tool that was sufficient for evaluating channel capacity and throughput. The best candidate, a tool called *UDPload*[GLL05], was not available for the Linux environment and to port it from Windows to Linux could not be done due to a lack of time and differences in network programming between Windows and Linux environments.

Figure 2 shows a CDF for the average TCP throughput for both a weekday and a weekend day, obtained from *TCPdump* and *TCPtrace* packet traces of the 20 MB transfer using *wget*. The x-axis represents throughput in kilobits per second, and the y-axis represents cumulative distribution. The red line represents the weekday, and the green line represents the weekend. The average throughput for this calculation was derived by dividing total number of bytes transferred during a given run (including retransmissions) by the dispersion, which was in microseconds, between first and last packet seen.

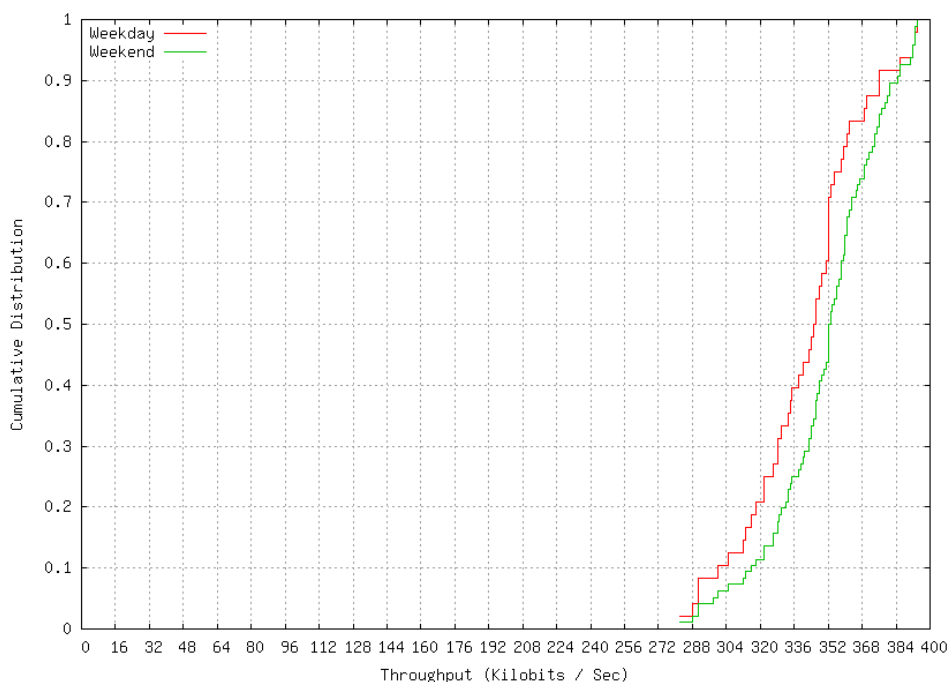


Figure 2: CDF of Weekday vs. Weekend Average Throughput

The shape and the relatively small spread of values of the CDFs indicate that TCP speeds remained reasonably consistent. Ninety percent of throughput values for a weekday were below 375 kbps and for a weekend were below 384 kbps. Figure 2 also shows that most TCP throughput, namely those outside of the top and bottom five percent, are slightly higher for a weekend than for a weekday. This suggests that the network provides greater available capacity for users on a weekend than during a weekday.

Figure 3 provides the average throughput of the *Wget* TCP download for every half hour run throughout a weekday compared to the weekend day. The x-axis shows the time of day in military time, and the y-axis shows average throughput in kbps. The red line represents the weekday, whereas the green line represents the weekend.

Notice that there is no real consistency in Figure 3 to the average throughput at any time of day. Additionally, no time of day correlation can be seen regarding TCP throughput performance. Even during the early hours of the weekday an average throughput of 376 kbps was reached at 5:30, but

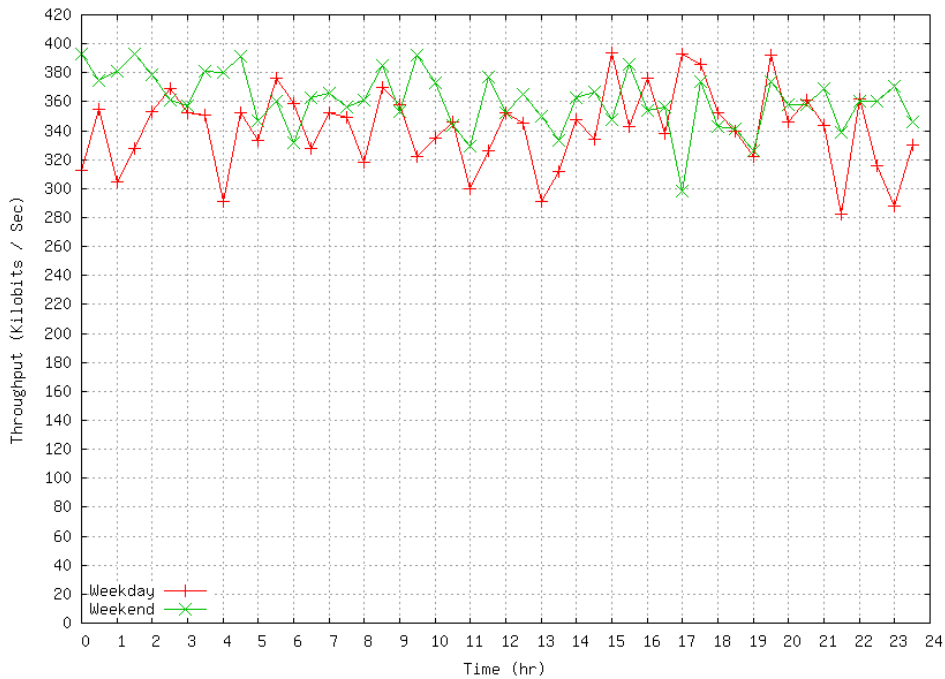


Figure 3: Weekday vs. Weekend Average Throughput

this followed by a drop to 360 kbps at 6:00 down to 328 kbps average throughput at 6:30.

An equivalent weekday result can be found after the afternoon peak at 15:00. The average throughput drops to 344 kbps just a half hour later, then increases again to 376 kbps by 16:00, only to drop down to 340 kbps a half hour afterwards.

Similar to the weekday line, no real consistency can be seen in the weekend results, but it can be observed that throughout the day average TCP throughputs are generally higher than the weekday graph. Peaks of around 392 kbps can be seen even in the very early hours of 12:00, 1:30, and at 4:30, that are higher than their counterparts in the weekday graph. This is a stark contrast to the earliest hours of our weekday observations, where 1:00 only had an average throughput of around 304 kbps, for example. Yet, several times during the weekend, average throughputs were below their weekday counterpart, such as a significant drop from 16:30 - 17:30.

4.1.2 ICMP Round Trip Times

To measure round trip times, 100 ICMP pings were sent every half hour from the laptop on the EvDO network to a gateway within the Sprint network. Similar to the TCP analysis, weekday versus a weekend day performance is considered.

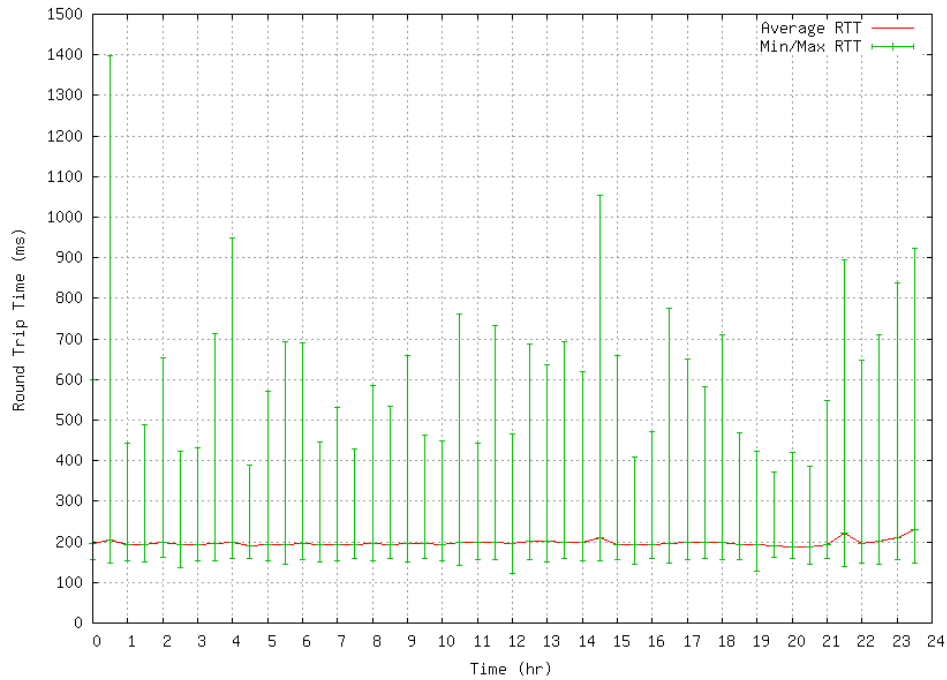


Figure 4: Round Trip Times for Wednesday, November 30, 2005

Figure 4 shows three pieces of RTT information. The x-axis represents time of day, and the y-axis represents round trip time in milliseconds. The vertical green bars show the minimum and maximum round trip times among the 100 ICMP packets sent during that experimental run. The red line going horizontal provides the average round trip time for all 100 packets during each run.

Similar to the previous graphs of average TCP throughput, Figure 4 indicates there exists no time of day correlation for ICMP round trip times. The maximum round trip times vary greatly throughout the day, with a high of 1400 milliseconds at 12:30 AM and a low around 370 milliseconds at 7:30 PM. However, the minimum round trip times exhibit little variability throughout day, hovering around the 160 millisecond range.

The most interesting observation from this graph is, despite the variations in maximum RTTs, the average RTT remains very consistent throughout the day, at around 200 milliseconds. This suggests that the maximum RTTs are outliers and the majority of the packets have RTTs near the average RTT for each run throughout the day.

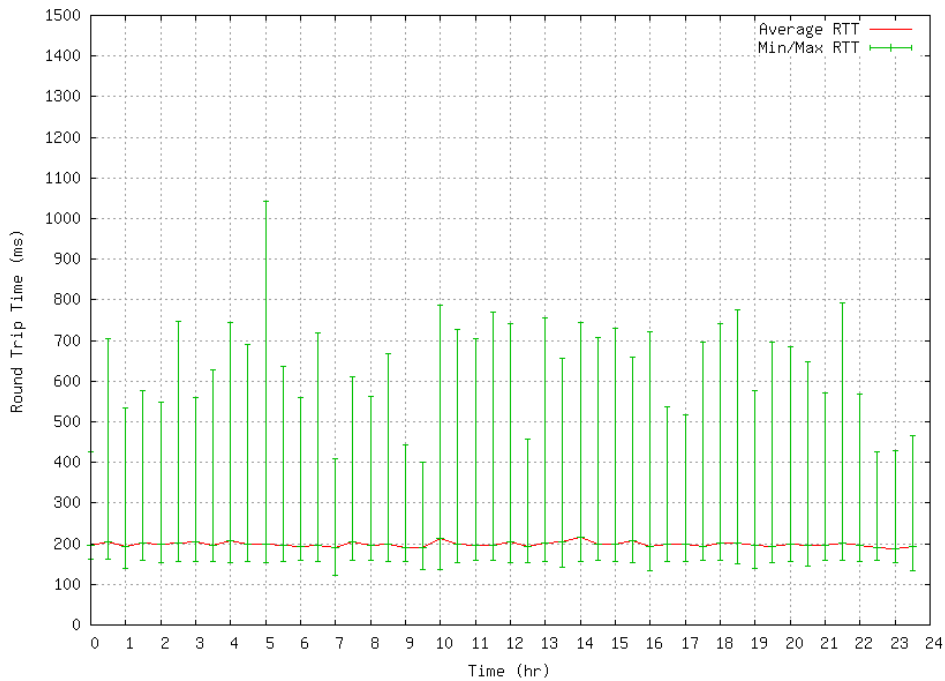


Figure 5: Round Trip Times for Saturday, December 3, 2005

Figure 5 is similar to the previous round trip time graph, except that it provides RTT data for a weekend day. The axes, vertical bars, and line represent the same metric as in Figure 4. Again, there is no real correlation between time of day and RTT performance. Similar to the weekday graph, RTT minimums hover around 180 milliseconds, and the average RTT of each experimental run is around 200 milliseconds.

Comparing Figures 4 and 5, it is clear that there is no difference in RTT performance between a weekday and a weekend.

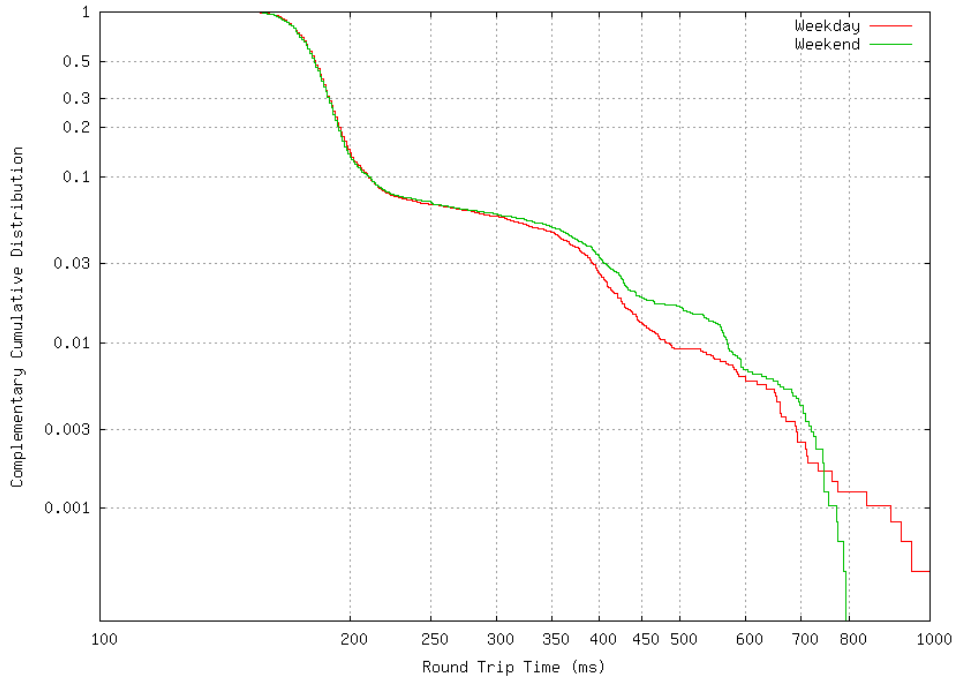


Figure 6: CCDF for Round Trip Times

Figure 6 depicts a complementary CDF (CCDF) of all the RTT measurements gathered on the weekday and the weekend. The red line represents the weekday RTT data, and the green line represents the weekend RTT data. The x-axis represents round trip time in milliseconds, and y-axis represents complimentary cumulative distribution in logarithmic scale.

Note that the median of the weekday RTT measurements and that of the weekend RTT measurements overlap at around 180 milliseconds, and only about ten percent of the RTTs were above 220 milliseconds throughout the entirety of either 24-hour cycle.

Judging from the three RTT figures shown, it can be seen that RTT performance does not vary greatly throughout the day or whether it is a weekday or weekend.

The latency of the studied EvDO network is still too high for real-time video applications, such as video conferencing. Significant RTT spikes shown will result in jitter and have an adverse effect on such applications. Despite that fact, the results of this experiment could allow for software optimizations to be put in place to allow quality streaming of pre-recorded content over the EvDO

network.

4.2 Bandwidth Estimation and Summary

This section presents a sample of the bandwidth estimation results.

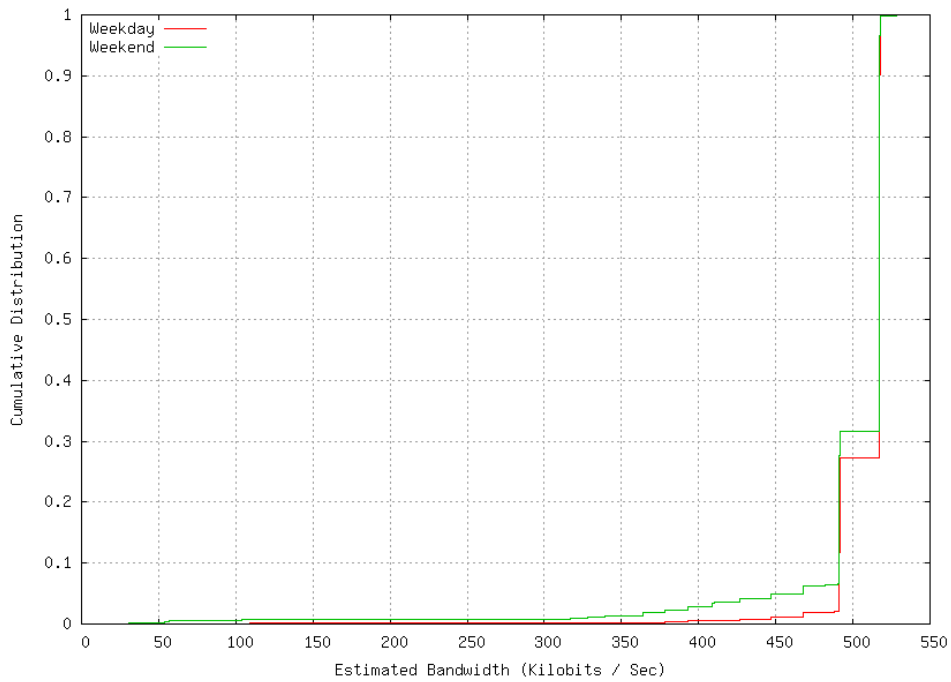


Figure 7: CDF of Bandwidth Estimation for 1200 Byte Trains for 5:30 PM

Figure 7 presents a CDF of estimated bandwidth at 5:30 PM for the weekday and weekend previously discussed. The x-axis shows estimated bandwidth in kilobits per second, y-axis shows cumulative distribution, red line represents the weekday, and the green line represents the weekend. The 1200-byte packet results are presented because the largest byte size chosen, 1472 bytes, gave slightly different results compared to the other byte sizes.

Seventy percent of the bandwidth estimation, from 0.3 to 1.0 cumulative distribution, is the same, at 520 kbps. This value fits in with the lower end maximum capacity for EvDO networks, specified to be 500 kbps.

One observation from this result is that bandwidth estimation can be performed over the EvDO

network with smaller number of packets in a train, since there is a high likelihood that the estimated bandwidth will be near the correct capacity of the channel. Thus applications that need a prior bandwidth estimation do not have to do it using as many trains or packets as used in this investigation. at

4.2.1 Bandwidth Estimation

This section presents data showing bandwidth estimation and average TCP throughput for different packet sizes. For each graph, the x-axis represents time of day, the y-axis represents bandwidth in kilobits per second, the red line shows average TCP throughput, the blue line represents fifth percentile bandwidth estimation, and the green line represents median bandwidth estimation.

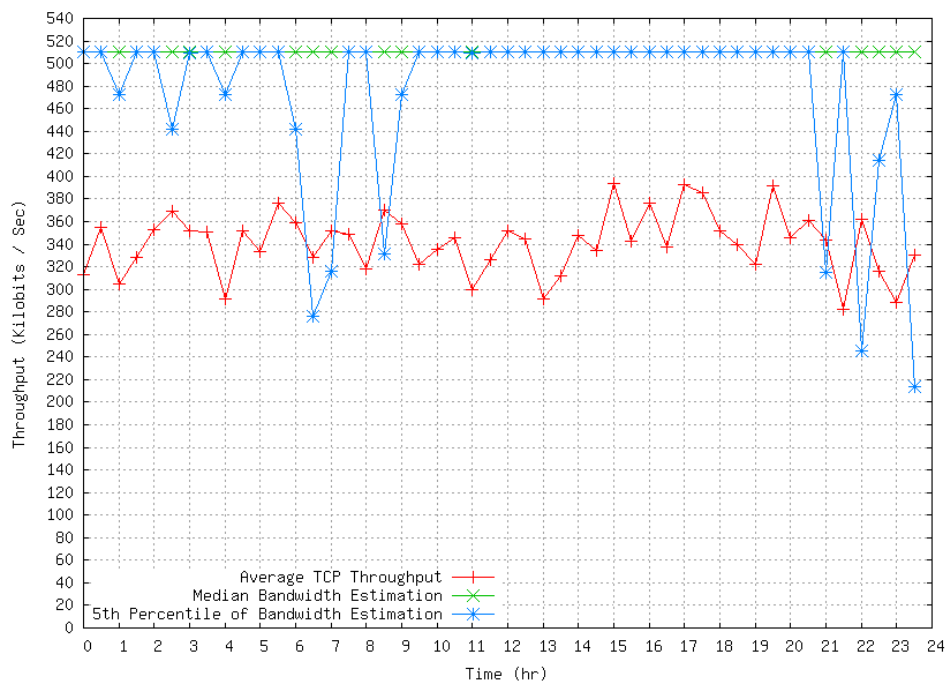


Figure 8: Weekday 800-byte Packets

Figure 8 displays the results of 800 byte packets for the weekday. Note that the median bandwidth estimation is a consistent 510 kbps. The worst case fifth percentile bandwidth estimation significantly deviates from the median, such as at 6:30 AM, where worst case performance is es-

estimated to be approximately 280 kbps. During the peak hours of the day (late morning to evening hours), the fifth percentile bandwidth estimation is very consistent at 510 kbps, the same as median bandwidth estimation. The median for average TCP throughput is around 330 kbps. As expected, due to the overhead involved in TCP protocol this value is significantly lower than the bandwidth estimation value.

The dips seen in the fifth percentile estimation from 5 AM to 10 AM probably result from early morning users utilizing the network before peak hours take effect.

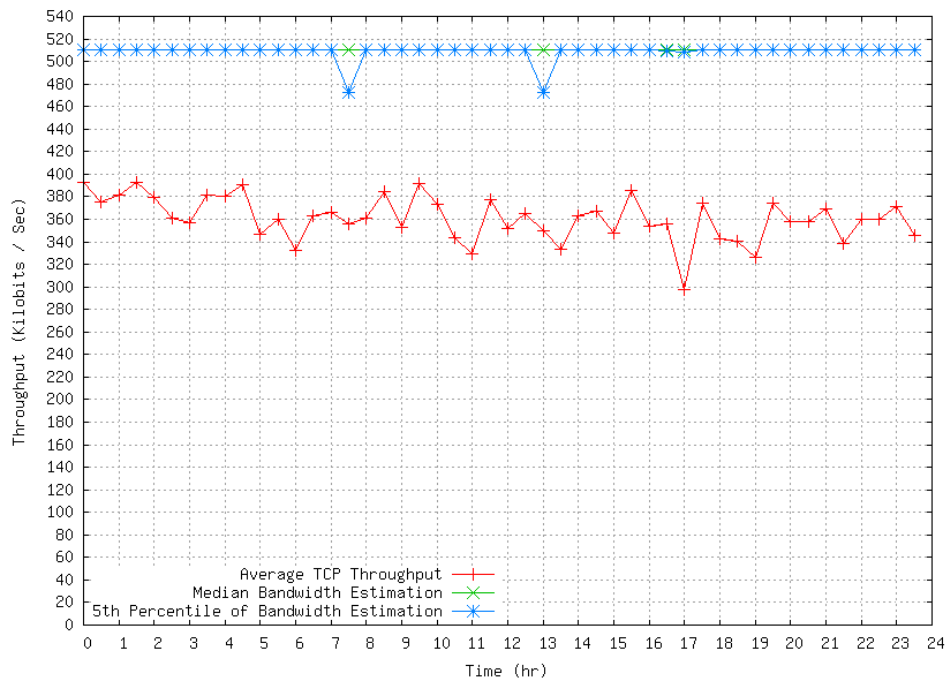


Figure 9: Weekend 800-byte Packets

Figure 9 presents a summary of results of 800 byte packets for the weekend. The median bandwidth estimation is again 510 kbps, same as with the weekday. The fifth percentile bandwidth estimation, however, is more consistent in Figure 9 than in the weekday results. This is most likely due to weekend minutes being free for network subscribers, and as such calls are distributed more evenly throughout the day unlike on a weekday. The average TCP throughput is also more consistent, with a median around 360 kbps.

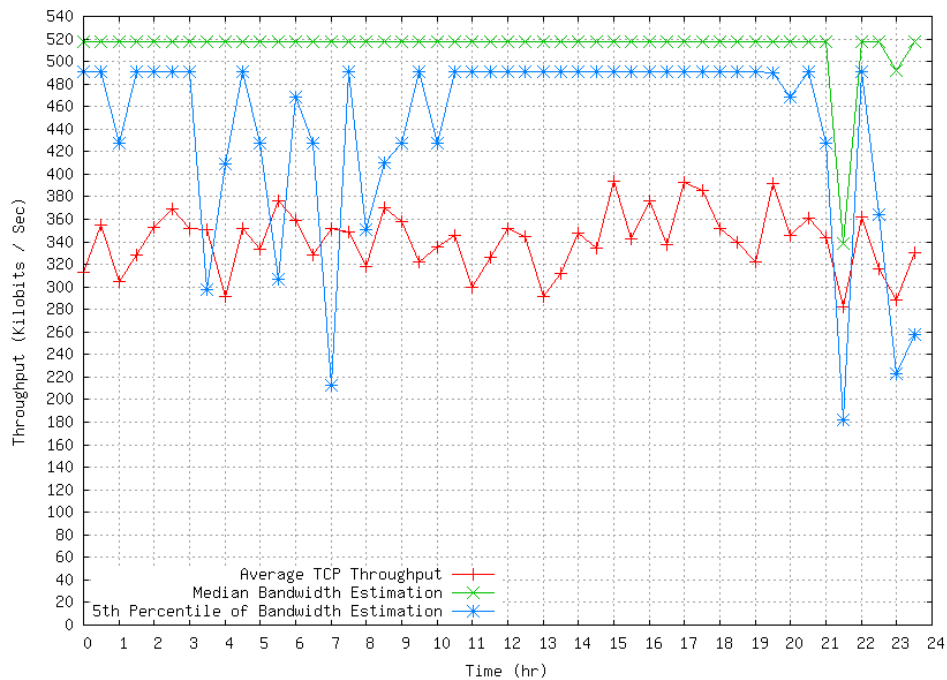


Figure 10: Weekday 1200-byte Packets

Figure 10 depicts results for 1200 byte packets for the weekday. Compared to the graph for 800 byte packets on the weekday, there is now a gap of 30 kbps between the median and fifth percentile estimations. Median bandwidth is also estimated at 520 kbps throughout almost the entire day except in the late evening.

The fifth percentile estimation is consistently lower than the 800 byte estimates previously shown, and the highest estimate is 490 kbps, as opposed to the 510 kbps estimations achieved with 800 byte packets. This is most likely due to the increased packet size, as larger packet sizes can get queued up for longer periods of time in the intermediate hops as the packets travel from server to client.

This graph follows a similar trend to the 800 byte graphs for fifth percentile bandwidth estimation. The middle of the day from 11:00 AM to around 7:30 PM has a very consistent worst case performance of 490 kbps throughout. Just as in the weekday 800 byte graph, there are many dips in the early morning and late at night for the fifth percentile graph.

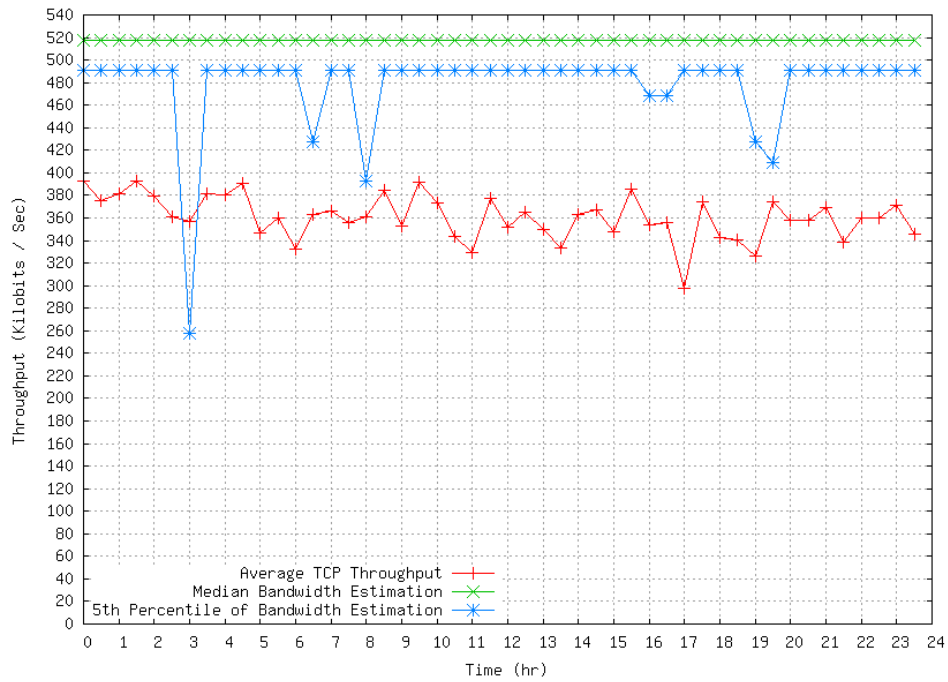


Figure 11: Weekend 1200-byte Packets

Figure 11 shows results 1200 byte packet sizes for the weekend. As in the previous graphs, the median bandwidth at stays at 520 kbps throughout the day. In the worst case fifth percentile graph, more consistency can be seen as compared to the weekday summaries for the same packet size. However, there are more dips in the fifth percentile graphs for 1200-byte packets than the equivalent 800-byte packets in Figure 9.

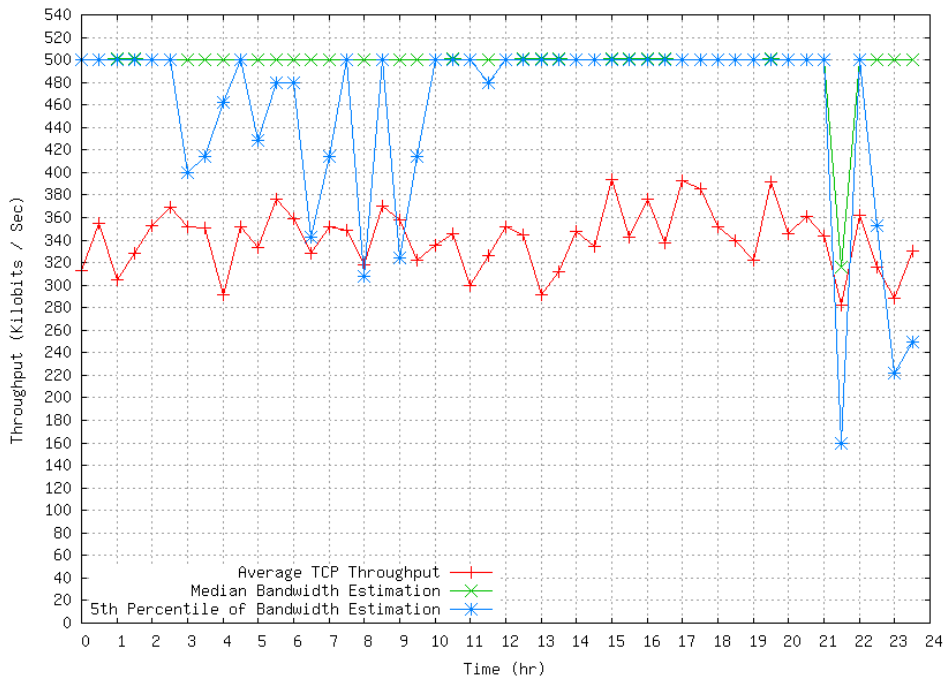


Figure 12: Weekday 1472-byte Packets

Figure 12 depicts results for 1472 byte packets on the weekday. This packet size produces differences from either the 800 or 1200 byte graphs. Median bandwidth through the day is now estimated to be 500 kbps, and in the worst case performance 500 kbps. The median bandwidth estimation is, of course, estimated to be lower than when 800 or 1200 byte packets are used. This result is most likely due to 1472 byte packets being larger than the MTU of the EvDO network, causing additional latency when packets are fragmented and reassembled, which will affect the bandwidth estimation, dependent on dispersion time between packets.

The graph shows the same trends as in the other weekday summary graphs, with consistent worst case performance of 500 kbps throughout peak hours of the day, and high variability in the fifth percentile bandwidth estimation during the morning and late evening.

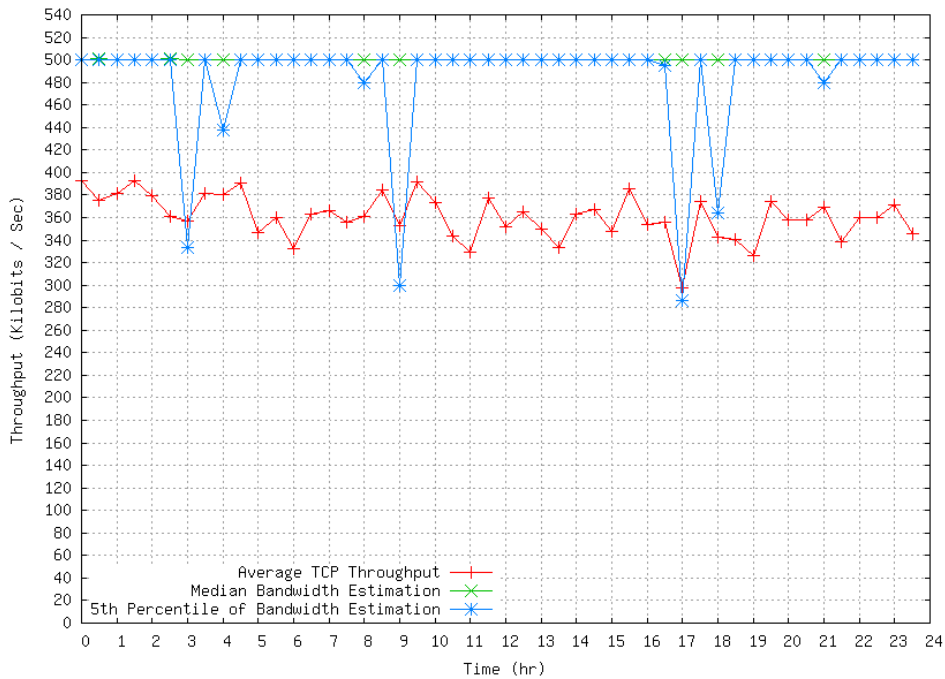


Figure 13: Weekend 1472-byte Packets

Finally, Figure 13 represents results for the weekend when 1472 byte packets are used. The same results as in the weekday graph can be seen. Median bandwidth is estimated at 500 kbps, consistent throughout the day, and the fifth percentile performance hovers around 500 kbps, except for a few dips throughout the day.

Note that for each of the previous graphs shown, estimated bandwidth is higher than the average throughput. The reasons for this behavior are that TCP traffic incurs overhead such as slow start and the need for acknowledgement (ACK) packets. Our bandwidth estimation was done with UDP traffic, which does not have the same problems.

	Weekday	Weekend
Throughput (KB/s)		
Average:	341	360
Standard Deviation:	27	19
Round Trip Time (ms)		
Average:	197	198
Standard Deviation:	68	72
Loss (%)		
Average:	0	0
Standard Deviation:	0	0

Figure 14: Summary for Weekday vs. Weekend

Figure 14 summarizes average throughput, round trip time, and packet loss for the weekday and weekend from the previous graphs. The average throughput can easily be seen to be higher throughout the day during the weekend than during the weekday. This is the same result shown by the average throughput CDF show in Figure 2.

Average round trip times were nearly the same between the weekday and weekend, and the standard deviation from the average is nearly the same for both. Note that throughout all the ping packets sent during a day, there was no packet loss on either the weekday or weekend. This is an expected result, since CDMA utilizes FEC (forward error correction) and ARQ (automatic repeat request) to correct errors and retransmit frames at the wireless data link layer.

Byte Size (B)	Median Bandwidth (kb/s)	Average Dispersion Time (ms)	5 th Percentile (kb/s)
1472	500	24.0	500
1200	517	19.0	491
1000	514	16.0	484
800	510	13.0	510

Figure 15: Bandwidth Estimation for November 30, 2005

Figure 15 summarizes bandwidth estimation results for the weekday discussed throughout this report. The table indicates that average dispersion time between packets does increase as the packet sizes get larger, but the median bandwidth estimation does not show the same trend. Whereas dispersion time increases as the size of the packets increased, median bandwidth estimation increases

from 800 to 1000 to 1200 byte packets, but then drops to 500 kbps at 1472 packets.

Again, the cause of this behavior would be packet fragmentation, for packets of size 1472, as discussed earlier. This would also explain why the median bandwidth estimation does not follow the trend seen when progressing from 800 to 1000 to 1200 byte packets, which is a increase in median bandwidth along with increase in dispersion times.

5 Conclusions

This project originally sought to study EvDO network performance based on various types of Internet traffic, including TCP, UDP, and ICMP traffic, perform bandwidth estimation tests using packet trains on the network, and provide a foundation for additional research into EvDO networks. However, due to time and tools constraints, characterization of the EvDO network could not be fully done, since UDP traffic experiments were not conducted.

This project succeeded in building a working testbed on the WPI's network and conducting a series of experiments on Sprint PCS EvDO network in Worcester, MA. These experiments were able to collect measurements of TCP throughput and ICMP RTT times. A program developed by Constantine Dovrolis was successfully adopted and modified to perform bandwidth estimation. Bandwidth estimation analysis in terms of size of packet trains and size of packets was performed. This project was able to show the difference in performance during peak weekday hours versus off peak time and compare performance between a typical weekday and a typical weekend.

Study of the EvDO link was performed based on TCP and ICMP traffic, and bandwidth estimation tests. From the data gathered, it can be concluded that RTT performance is not dependent on time of day or day of the week. TCP performance shows a similar result, but is slightly better on the average during a weekend than during a weekday.

In terms of bandwidth estimation, the results show that estimations are very likely to be close to the theoretical channel capacity of a low-end EvDO network. This conclusion suggests that accurate bandwidth estimation could be performed without having to generate large numbers of packet trains or long packet trains as were used for this project's experiments.

The project, however, was not carried out perfectly. Several obstacles arose, such as the inability to examine UDP performance across the EvDO network, as mentioned earlier. The tool that could be used for such a purpose was not completely portable to the environment the experiments were conducted in, and time constraints did not allow for a new tool to be created. The bandwidth estimation tool created for the project, *npath Experimental*, was not as robust as the original

Dovrolis source tools. Parameters such as packet sizes and train length hard-wired into *npath Experimental*. The code, however, should be easily modifiable for further bandwidth estimation experiments.

5.1 Future Work

An original goal of this project was to perform UDP traffic tests on the EvDO network. A future project could study TCP traffic against UDP traffic and bandwidth estimation results. UDP traffic tests could be performed with UDPLoad, which is the Windows-based UDP traffic tool that could not be fully ported to the Linux environment for this project. Since it is conjectured that TCP throughput is lower than estimated bandwidth due to additional overhead such as ACKs, studying UDP throughput would show whether or not this hypothesis is true.

Since the experiments of this project were carried out on the Sprint EvDO network, another future project could be carried out on competitors' EvDO networks, such as that of Verizon Wireless. A study could also examine GSM-based networks, such as T-Mobile's or Cingular Wireless's, and see if performance is as consistent compared to EvDO networks.

Additionally, further work could build upon the experiments conducted in this project. Packet trains of different lengths were not experimented with in this project, so future research could be conducted with packet trains of varying lengths.

Those who carry out such future work, however, would have to modify the custom tool used in this project, *npath Experimental*, for their specific purposes. For end-to-end capacity estimation, the tool would only have to be slightly modified for different packet sizes, number of trains, and train length. For additional bandwidth estimation methods such as measuring available channel capacity, a new tool based on *Pathload* [JAI03], also developed by Constantine Dovrolis, would have to be created.

References

- [AIR04] "All-IP 1xEV-DO Wireless Data Networks: A Technical White Paper," August 2004, http://www.airvananet.com/files/Airvana_1xEV_Tech_White_Paper.pdf.
- [DOV04] Dovrolis, C., Ramanathan, P. and Moore, D. "Packet-Dispersion Techniques and a Capacity-Estimation Methodology," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 963-977, December 2004.
- [GLL05] Gretarsson, J. Li. F, Li. M, Samant, A. Wu, H. Claypool, M., and Kinicki, R. "Performance Analysis of the Intertwined Effects between Network Layers for 802.11g Transmissions," In Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP), Montreal, Canada, October 2005. Online at: <http://www.cs.wpi.edu/~claypool/papers/co-ap/>
- [JAI03] Jain, M. and Dovrolis, C. "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 537-549, August 2003.
- [LI05] Li, Mingzhe. "Techniques for Improving Performance of Streaming Multimedia over Wireless Networks," WPI Ph.D Proposal, submitted August 2005.
- [THE05] Thelander, M.W. "Taking CDMA2000 into the Next Decade," October 2005, http://www.cdg.org/resources/white_papers/files/3G_Evol_Oct05.pdf.
- [WEI05] Wei, W., Zhang, C., Zang, H., Kurose, J., and Towsley, D. "Inference and Performance Evaluation of Cellular Data Networks through End-to-End Measurements," November 2005, http://www-net.cs.umass.edu/~weiwei/Wei06_cellular.pdf.
- [YEE91] Yee, William C. Y. "Overview of Cellular CDMA," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 291-302, May 1991.

6 Appendix A - Setting up the Aircard 580 under a Linux Environment

One of the prerequisites to using the tools and running the experiments in this project is setting up an EvDO card under Linux. For this project, the Sierra Wireless Aircard 580 was used, which requires manual installation in a Linux environment due to a lack of drivers. This guide will detail how to do so. All instructions have been tested on Debian Sarge, Ubuntu 5.0.4 and 5.10, and Fedora Core 3 and 4, but the instructions should apply to any modern Linux distribution based on the 2.6 kernel. The instructions have not been applied to a Linux system running a 2.4 series kernel.

Prerequisites: The steps in this guide have been performed on Linux kernels newer than 2.6.9, and udev versions newer than version 46. The point-to-point protocol daemon (pppd) should already be installed on the host machine. USB hotplug support should be enabled within the kernel or installed otherwise.

This guide assumes knowledge of the Linux command line on the part of the user.

1.) Insert the Aircard 580 into the PCMCIA slot.

2.) From the command line, type:

```
modprobe usbserial vendor=0x1199 product=0x0112
```

3.) From the command line, type these two lines:

```
mknod /dev/ttyUSB0 c 188 0
```

```
mknod /dev/ttyUSB0 c 188 1
```

4.) Now create a pppd profile with a text editor, for example: **vim /etc/ppp/peers/1xevdo**

5.) Place these lines into the profile:

```
-detach
```

```
ttyUSB0
```

```
115200
```

```
debug
```

```
noauth
```

```
defaultroute
```

```
usepeerdns
```

```
user <ID>
```

```
show-password
```

```
crtstcts
```

```
lock
```

```
connect '/usr/sbin/chat -v -t3 -f /etc/ppp/peers/1xevdo_chat'
```

For Sprint, <ID> can be anything, even blank. Under other EvDO carriers, <ID> might be

the card's 10 digit phone number.

6.) Create a file called 1xevdo_chat in /etc/ppp/peers.

7.) Paste these lines into 1xevdo_chat:

```
" 'AT'  
'OK' 'ATE0V1&F&D2&C1&C2S0=0'  
'OK' 'ATE0V1'  
'OK' 'ATS7=60'  
'OK' 'ATDT#777'
```

8.) To connect to the EvDO network, execute this command from a terminal:

```
pppd call 1xevdo
```