Best-Case WiBro Performance for a Single Flow

Shinae Woo[†], Keon Jang[†], Sangman Kim[†], Soohyun Cho[*], Jaehwa Lee[*], Youngseok Lee[‡], and Sue Moon[†]

[†] Computer Science Department, KAIST

[‡] School of Computer Science & Engineering, Chungnam National University

[*] KT

ABSTRACT

WiBro (Wireless Broadband Internet), the Korean version of mobile WiMAX compatible standard, provides high-speed mobile data service. Although mobile WiMAX services are being deployed, there exist few reports about WiBro performance. In this work, we measure and analyze best-case performance of WiBro. In order to measure one-way delay, we develop a GPS synchronization device to measure one-way delay.

Our measurement shows that the maximum throughput over the WiBro network is 10 Mbps in downlink and 2.5 Mbps in uplink. We estimate that both the base station and WiBro modems have large buffers up to 2 s and 500 ms and minimum one-way delay of 76 ms and 11 ms for uplink and downlink, respectively. We measure TCP throughput over WiBro by varying the send buffer and receive buffer sizes. To fully exploit the high bandwidth of Wibro, we conclude the TCP needs along with the minimum of 128 KB buffer size.

1. INTRODUCTION

WiBro (Wireless Broadband Internet), the Korean version of mobile WiMAX compatible standard, provides high-speed mobile data service. The strength of WiBro over cellular data services is its data link speed. It supports up to 37 Mbps downlink and 10 Mbps uplink in theory and 10 Mbps downlink and 2.5 Mbps uplink in current deployment. The offered rates are faster than CDMA 1x EV-DO, W-CDMA or HSUPA. WiBro supports mobility up to 120 km/h, which is slower than 300 km/h cellular technologies, but still sufficient for vehicular mobility. KT launched the world's first WiBro service in Korea in 2006 and it has since acquired more than 200,000 subscribers. Mobile WiMAX services are slowly gaining foothold in other parts of the world. Clearwire started mobile WiMAX service in Portland, Oregon, in September 2008 and UQ communications begun a trial service in Japan in February 2009.

Although mobile WiMAX services are being deployed, there exist few reports about WiBro performance due to the following difficulties in measuring it. First, the infrastructure-based WiBro is yet to be widely deployed, and only a few cities have access. Second, no information about the PHY and MAC layer is available to end users. WiBro provides various MCS (Modulation and Coding Schema) levels to maximize the physical layer throughput. The MCS level changes based on signal quality and determines the transmission rate. Without the information about the MCS level, it is hard to know the maximum available bandwidth of the moment. In another example, it is hard to differentiate the sources of latency. A base station receives bandwidth requests from WiBro modems and allocates time and frequency slots. HARQ (Hybrid Automatic Repeat reQuest) conducts automatic repeat of a frame at the physical layer or recovers bit errors in case of a failure. It reduces the link layer loss rate, but increases the latency. Without information about HARQ or scheduling at the base station, we cannot tell if the latency of a particular packet is due to automatic repeat at the physical layer or scheduling.

As described above, many factors contribute to the performance of the WiBro network and complicates the analysis. The goal of this work is to measure and analyze baseline performance of WiBro. In order to reduce the variables in our experiment, we assume no mobility and low signal variation by limiting the experiment site to one physical location. We focus on the performance of a single flow with no competing flow from the same end host.

In this work we study the performance of a single UDP flow and a single TCP flow. In our UDP experiment, we have observed 2.5 Mbps and 10 Mbps for uplink and downlink, respectively. These rates are larger than reported about any cellular networks. From our measurments, we estimate that both the base station and WiBro modems have large buffers up to 2 s and 500 ms and minimum one-way delay of 76 ms and 11 ms for uplink and downlink, respectively. In our TCP experiments, we have found that the small TCP send buffer size is the bottleneck of TCP throughput because of WiBro's high bandwidth-delay product (BDP) characteristics. In case of downlink, a single TCP flow achieves 1.5 Mbps with 17 KB TCP socket buffer, the default size in Windows XP, and 5 Mbps, just half of maximum UDP bandwidth, with 64 KB TCP socket buffer (the maximum size on Windows XP without RFC 1323 extention). To fully exploit the high bandwidth of Wibro, the TCP session needs the minimum of 128 KB or larger buffer size.

The rest of this paper is organized as follows. We provide WiBro's MAC (Medium Access Control) layer mechanisms as an overview and summarize related work in Section 2. Next, we describe the architecture of our testbed, synchronization techniques and validate our testbed by examining bandwidth and delay of each hop in Section 3. In Section 4, we analyze TCP performance. We provide the UDP performance of WiBro to compare it against TCP performance. Then we demonstrate how the send buffer size and receive window size of a TCP session affect TCP throughput. Finally, we conclude in Section 5.

MICNET'09, September 21, 2009, Beijing, China.

2. BACKGROUND AND RELATED WORK

Here we give a brief overview of WiBro technology. WiBro uses Orthogonal Frequency Division Multiple Access (OFDMA) that allows simultaneous transmission of multiple users by different carriers. The core of WiBro is an IP-based packet-switching network, while WiBro employs a connection-oriented MAC layer service at the base station. A WiBro modem requests bandwidth by sending a stand-alone request message or piggybacking it on other uplink data messages [9]. Uplink has a bigger latency than downlink because uplink requires an additional bandwidth request step. Theoretically, uplink provides up to 10 Mbps bandwidth and downlink up to 37.5 Mbps with mobility up to 120 km/h [3, 9]. Depending on the quality of channels, WiBro adapts the MCS (Modulation and Coding Scheme) level to maximize the data transmission rate. WiBro has 8 different MCS levels for downlink and 4 levels for uplink. The WiBro standard includes five kinds of QoS including UGS (Unsolicited Grand Service), rtPS (Real-time Polling Service), nrtPS (Non-real-time Polling Service), BE (Best Effort), ertPS (Extended rtPS) as in WiMAX specifications, but only a BE service is currently available.

Han *et al.* have evaluated Voice over IP (VoIP) performance over WiBro [6]. Their results show that the WiBro network has small jitter and low loss: WiBro offers as good as or better than toll quality. Their results are overly optimistic, for there apparently was no other user contending for transmission. Kim *et al.* have reported on achievable throughput over WiBro [8]. They have demonstrated that the small default TCP receive window size prevents TCP from fully utilizing the available bandwidth.

In this work we experiment with various TCP buffer sizes and demonstrate that both the send and receive window sizes should be adjusted to utilize the bandwidth of WiBro network to its fullest.

3. MEASUREMENT ENVIRONMENT

3.1 Overview

In all our measurements we use a mobile node (MN), equipped with a WiBro Modem *(SWD-H300K)* and a desktop PC, called a corresponding node (CN), connected to KREONET (Korea Research Environment Open NETwork), a research network in Korea. KREONET is directly connected to KT's IP backbone network and has low utilization such that we assume packet losses and delays to occur mostly on the WiBro network. Both the MN and CN run Windows XP.

We conducted all our experiments from a single location in Daejeon, Korea, and the data was collected from April to May, 2009. The 802.16 Standardization Forum defines two separate steps of certification, conformance and interoperability testings, and each step is known as Wave. Wave1 has the basic characteristics of mobile WiMAX. Wave2 has enhanced features in the physical layer and MAC layer, including MIMO (Multiple Input Multiple Output) technology, scalable OFDMA and beam forming. The KT WiBro service was upgraded to Wave2 in September 2008.

We use Iperf [1] for traffic generation and use WinPcap [2] to capture transmitted packets. Original WinPcap creates a timestamp using local time. We modify WinPcap to mark each packet with CPU cycles instead of local time for ease of time synchronization. Further details about time synchronization follow in the next section.

3.2 Clock Synchronization

In order to measure one-way delay, we need two end hosts to be synchronized. Two popular methods to synchronize computers are NTP (Network Time Protocol) and GPS (Global Positioning System). NTP is widely regarded as inadequate for one-way network delay measurement [11]. Veitch *et al.* have proposed a robust clock synchronization mechanism based on the TimeStamp Counter (TSC) register of Pentium class PCs [10]. Their approach requires kernel patches but no support is available for Windows XP. In order to conduct outdoor measurement, we require a small low-power GPS time synchronization device. Unfortunately, to the best of our knowledge, a small off-the-shelve GPS time synchronization device is not available. We develop a small GPS time synchronization device that provides accurate UTC (Universal Time Coordinated) information.

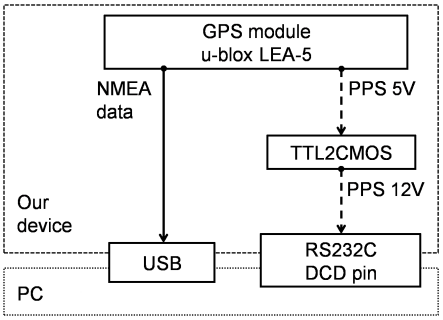


Figure 1: The Structure of GPS module

Figure 1 is a structural diagram of our synchronization device. Our device consists of the GPS module u-blox LEA-5, a USB interface, an RS232 interface, and LAN cables connecting the GPS module to the two interfaces. The GPS module outputs the NMEA (The National Marine Electronics Association) 0183 signal and a 5V pulse-per-second (PPS) signal. The NMEA 0183 application data sentences include the UTC time, the geographic position, and the moving velocity of the module. A typical GPS device in today's market delivers the NMEA signal via USB for location-based service applications and does not use the PPS signal. However, the USB interface adds fluctuating delay up to 50 ms and is inadequate for our purpose. We rewired the output of the GPS module such that the NMEA signal connects to the PC via USB, and the PPS signal reaches the PC via RS232C. Upon receipt of the PPS signal, the PC records the CPU cycles counted from the machine startup. As we describe earlier, our modified WinPcap records the CPU cycles for each packet. By aligning the CPU cycles recorded at every PPS signal with those recorded per packet, we can infer the time of each packet accurately in a globally synchronized manner.

We test our GPS synchronization device in a LAN environment. We connect two computers via an Ethernet switch. The two computers send and receive UDP traffic and we use our synchronization device and mechanism to calculate one-way delay for both directions. One-way delays for both directions are positive and strictly smaller than 0.8 ms, but greater than 0.2 ms. This indicates our synchronization mechanism provides sub-millisecond accuracy. Sub-millisecond accuracy is adequate in measuring the latency of WiBro that we expect to be in tens of milliseconds in both directions.

4. BASIC CHARACTERISTICS OF WIBRO

We begin our WiBro performance evaluation with UDP traffic. We measure the one-way delay of low rate UDP traffic and calculate the minimum delay of the WiBro network. Then we generate enough traffic to saturate the WiBro link in order to see the maximum throughput and one-way delay under congestion. From the

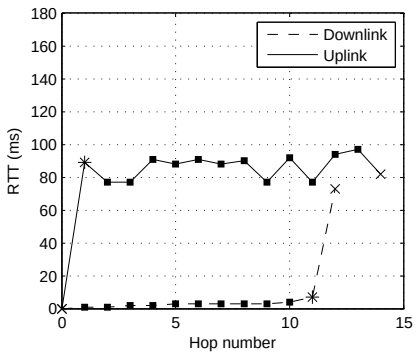


Figure 2: RTT by the hop

worst-case one-way delay, we estimate the buffer size at both the uplink and downlink directions.

All traffic we collect is 5-minute-long traces of UDP traffic generated by Iperf. Our measurement location has CINR (Carrier to Interference plus Noise Ratio) values higher than 30 dB and variance of less than 1. This is the best signal strength in WiBro. For reference, the signal strength we measured at various locations in downtown Seoul was between 3 dB to 25 dB.

4.1 Validation of Our Experiment Setting

In order to study the characteristics of the WiBro link, we first check to see if the WiBro link is the bottleneck in the end-to-end path of our experiment setting. We use two tools `ping` and `tracert` in order to show that the WiBro link accounts for the majority of delay and jitter, as well as loss. RTT at each hop in Figure 2 is the minimum from 15 runs of `tracert`. The first hop from the MN in the uplink direction and the last hop before the MN in the downlink direction did not respond to `tracert`. We looked at the DHCP configuration and obtained the default gateway. We then sent `ping` to the default gateway and found the gateway to answer to `ping` requests. We mark delays obtained from `ping` in the figure with an asterisk to distinguish from `tracert` measurements. There are 14 hops from the CN to the MN (uplink) and 12 hops the other way. This path asymmetry is at the router level, not at the AS path level, and is due to AS-internal network configurations. The uplink RTT measurements exhibit larger variation. It is because the WiBro link is included in every measurement, while the downlink RTT measurement includes the WiBro link only at the last hop. From the RTT measurements in both directions we are convinced that the majority of the one-way delay in our measurement setting comes from the WiBro link.

We have checked the link utilization on all links from the CN to the link between KT and KREONET and seen that they were never utilized over 50% during our measurement experiment.

4.2 Minimum one-way delay

In order to measure the minimum one-way delay and loss (or the best performance), we use 4 Kbps UDP traffic (20 bytes of payload every 40 ms) and measured the one-way delay and loss. The downlink delay has very little variance and has the minimum of 14 ms. The uplink delay fluctuates between 25 ms and 100 ms. The main reason behind this variation is packet bundling, similar to piggybacking, as explained in Section 2. Packet bundling takes place when one packet is scheduled to be delivered uplink, another packet arrives and is delivered along with the first packet, thus saving scheduling time. The difference between bundling and piggybacking lies in what gets opportunistic scheduling advantage, the packet itself or a request for a transmission slot. Without bundling the minimum uplink delay is 80 ms, and with bundling the delay reduces to 25 ms.

We have shown that the RTT of the wired portion is less than 7 ms and one-way delay of wired link is less than 4 ms. Therefore, the one-way delay of WiBro link is about 76 ms uplink and 10 ms downlink. During this measurement, no packet was lost.

4.3 Performance of a saturated WiBro link

Now we turn our attention to WiBro performance under heavy traffic. In order to saturate the WiBro link, we generate 5 Mbps and 12 Mbps for uplink and downlink, respectively. We use the packet size of 1410 bytes. Figure 3(a) plots the throughput at the receiving end, calculated per second. The maximum uplink throughput is 2.5 Mbps and downlink 10 Mbps. The rest of the generated traffic is dropped. The one-way delay is expected to be larger than what we report in Section 4.2, as the queues before the Wibro link build up due to heavy traffic. Figure 3(b) plots the first 3 seconds of one-way delay of the same trace and Figure 3(c) shows the cumulative distribution function (CDF) of one-way delay. We see the one-way delay gradually increasing as the queue builds up over time. The one-way delay increases until the queue is full and starts to drop packets.

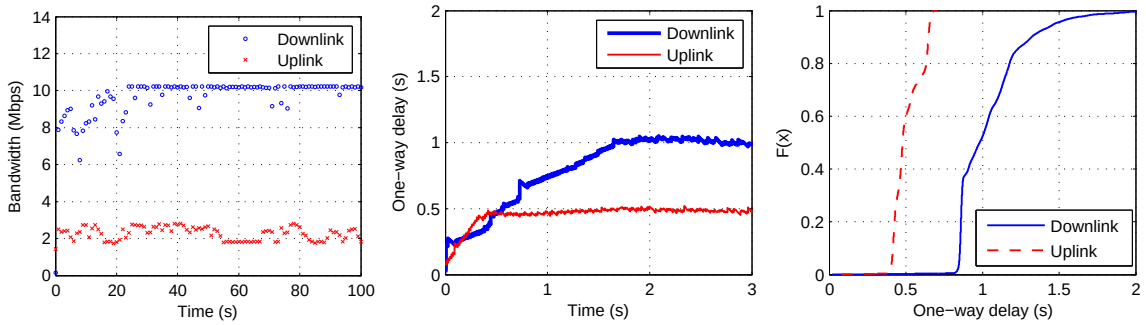
As we use packets of a different size from Section 4.2, we measure the minimum one-way delay again, this time with 256 Kbps uplink and 1 Mbps downlink traffic. The minimum one-way delays are 30 ms uplink and 15 ms downlink, larger than reported in Section 4.2. This increase in delay could easily be explained with increase in transmission delay, as the packet size grew from 62 to 1410 bytes.

The one-way delay when the queue is full translates to the worst-case performance of the network. Because we generate more traffic than maximum measured throughput, the queue is never drained once the queue has built up to its full capacity. We calculate the minimum one-way delay excluding the first and last 5 seconds to assure that the queue remains full. The minimum delay when the queue is full includes the full queueing delay, while minimizing the effect of HARQ and low signal variation. The difference between minimum one-way delay with and without queueing delay translates to the queue size. We estimate the queue size in bytes by multiplying the queue size in time and the bandwidth at the time. The estimated queue sizes are about 110 KB uplink and 1,100 KB downlink. We note that the downlink has 10 times larger queue than uplink, although the bandwidth is only 5 times larger. The delay larger than what can be accounted for queueing in Figure 3(c) is likely to be due to HARQ retransmissions, scheduling, and variation in signal strength. Without access to PHY and MAC layer information, we can not verify how much each contributes to the overall delay. About half the packets experience more than 1 s delay downlink when the queue is saturated. The large buffer size in the wired network has been a topic of hot debate[4]. The large buffer size in wireless network can be as serious or doubly serious as we observe in this work.

5. ANALYSIS OF TCP PERFORMANCE

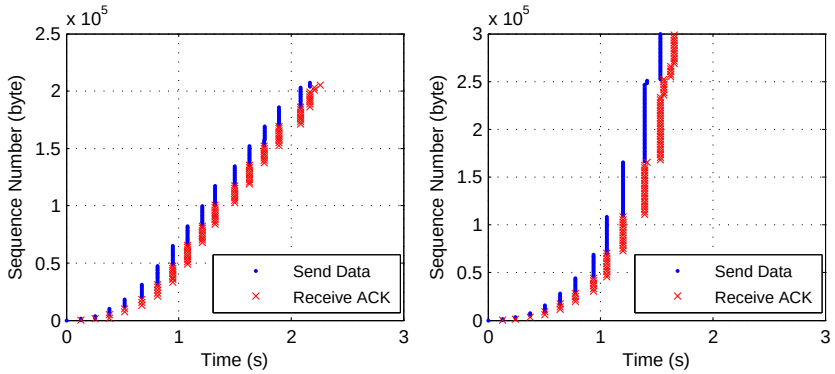
In this section we measure and analyze the TCP performance. We first measure the TCP throughput and compare it with that from UDP. We also measure and compare the TCP delay against the minimum round-trip time and one-way delay of UDP traffic.

All traffic we collect is 5-minute-long traces of TCP traffic generated by Iperf. We collect TCP traces with different TCP socket buffer sizes. We increase Windows XP's default TCP socket buffer



(a) Bandwidth (b) One-way delay (c) One-way delay (CDF)

Figure 3: UDP performance over a saturated WiBro link



(a) Sender 17KB / Receiver 64KB (b) Sender 256KB / Receiver 256KB

Figure 4: Sequence graph in slow start (Downlink)

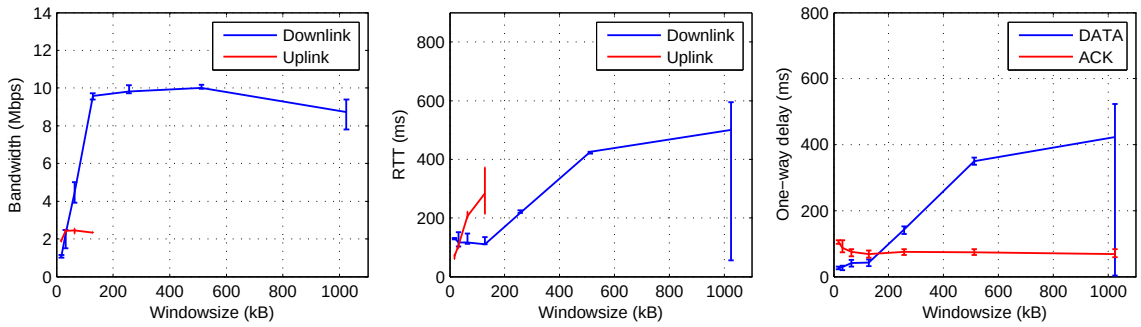
size to 1 MB using RFC 1323 [7] option and change the TCP socket buffer size of each TCP flow by setting the option in the application layer.

We measure the TCP throughput with Windows XP's default settings and it comes out to be only about 1 Mbps, ten times smaller than UDP's. Even taking into consideration the undulating nature of TCP traffic due to congestion control, we find tenfold decrease to be too significant. In order to identify the cause of the small throughput, we investigate how packets are transmitted and whether many packets are retransmitted. Figure 4(a) plots the sequence numbers of both data and acknowledgement packets against time. In this experiment we use the Windows default socket buffer sizes: 17 KB for the send socket buffer and 64 KB for the receive side. Figure 4(a) shows that after a few rounds in the slow start phase, the packets worth of the full send buffer size are transmitted, but the send buffer size does not grow in the next round. That is, the TCP congestion window size reaches the maximum of 17 KB and does not increase any more. This literally caps the TCP throughput.

Bandwidth-delay product indicates the maximum amount of data that can be in transit from a sender to a receiver and is used in provisioning the buffer size inside the network. If the available bandwidth of an end-to-end path is small or delay short, the bandwidth-delay product is small and a small send buffer size is sufficient to keep the pipe full. In case of WiBro, the delay is relatively large, but the bandwidth is also large, which in turn increases the bandwidth-delay product. However, with the TCP send buffer size capped at 17 KB, a single TCP flow cannot exploit the full capacity of the WiBro network. Now that we understand the cause of low TCP throughout, we conduct the same experiment with 256 KB for both TCP send and receive buffer sizes and plot the sequence numbers against time in Figure 4(b). In contrast to Figure 4(a) the congestion window continues to grow after the first second in Figure 4(b).

Kim *et al.* have shown that the small receive window size of Windows is indeed the bottleneck in WiBro [8]. According to their results, the TCP throughput increases when the receive window size is changed from 17 KB to 64 KB. Halepovic *et al.* show that increased throughput of TCP with increasing socket buffer sizes up to 64 KB and auto-tuned socket buffer size. They show that 64 KB is enough to support 1.5 Mbps in WiMAX [5]. We have found out that not only the receive window size, but also the send socket buffer size affects the performance of TCP. When a TCP sender receives an ACK, it can grow its window size, but never over the limit exceeding the sender window size. As our WiBro environment supports much higher bandwidth and low loss rate, we conduct the next experiment varying both the send and receive buffer sizes.

We vary the send and receive buffer sizes from 17 to 32, 64, 128, 256, 512, and 1,024 KB and measure the TCP throughput. We see in Figure 5(a) that only with the buffer size set at 128 KB the downlink TCP throughput reaches about 10 Mbps comparable to that of UDP traffic. If the buffer size grows over 512 KB, the single TCP flow induces queueing and loss onto itself and experiences reduced throughput.



(a) Bandwidth (b) RTT (c) One-way delay (Downlink)

Figure 5: Characteristics TCP traffic in Wave2

Samke *et al.* have proposed that auto-tuning of send buffer size for high-speed WAN networking environment (in their times in the order of 100 Mbps). We find it rather interesting to see their auto-tuning to apply these days to wireless networking of drastically increased bandwidth. Yet still our work suggests that not only the send buffer size, but also the receive buffer size should increase. Linux 2.6 and later versions implement auto-tuning of both send and receive buffer sizes. Windows Vista implements receiver-side auto-tuning. We leave evaluation of Linux and Windows Vista's auto-tuning mechanisms for future work.

We then examine the RTT in our TCP experiments. Figure 5(b) plots the median and inter-quartile RTTs at various buffer sizes. As the buffer size grows, so does RTT, indicating queueing. Once the buffer size grows over $1,024$ KB, the RTT fluctuates greatly. This indicates that with the buffer size of $1,024$ KB, the single TCP flow enters congestion regime and the queue is drained after timeouts and the shrunk congestion window size. The one-way delay follows similar pattern as the RTT.

6. CONCLUSION

In this work we measure the best-case performance of a WiBro network for a single flow. Our measurement shows that the maximum throughput over the WiBro network is 10 Mbps downlink and 2.5 Mbps uplink. We develop a GPS synchronization device to measure one-way delay. The minimum one-way delay in downlink is 11 ms and that in uplink varies from 21 ms to 76 ms depending on the packet bundling. We estimate that both the base station and WiBro modems have large buffers up to 2 s and 500 ms and minimum one-way delay of 11 ms and 76 ms, respectively. We measure TCP throughput over WiBro by varying send and receive buffer sizes. Due to high bandwidth and long delay over the WiBro network, at least 128 KB is required to fully utilize the link.

Acknowledgement We would like to thank people from KT, Boyoun Chang, Jung-Sik Park for their valuable comments and Hyungsoo Kim for granting us the use of their testbed. This work was supported by the IT R&D program of MKE/IITA [A1100-0801-2758, "CASFI : High-Precision Measurement and Analysis Research"].

7. REFERENCES

[1] Iperf. http://sourceforge.net/projects/iperf/?abmode=1.

[2] Winpcap. http://www.winpcap.org/.

[3] J. G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX*. Prentice Hall, 2007.

[4] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Routers with very small buffers. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006.

[5] E. Halepovic, Q. Wu, C. Williamson, and M. Ghaderi. TCP over WiMAX: A measurement study. In *IEEE/ACM MASCOTS 2008*.

[6] M. Han, Y. Lee, S. Moon, K. Jang, and D. Lee. Evaluation of voip quality over wibro. In *PAM*, 2008.

[7] V. Jacobson, R. Braden, and D. Borman. RFC1323: TCP extensions for high performance, 1992.

[8] D. Kim, H. Cai, M. Na, and S. Choi. Performance measurement over mobile WiMAX/IEEE 802.16e network. In *WOWMOM*, 2008.

[9] B. G. Lee and S. Choi. *Broadband Wireless Access and Local Networks: Mobile WiMAX and WiFi*. Artech House Publishers, 2008.

[10] V. Paxson. On calibrating measurements of packet transit times. *SIGMETRICS Perform. Eval. Rev.*

[11] D. Veitch, J. Ridoux, and S. B. Korada. Robust synchronization of absolute and difference clocks over networks. *Accepted for publication, IEEE/ACM Transactions on Networking, to appear June 2009*.