석 사 학 위 논 문

Master's Thesis

상용 와이브로 망에서 UDP/TCP 플로우와 어플리케이션의 성능 측정

Measured Performance of Flows and Applications over Commercial WiBro Network

우 신 애 (禹 伸 愛 Woo, Shinae)

전자전산학부, 전산학전공

School of Electrical Engineering and Computer Science

Division of Computer Science

KAIST

2010

상용 와이브로 망에서 UDP/TCP 플로우와 어플리케이션의 성능 측정

Measured Performance of Flows and Applications over Commercial WiBro Network

Measured Performance of Flows and Applications over Commercial WiBro Network

Advisor : Professor Moon, Sue Bok

by

Woo, Shinae
School of Electrical Engineering and Computer Science
Division of Computer Science
KAIST

A thesis submitted to the faculty of the KAIST in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the School of Electrical Engineering and Computer Science, Division of Computer Science

Daejeon, Korea
2009. 12. 5.
Approved by

_____

Professor Moon, Sue Bok
Advisor

상용 와이브로 망에서 UDP/TCP 플로우와 어플리케이션의 성능 측정

우 신 애

위 논문은 한국과학기술원 석사학위논문으로 학위논문심사 위원회에서 심사 통과하였음.

2009년 11월 30일

심사위원장 장 기 주 (인)

심사위원 김 종 진 (인)

심사위원 신 성 철 (인)

MCS 20083308 우 신 애. Woo, Shinae. Measured Performance of Flows and Applications over Commercial WiBro Network. 상용 와이브로 망에서 UDP/TCP 플로우와 어플리케이션의 성능 측정. School of Electrical Engineering and Computer Science, Division of Computer Science . 2010. 44p. Advisor Prof. Moon, Sue Bok. Text in English.

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. We developed 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 meausred TCP throughput over WiBro by varying a send buffer and a receive buffer sizes. To fully exploit the high bandwidth of Wibro, the auto-tuning feature of TCP is needed along with the minimum of 128 KB buffer size.

Contents

List of Tables

List of Figures

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 3.6 Mbps and 284 Kbps 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 services in Korea in 2006 and it has since acquired more than two hundred thousand 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 the performance of WiBro. First, the infrastructure-based WiBro is yet to be widely deployed, and only a few 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 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. We also analyze the performance of a single TCP flow. We have found that the small TCP send buffer size has become 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 auto-tuning feature of TCP is needed along with the minimum of 128 KB 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 the representative previous measurement research on WiBro as related works 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 and inspect the existence of split TCP implementation. Then we demonstrate how send buffer size and receive window size of TCP session affect TCP bandwidth. Finally, we conclude this paper in Section 5.

2. Background and Related Work

2.1 Mobile WiMAX and WiBro

IEEE 802.16 Working Group (WG) is organized in 1999 for standardization of Broadband Wireless Access (BWA). Since the committee's organization, 802.16-2004 standard for PHY and MAC layer of fixed BWA service was approved in 2004. In parallel, Telecommunications Technology Association (TTA) of Korea started to lead standardization of mobile WiMAX, called WiBro in 2003 and and Telecommunications Research Institute and Samsung Electronics complete the WiBro phase 1. 802.16 TGe and TTA harmonized of their standards and include enhanced feature such as HARQ, MIMO.[17]

WiMAX forum which consist of industrial members is organized for the commercialization of IEEE 802.16 standard. They are working on balancing the customer and producer's meet, Inter Operability Testing (IOT) and promotion. They divides the certification process, conformance and interoperability testings about WiMAX systems, into two steps and each step is know as Wave. Wave1 has the basic characteristics of mobile WiMAX and Wive2 has enhanced features in the PHY and MAC layer including MIMO, sOFDMA and beam forming.

Strated with the reallocation of the 100-MHz frequency at 2.3GHz spectrum for WiBro services in 2002, Samsung, KT, and SKT lead the Mobile WiMAX industry. Samsung Electronics developed the world's first mobile WiMAX system and KT also deployed the world's first commercial WiBro services in the Seoul. The KT WiBro service is compatible with IEEE 802.16e standard and it was upgraded to Wave2 in September 2008.

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 [17]. 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 [10, 17]. 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.

The main components of WiBro are ACR (Access Control Router) and RAS (Radio Access Station). ACR, which is connected to directly from KT's IP Networks, has the responsibility of IP and MAC layer processing such as IP packet routing and Quality of Service (QoS) control. It also controls RASes. The RAS has the responsibility of physical layer processing such as allocating spectrum and packet retransmission. It is connected to WiBro modem over wireless. The only wireless part in WiBro is the links between RASes and WiBro modems. There exists repeaters between a RAS and a WiBro Modem to enhance signal strength.

WiBro MAC layer, PHY layer

2.2 Measurement Study

WiFi

Cellular Network

Han *et al.* have evaluated Voice over IP (VoIP) performance over WiBro [13]. Their results have shown 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 [16]. They have demonstrated that the small default TCP receive window size prevents TCP from fully utilizing the available bandwidth.

2.3 TCP over wireless

Fluctuated bandwidth and high loss rate of wireless link cause a degradation of TCP performance. Several methods to improve TCP performance over wireless link has been provided. ARQ (Autonomic Repeat re- Quest) is a link layer level solution. It rapidly retransmit the loss block until it is successfully transmitted. ARQ hide link layer losses from higher layers. The other way of hiding link layer losses from TCP hosts is using proxy[ITCP, M-TCP, Split-TCP, Snoop-TCP]. A TCP session is split into two session in proxy and losses and fluctuated bandwidth of wireless link do not affect the other TCP session. These link layer and proxy solution do not need modification of TCP stacks. Finally, lots of methods modifying each end-to-end protocol have been proposed. New Reno introduce the fast recovery mechanism to overcome the low performance of Reno when burst loss is occurred. Vegas and Westwood control congestion by estimating available bandwidth. Veno and TCP-Jersey suggests a way to differentiate the cause of packet loss and reduce the congestion

window with only congestion cases.

2.4 Extended TCP for Wireless

Windows socket buffer size

- RFC 1323: TCP extensions
- New TCP in Windows Vista
- Linux 4.2 and 4.6

3. Experimental Environment

3.1 Overview

Figure 3.1 describes the topology of our experiment environment. 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) [3], 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 in WiBro. Mostly, both the PC and the laptop operate on Windows XP.

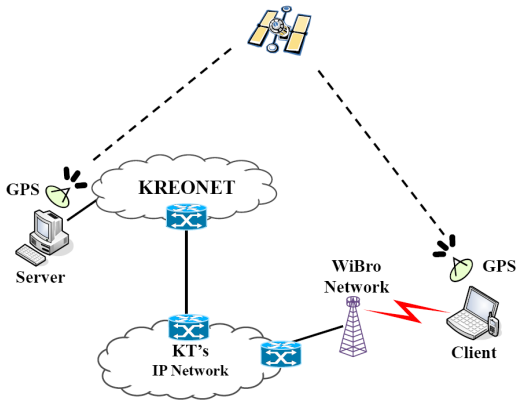
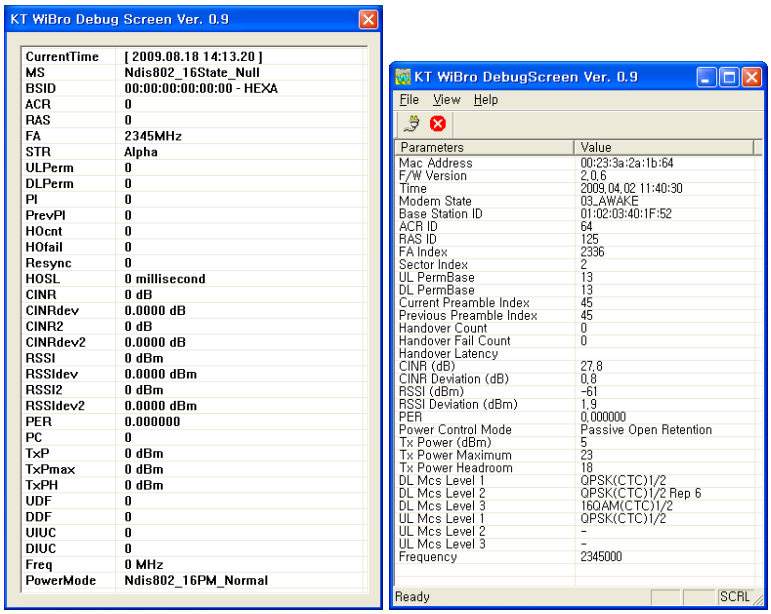


Figure 3.1: The measurement testbed

The commercial WiBro service is deployed in Seoul, satellite cities of Seoul, and a few hotspots in other cities including Daejeon in Korea. We conducted all our experiments in Daejeon for its accessibility from KAIST. Our data set was collected from April to May, 2009. WiBro is known to implement five different QoS levels. However, KT does not support QoS yet, and all of our measurements were conducted with BE service. WiMax Forum divides the certification process, conformance and interoperability testings about WiMAX systems, into two steps 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. Our experiments are conducted on both Wave1 and Wave2 services.

As we mentioned before, it is hard to get detailed information or control parameters of base stations on WiBro. Lack of information makes it hard to analyze the cause of performance characteristics. With the cooperation of KT, the WiBro service provider, we can get more detailed information of WiBro Connection from the debug screen of WiBro CM. WiBro Connection Manager (WiBro CM) installed on the mobile node manages WiBro connection. The information we can get is ACR and RAS ID, MAC address, CINR, RSSI, up/down link MCS (Modulation and Coding Scheme) level and frequency which is used for sending signals. Figure 3.2 shows the WiBro CM debug screen and its parameters.



(a) KWM-U1000 (b) SWD-H300K

Figure 3.2: KT WiBro debug screen

We use *Iperf* for traffic generation and use *WinPcap* to capture packets transmitted through a specified network device. Original *WinPcap* creates a timestamp using local time and CPU cycles. We modified *WinPcap* to mark each packet with CPU cycle instead of local time for ease of time synchronization. Further details about time synchronization are explained in the next chapter4.

3.2 Validation of Experimental configuration

Before experiments on WiBro, we first check to see if the WiBro link is the bottleneck in the end-to-end path of our experiment setting in terms of delay and bottleneck.

We use two tools `ping`[5] and `tracert`[7] in order to show that the WiBro link accounts for the majority of delay and jitter. RTT at each hop in Figure 3.3 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 marked 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.

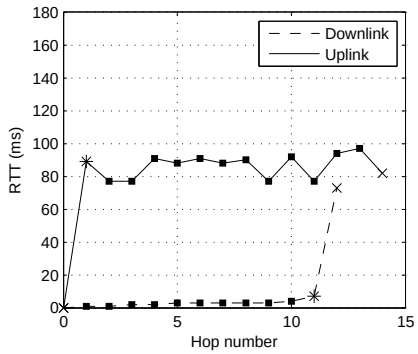


Figure 3.3: RTT by the hop

We have also checked the link utilization on all links from the CN to the link between KT and KREONET. Figure 3.4 is the KT-KREONET MRTG Graph[4] by 5 minute average. The light line (green line in color) indicates KREONET to KT traffic and the dark line (blue line in color) represents KT to KREONET traffic. Bandwidths of both traffic are less than 400Mbps and this is far less than the maximum bandwidth between KT and KREONET. From our extensive observations of the KT-KREONET MRTG Graphs, we have seen that they were never utilized over 50% during our

measurement experiment. Therefore, we assumed that the wired link has a relatively light weight usage and the wireless link is the bottleneck in terms of bandwidth and congestion.

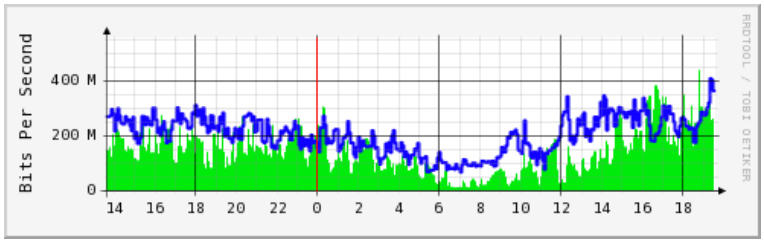


Figure 3.4: KT - Kreonet MRTG Grapn (updated at 26 April, 19:40:14 KST)

4. GPS Synchronization Techniques

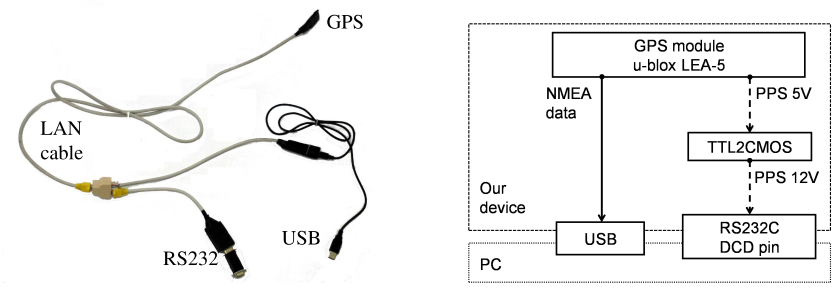
In order to measure one-way delay, we need two end hosts to be synchronized. In this chapter, I briefly introduce widely used synchronization techniques, and our synchronization method.

NTP is widely used for synchronizing packet switched networked computer[19, 18, 20]. NTP has a hierarchical architecture and each level is called *stratum*. The highest level, *stratum 1* has a local clock sources such as GPS or atomic clocks. These server is act as time servers and provide clock sources to next level, called *stratum 2*. Stratum 2 servers also provide clock sources to stratum 3 servers. The stratum numbers indicate the distance from accurate clock sources but smaller number can not assure the more reliability and accuracy.

NTP is widely regarded as inadequate for time difference measurement. NTP adjust its time by changing the rate of clock and it is inadequate for measuring delay variance[21, 22]. Veitch *et al.* have proposed a robust clock synchronization mechanism based on the Time Stamp Counter (TSC) register of Pentium class PCs [22] rather than Software clock. TSC is robust and high resolution enough to provide clock source. Their method need symmetry one-way delay and small RTT between host and time server.

In WiBro measurement, we can not use NTP or TSCclock. WiBro, or Mobile WiMAX, has inherently asymmetric delay and uplink delay is vary because of its scheduling mechanism. In addition, 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 developed a small GPS time synchronization device that provides accurate UTC (Universal Time Coordinated) information.

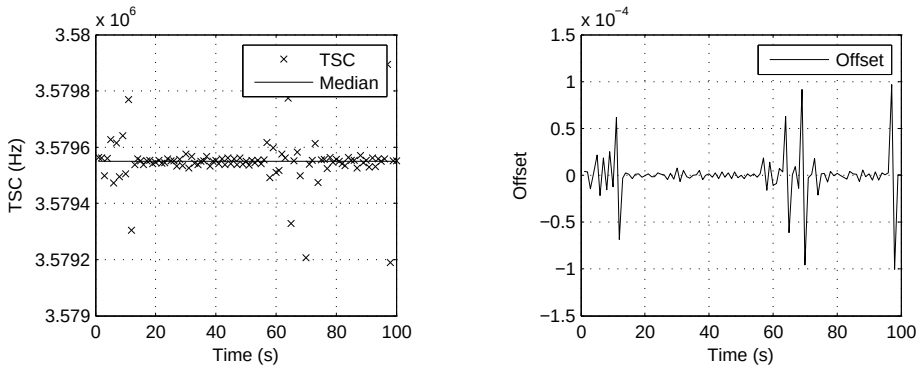
Figure 4.1 is a picture and structural diagram of our synchronization device. Our device consists of the GPS module u-blox LEA-5, a USB interface, an RS232 interface, and LAN cables connecting the GPS module to the two interfaces. The GPS module outputs the NMEA (The National Marine Electronics Association) 0183 signal and a 5 V 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 RS232 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 rewire 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.



(a) Picture (b) Structure

Figure 4.1: GPS synchronization device

Upon receipt of the PPS signal, the PC records the CPU cycles counted from the machine startup. The CPU cycles can be read via TSC (Time Stamp Clock) register using RDTSC, an assembly commend in Inter Pentium Computer and the commend takes only 6 to 11 clocks. I also modify WinPcap which captures the all packets through pre-specified network devices. Modified WinPcap records the CPU cycles for each packet. By aligning the CPU cycles recorded at every PPS signal with those recorded per packet, I infer the time of each packet accurately in a globally synchronized manner.



(a) TSC per second (b) Offset

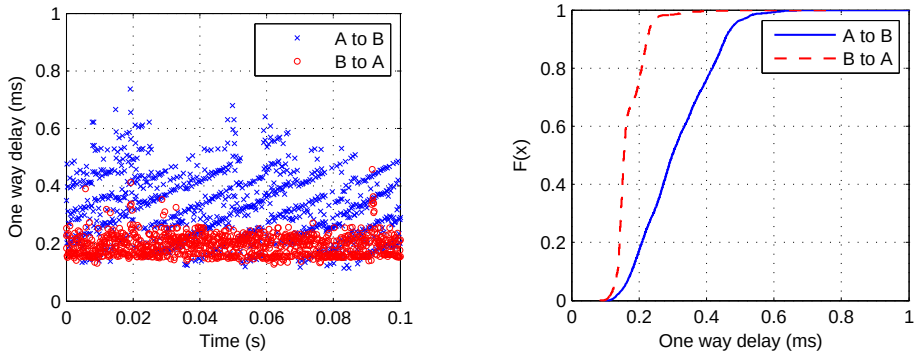
Figure 4.2: One-way delay of Ethernet (Both direction)

I can assume that the rate of CPU clock is constant in small time scale, like few seconds[22]. Therefore, I use linear fitting between every second to change each packet's CPU clock cycles to UTC time. Figure 4.2 (a) shows the difference of TSC values between every second. Sometimes,

the values is doubled or it has high variance due to system noisy . As we use only two TSC values to infer each packet's exact UTC time stamp, removing the noisy TSC values is critical. We couldn't use the mean value of the total TCS values for filtering, because few doubled TSC value make the mean value bigger. We filter the noisy TSC value using median. Figure 4.2 (b) shows the offset error of each value from median.

$$\text{Offset\_Error}_i = \frac{TSC_i - \widehat{TSC}}{\widehat{TSC}}$$

We use the 10^{-4} as threshold and throw the value which has bigger offset than the threshold.



(a) Time series

(b) CDF

Figure 4.3: One-way delay of Ethernet (Both direction)

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 used our synchronization device and mechanism to calculate one-way delay for both directions. In Figure 4.3, one-way delays for both directions are positive and strictly smaller than 0.8 ms, but greater than 0.2 ms. As our LAN environment has less than 100us of one-way delay, the positive delay 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.

5. Performances of UDP and TCP traffics in Wave1

I started WiBro performance evaluation with UDP traffic. I generate enough traffic to saturate the WiBro link in order to see the maximum throughput and one-way delay under congestion with different CINR (Carrier to Interface Ratio). All traffic we collected is 5-minute-long traces of UDP traffic generated by Iperf[2]. This is the best signal strength in WiBro.

5.1 Relations between bandwidth and CINR

I plot CINR and bandwidth with timescale in Figure 5.1. CINR (Carrier to Interface Ratio) is a quality basis of OFDMA technology system and measured by pilot power of device. High CINR value means high quality of signal. In Figure 5.1, the bandwidth is changed with CINR variation. With low and fluctuated CINR values, bandwidth is also low and fluctuated.

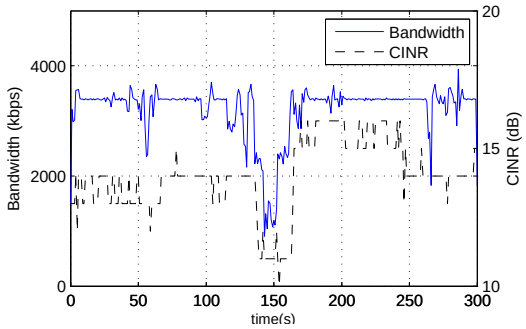
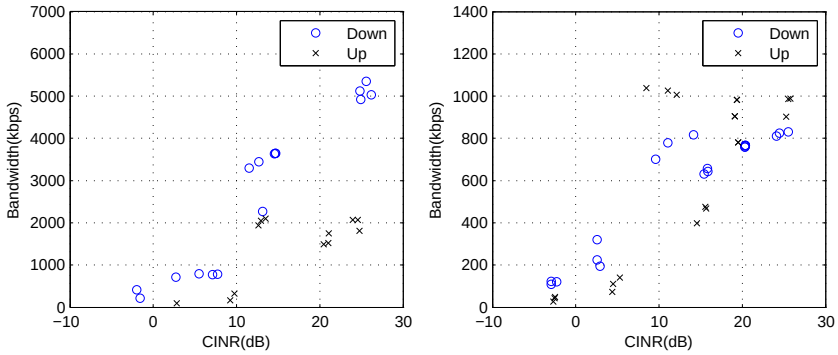


Figure 5.1: Relationship between bandwidth and CINR

To find out more relationship between CINR and bandwidth, I collect UDP and TCP traces in different location of CINR. The averaged CINR and bandwidth of each trace is plotted in Figure 5.2. The Figure 5.2 (a) shows that bandwidth of downlink and uplink can be divided into 3 groups and 2 groups each. WiBro select the most efficient MCS (Modulation and coding schema) level according to channel condition. Each MCS level affect the bandwidth. Each clustered bandwidth of downlink mapped with WiBro's 3 modulation schema QPSK, 16-QAM and 64-QAM. With more than 10dB of CINR, we could get more than 2Mbps of bandwidth and with 25dB of CINR, 5Mbps of bandwidth.

Compared with UDP traffic, TCP shows smaller bandwidth less than 1Mbps even with best

CINR values. This is because of Windows XP's default TCP socket buffer setting for WiBro. WiBro link has high bandwidth-delay product compared to the Ethernet and WiFi. Therefore, WiBro need more than 128KB buffer size. But Windows XP set the TCP socket buffer size as only 17KB. I'll explain this phenomena later in 6.3.



(a) UDP traffic (b) TCP traffic

Figure 5.2: Bandwidth vs. CINR

5.2 Delay characteristics

Delay is important characteristics of links. Delay affect the quality of services such as VoIP (Voice over IP) and Video conference. Delay is also important the performance of TCP. Longer delay makes slower growth of TCP window sizes. WiBro has asymmetric link characteristics - in both bandwidth and delay. So measuring one-way delay rather than RTT (Round trip time) is needed. To measure one-way delay, the two sending and receiving node have to synchronized by time. I used GPS synchronization technique explained in 4.

I saturated the link using UDP traffic and measured downlink one-way delay. The measured delay include transmission delay, propagation delay, scheduling delay and queueing delay. Figure 5.3 shows the delay of UDP downlink. The delay consistently increase up to 25 second and decrease 10's second. Most of measured traces of UDP, the delay is very high - tens of second with full queue. The large portion of one-way delay is from queueing delay. The large delay can degrade the quality of services like VoIP and video conference. However, I couldn't find out the portion of queueing delay and the others. There exist a lots of other factors in commercial networks such as signal interference and sharing link with other users.

Next, I measure the delay of TCP traffic. I compare the one-way delay with different signal

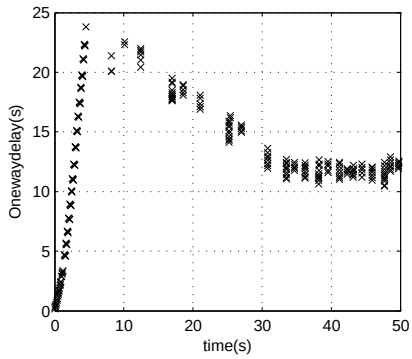
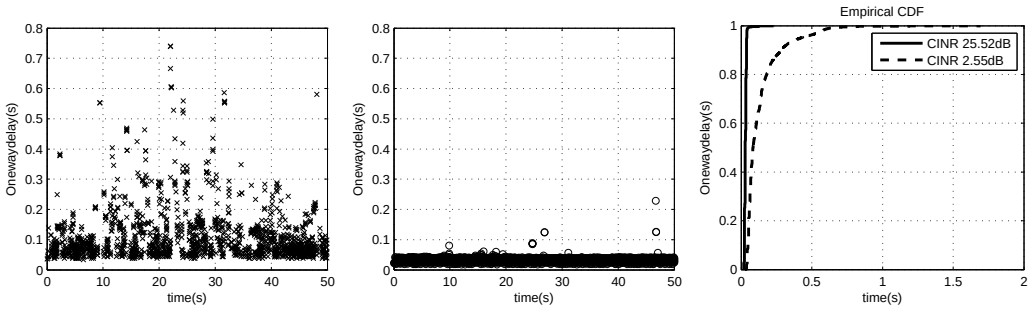


Figure 5.3: Onewaydelay of UDP Downlink

quality. Figure 5.4 (a) shows one-way delay with bad signal quality and (b) shows with good signal quality. Not much big delay as UDP, but TCP shows also quite big one-way delay, average 136ms, with bad signal quality. In case of good signal quality, the average is 31ms. In both case, high spikes are shown in the figure. These spikes is from HARQ (Hybrid Automatic Repeat reQuest) mechanism. The spikes on delay is also observed in WCDMA or HSDPA. Figure 5.4 (c) shows the CDF of both case one-way delay. Longer delay makes slower growth of TCP window size and lower throughput.



(a) CINR 2.55dB (b) CINR 25.52dB (c) CDF

Figure 5.4: One-way delay of TCP traffic in Wave1

5.3 Packet overwriting bugs

While conducting experiments on Wave1, I found a bug on packets. The packet we sent and receive have different payload and receive packet is thrown by failing checksum test. Figure 5.5 shows an example of bug. Matching sent and received packets can be easily done by comparing packet identification at IP header, sequence number at TCP header and front 10's bytes of payload. The sent packets has 1360 bytes of payload. But sometimes, the received packet has smaller payload. Futher more, the end of smaller packet's payload has same contents of same packets IP, TCP header and payload. Th tion.

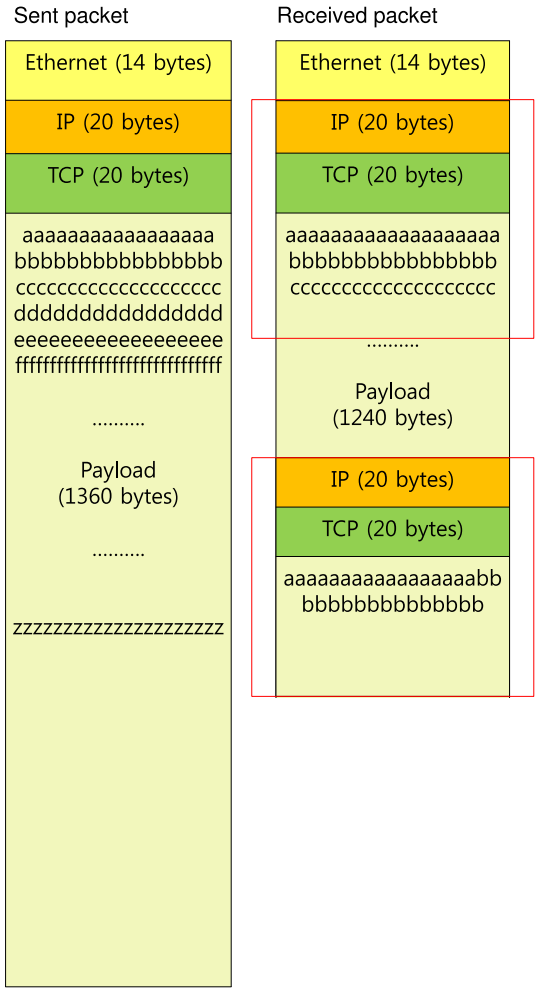


Figure 5.5: Packet overwriting bug

As I have shown, the experiments on WiBro wave 1 was very hard to pick major factors to affect performance. The reason is bellows:

First, there are too much factors affected the performance of WiBro. Our experiment was conducted in commercial networks, and the link is shared with the other user. But there is no way to check the other user's existence or their traffic. I couldn't remove the other user's interference. Also, the signal quality is fluctuated. Second, I couldn't saturate the link with TCP because of Windows XP's socket buffer setting. The maximum throughput of TCP I measured is only 1Mbps and it is far less than UDP's. Third, the packet overwriting bug I found can affect the performance. The corrupted packet regards as lost packets. When the bugs occurs severally, TCP reduce the window size or timeout occurs.

For this reason, I re-conduct experiments on KT testbed configured with WiBro Wave2. The testbed has same configuration with commercial network but I can exclude the other user's interference. Also the packet overwriting bug is removed for Wave2. I'll talk about the experiments result at next chapter.

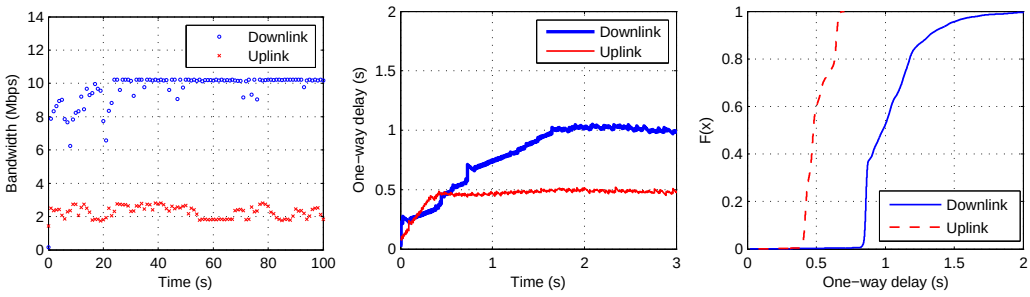
6. Performances of UDP and TCP traffics in Wave2

6.1 UDP Performances on Wave2

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

All traffic we collected is 5-minute-long traces of UDP traffic generated by Iperf. Our measurement location had 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. The signal strength we measured at various locations in downtown Seoul was between 3 dB to 25 dB.

6.1.1 UDP performance over Wave2



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

Figure 6.1: UDP performance over a saturated WiBro link

6.1.2 Minimum one-way delay

In order to measure the minimum one-way delay and loss (or the best performance), we used 4 Kbps UDP traffic (20 bytes of payload every 40 ms and measured the one-way delay and loss. The down-

Links	Dates	Min Delay (ms)	Bandwidth (Kbps)	Queue size (KB)
Up	May 8	409	2,650	108
	May 12	417	2,628	111
	May 22	380	2,876	109
Down	May 08	815	10,479	1044
	May 12	961	9,603	1133
	May 22	839	10,479	1077

Table 6.1: Queue sizes in WiBro links

link 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 reduced 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.

6.1.3 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 6.1(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 6.1.2, as the queues before the Wibro link build up due to heavy traffic. Figure 6.1(b) plots the first 3 seconds of one-way delay of the same trace and Figure 6.1(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 6.1.2, 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 6.1.2. 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 generated more traffic than maximum measured throughput, the queue was never drained once the queue had built up to its full capacity. We calculated the minimum one-way delay excluding the first and last 5 seconds to assure that the queue remained full. The minimum delay when the queue was 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 6.1(c) is likely to be due to HARQ retransmissions, scheduling, and variation in signal strength. Without access to PHY and MAC layer information, we could not verify how much each contributed 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. The large buffer size in wireless network can be as serious or doubly serious as we observe in this work.

6.2 Existence of Split TCP

Initially, TCP was designed for wired networks. In wired networks, packet losses are mainly caused by network congestion. Therefore, when a TCP sender detects lost packets, the TCP sender decreases its sending rate to react to network congestion. On the other hand, in wireless networks, packet losses occur not only by network congestion but also wireless link error. Though TCP does not distinguish whether a packet loss is due to network congestion or wireless link error, TCP reduces its sending rate to react to every packet loss event unnecessarily which results in low TCP performance over wireless network.

To solve the problem of TCP over wireless network, a wireless router (e.g., base station, ACR) processes the TCP layer to enhance TCP performance. Split-TCP and TCP proxy are some examples. Split-TCP splits the TCP session into two concurrent sessions. The router of the split point receives the TCP packets and sends acknowledgment (ACK) packets as replies instead of the real receiver. The packets are then forwarded to the receiver. As a result, the TCP session is divided and each TCP session works for only wired or wireless networks and the problem is resolved. In the case of TCP proxy, the proxy server (a router of the split point) firstly gets all TCP packets from the sender and then forwards the packets to the receiver. The difference between Split-TCP and

TCP proxy is that the former manages two concurrent TCP sessions and the latter manages one TCP session at a time.

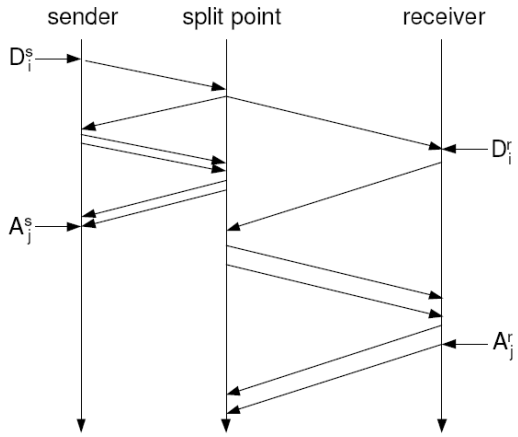


Figure 6.2: Finding Split TCP

Wei et al. presented the inference techniques to detect whether a network implements split-connection [23]. They also found that three commercial cellular networks use the split-connection TCP in practice. Figure 6.2 illustrates a split TCP example. When a split point receives a packet, it sends an ACK packet instead of the receiver. If the sender and receiver are synchronized, we can inspect the existence of a split connection by comparing the timestamp of the ACK packet. If a sender receives an ACK packet before it is sent by the receiver, this implies that the network is implementing split-connection. As we have synchronized time between the server and the client, we can easily inspect whether commercial WiBro implements split TCP connection. Existence of split TCP is important because split TCP traffic has very different characteristics from the normal TCP traffic and many configurations at the sender side may not affect its peer host.

We found that there was no split TCP session in WiBro and this is shown in Figure 6.3. If split TCP is implemented in WiBro, the router in a split point sends ACK packets before it is sent by the receiver, and as a result, we should see a negative one way delay of ACK packets. Figure 6.3 shows the one way delay and we can conclude that there is no implementation of split TCP in the current version of commercial WiBro.

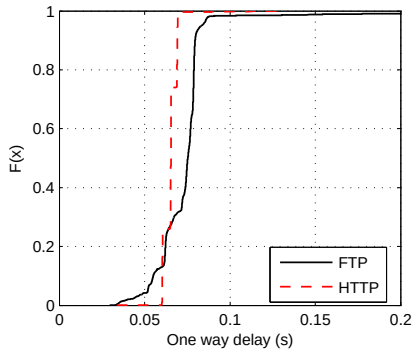
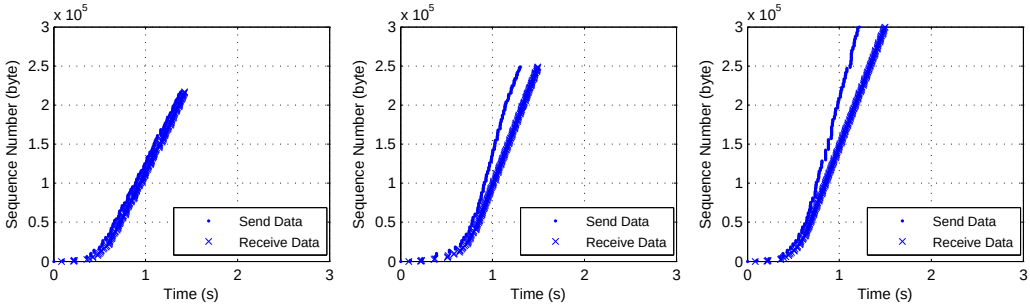


Figure 6.3: One-way delay of ACK (Downlink, cdf)

6.3 TCP Performances on Wave2

6.3.1 TCP performances on Windows XP



(a) Sender 17KB / Receiver 17KB (b) Sender 64KB / Receiver 64KB (c) Sender 128KB / Receiver 128KB

Figure 6.4: Sequence graph in slow start (Uplink)

In the previous section we use UDP traffic to measure the minimum one-way delay, maximum throughput, and queue size of the WiBro link. 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 collected is 5-minute-long traces of TCP traffic generated by Iperf. We collected TCP traces with different TCP socket buffer sizes. We increased Windows XP's default TCP socket buffer size to 1 Mbytes using RFC 1323 [14] option and changed the TCP socket buffer size of each

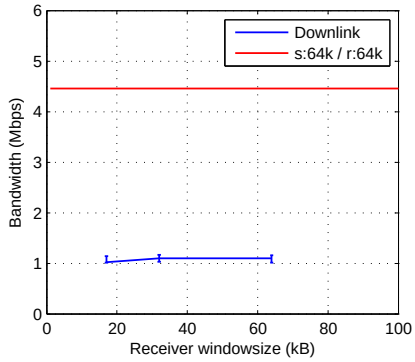
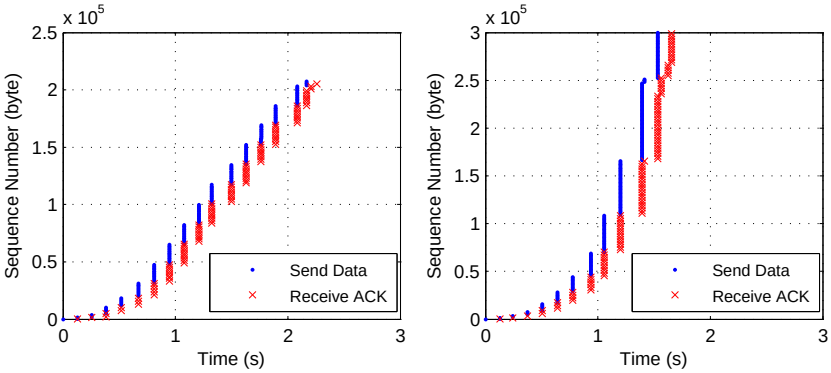


Figure 6.5: Bandwidth with different receive window size (Send buffer 17kB)



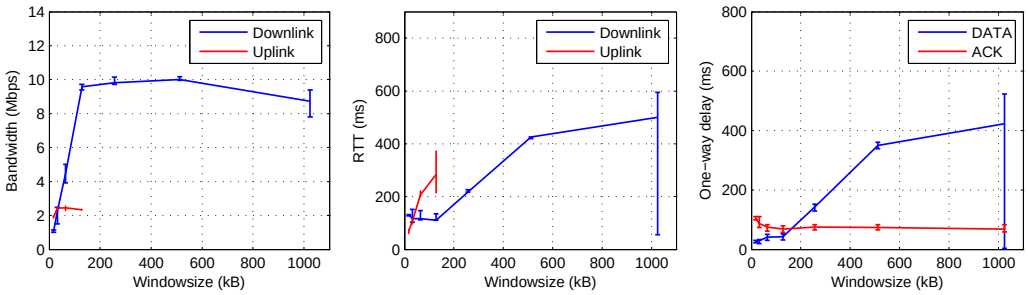
(a) Sender 17KB / Receiver 64KB

(b) Sender 256KB / Receiver 256KB

Figure 6.6: Sequence graph in slow start (Downlink)

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 6.6(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 6.6(a) shows that after a few rounds in the slow start phase, the packets worth of the full send buffer size are



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

Figure 6.7: Characteristics TCP traffic in Wave2

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 number against time in Figure 6.6(b). In contrast to Figure 6.6(a) the congestion window continues to grow after the first second in Figure 6.6(b).

Kim *et al.* have shown that the small receive window size of Windows is indeed the bottleneck in WiBro [16]. 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 showed that 64 KB is enough to support 1.5 Mbps in WiMAX [12]. 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 6.7(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.

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 sides. 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 6.7(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.

6.3.2 TCP performances on Auto-tuning

We have shown that TCP with small socket size can not utilize WiBro link. In other words, Windows XP, we have used it for experiments can not support WiBro link. However, Windows Vista and Linux with kernel version of more than 4.0 support auto-tuning of TCP socket buffer size following the link characteristics.

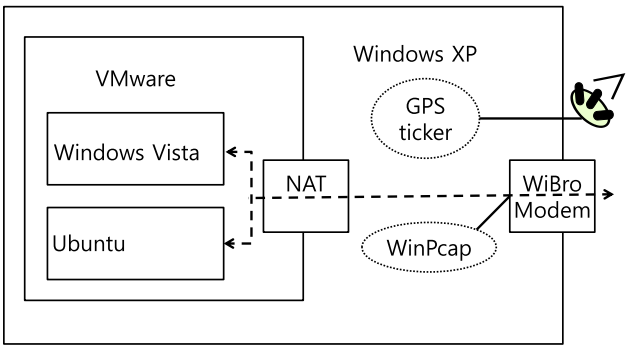
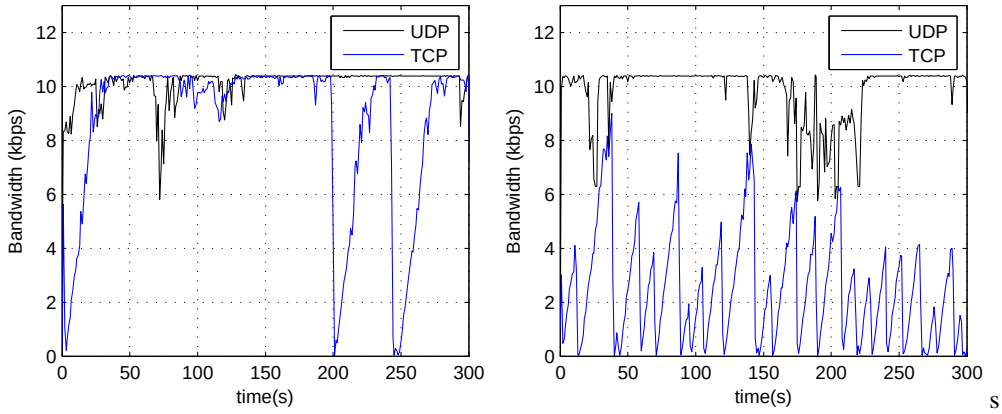


Figure 6.8: Setting for Auto-tuning Experiments (Receiver Side)

The experiments setting for auto-tuning TCP socket buffer is in 6.8. The sender-side setting is same as before. TCP socket buffer size is set as 1MB. In receiver-side, VMware is used to provide the both OS environment, Windows Vista and Ubuntu. NAT between VMware and Windows XP forward packets from and to Operation Systems in VMware. The drivers for WiBro Modem is only worked in Windows. In this way, we can use WiBro Modem for testing TCP over Ubuntu and re-use the experiments settings we set before. The delay in NAT is less than 250us [15] in 100Mbps of traffic load, which is enough small not to affect our experiment result. We capture packets in Windows XP, not in VMware, because we have to use synchronize technique in Windows XP.

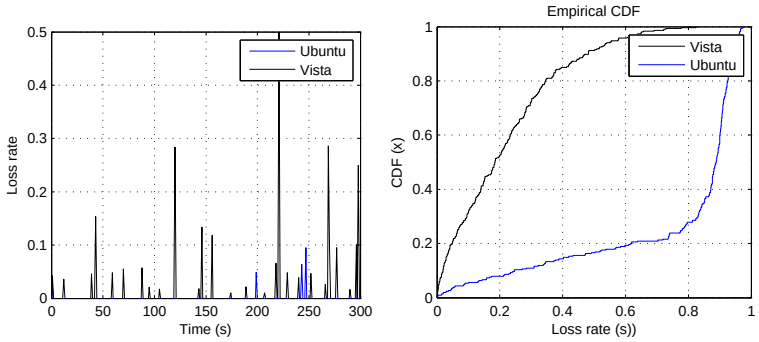


(a) Ubuntu (b) Vista

Figure 6.9: Bandwidth over with autotuned TCP socket buffer

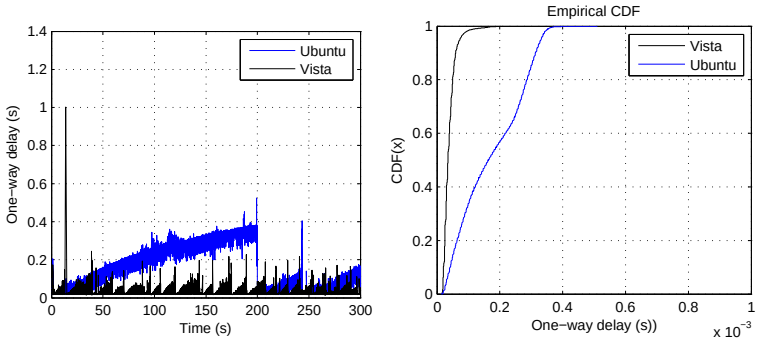
Figure 6.9 shows the bandwidth over time. Both Ubuntu and Vista shows 10.4 Mbps of bandwidth for a UDP flow. It is the same result with the result of experiment using Windows XP. This result give reliability in our experiment setting. The blue line present the bandwidth of TCP traffic. In Ubuntu, the TCP shows almost same bandwidth with UDP, but in Vista, TCP has very fluctuated bandwidth and also shows several timeout. As shown in Figure 6.10, The fluctuated bandwidth is from lost packet with bad signal quality. In both case, the bandwidth is much bigger than TCP's bandwidth in Windows XP, and peak bandwidth is almost similar to UDP's. Therefore, we can conclude that auto-tuning in TCP socket buffer size in Linux kernel and Vista works well for high bandwidth-delay product network like WiBro.

Figure 6.11 shows the delay characteristics of TCP traffic. The delay increase with large bandwidth. As we have explained in 6.3, large bandwidth indicate more queuing. With 10Mbps of bandwidth, the one-way delay is about 400ms which is similar result for experiments in 6.3.



(a) UDP (b) TCP

Figure 6.10: Loss rate with autotuned TCP socket buffer



(a) UDP (b) TCP

Figure 6.11: One-way delay with autotuned TCP socket buffer

In conclusion, TCP over WiBro can not work with small TCP socket size. To utilize TCP throughput over WiBro, large buffer size or the auto-tuning function in Linux kernel or Windows XP is needed. We also check the auto-tuning functions can work well for WiBro. Not even WiBro, the future mobile or wireless network which has long bandwidth-delay product will need same condition to utilize TCP performance.

7. Performances of Applications

I have measured a single flow's performance over commercial WiBro. Next, I measured the application's performance. I choose 3 application mainly used in WiBro network - Web, VoIP and VoD. These experiments are conducted in various location in Seoul, which configured with Wave1.

7.1 VoIP

The most standard way to measure the quality of speech is MOS (Mean Opinion Score). Human experts evaluate the quality of speech score of from 1 to 5. The MOS need human experts and it is time-consuming. The ITU-T E-model provide a computational model for predicting speech quality using input of impairment factor. The output of E-model is R-factor ranging from 0 to 100. We will use E-model because we need only predict the quality of speech transmitted through network and coded by codec.

The R-factor can be calculated following [9]:

$$R = 94.2 - I_d - I_{e-eff} + A, \tag{7.1}$$

where

I_d	: Delay impairment factor
I_{e-eff}	: Equipment impairment factor
A	: Advantage factor

The delay impairment factor I_d is calculated below[11]:

$$I_d = 0.024d + 0.11(d - 177.3)H(d - 177.3) \tag{7.2}$$

In this equation, d is the one-way delay in millisecond and H(x) is the step function.

$$H(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases} \tag{7.3}$$

The equipment impairment factor is following:

$$I_{e-eff} = I_e + (95 - I_e) \cdot \frac{Ppl}{\frac{Ppl}{BurstR} + Bpl} \tag{7.4}$$

I_{e-eff} means impairment factor with considering packet loss rate. Ppl means probability of packet-loss and Bpl means robustness of packet loss. For G729.2 codec, we use 10 for I_e and 18 for Bpl.

The average calculated R-factor for each 5-minute traffic with difference place is plotted in Figure 7.1. To satisfy the quality of call, more than 80 of R-factor is needed and less than 70 of R-factor is dissatisfied quality from impossible to recognize the contents or speechless call. In this graph, most of call quality is better with more than 80 of R-factor even less than 5dB of CINR. However, a traffic near 5dB of CINR, The R-factor value is very low around 50. The time-axis plot of this traffic is in Figure 7.2.

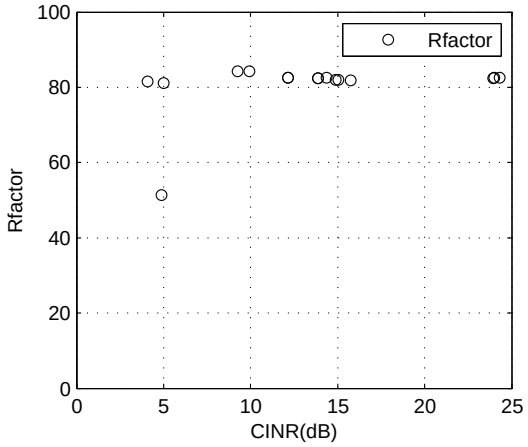
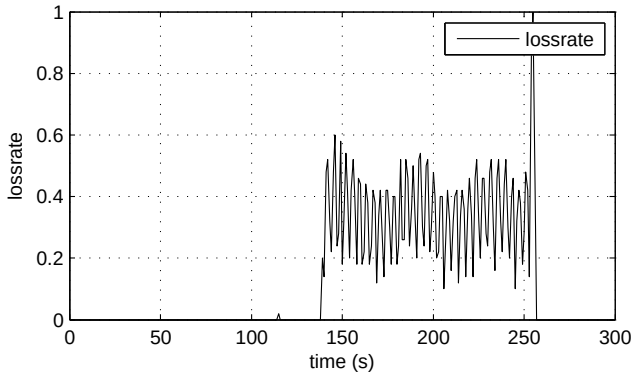
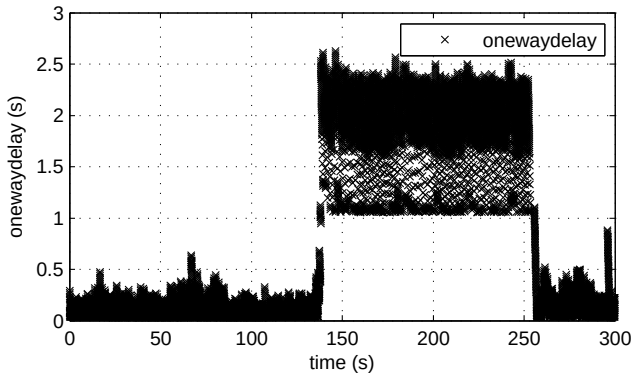


Figure 7.1: Rfactor with different CINR

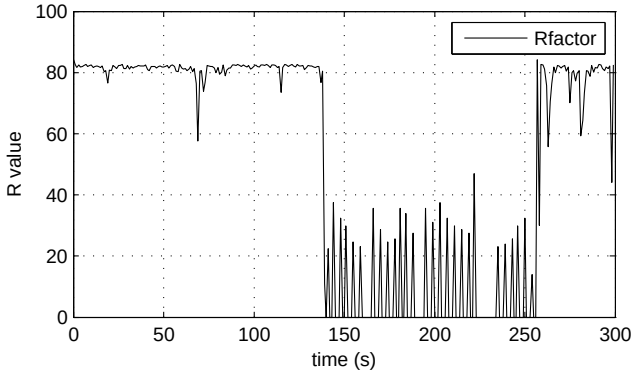
Figure 7.2 shows the loss rate, one-way delay and following R-values of a 5-minute traffic. Around 140 seconds, The one-way delay increase until 2.5 second and loss rate is increase up to 0.4 second urgently. The following R-value is less than 40 and it is very unacceptable quality. This worse quality is result of interfering with other user's flow. Despite VoIP traffic needs only 8kbit/s of bandwidth, it though lots of delay and loss rate. This is because other traffic is injected to the same queue with VoIP traffic. With full queue, The delay is larger and losses are occurred. To assure the quality of real-time application, providing QoS is very important. But that time we conducted experiments, WiBro only supports BE (Best Effort) service.



(a) Lossrate



(b) One-way delay



(c) R-value

Figure 7.2: Quality of VoIP

7.2 Web

Web is widely used application in laptop. A web page include several number of text, image or flash video. To measure the performance of web application, we measure the loading time of mostly visited top 100 web pages in Korea from Ranky [6]. An web page includes many component from external servers not only their own web servers. Our measured loading time is time for loading all component of a web site.

Location	CINR (dB)		
	MIN	MAX	Average
Suseo-starbucks	12	28	22.694743
Apgu-rodeo	12	23	19.349224
Ehwa-minto	10	23	18.735804
Seohyeon-starbucks	-3	13	13.634540
Yangjae-starbucks	-3	24	12.196400
Kyodae-provista	-2	11	4.684927
Hyehwa-starbucks	-3	13	4.118647
Yangjae-sandpresso	-3	9	2.504549

Table 7.1: Experiment Location of Measuring Web Quality

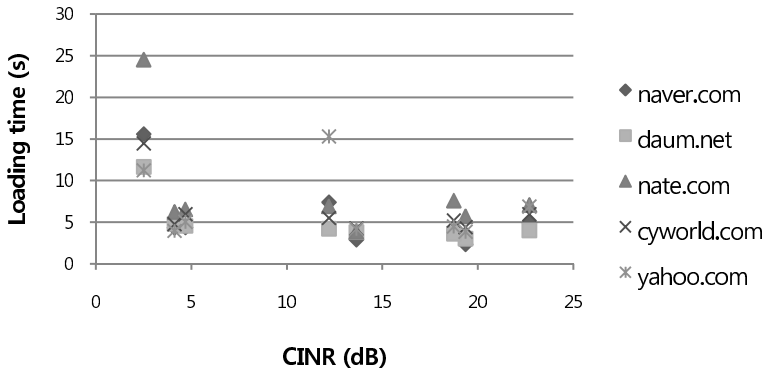


Figure 7.3: Loading time with different CINR

Figure 7.3 shows the loading times of top 5 web pages from Ranky. The top 5 web pages are all portal site. With more than 4dB of CINR value, the loading times are similar around 5 seconds.

With low CINR less than 4dB, the loading time is increasing up to several tens of second. Even the 5 seconds of loading time with good CINR is not good quality for web users in Korea. In Korea Internet environment, normal web page loading time of portal site is between 1 1 and 2 seconds [6]. And people expected less than 3 second of loading time and if the loading time is longer than 4 second, users feel uncomfortable and leave the web page[1].

Figure 7.4 compare the loading times of WiBro with WiFi. Normally, WiBro's loading time

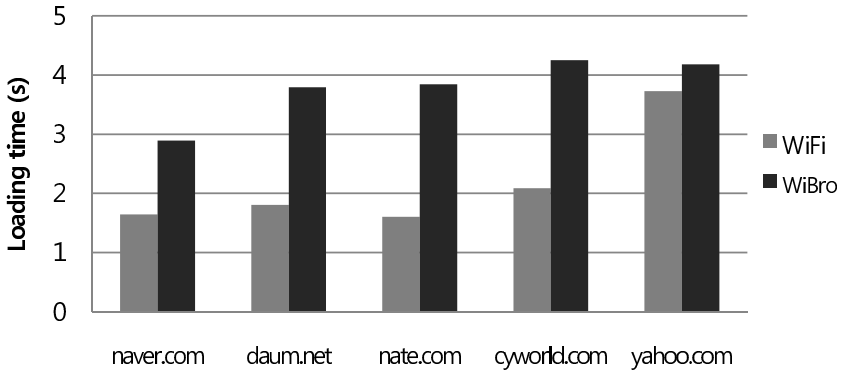


Figure 7.4: Loading time compared with WiFi

Table 7.1 includes experiment location's CINR(dB) information sorted by descending order of average. The CINR seems fluctuated, although the fluctuation is mostly temporal phenomenon. Most of time, the CINR value is near average.

7.3 Flash Video

UCC (User Created Contents) service becomes widely used with trend of Web 2.0. Major web portal service provide an interface for sharing UCC among users. The mostly used standard for transferring moving picture is FLV, a flash video format. Following Adobe.com, Adobe Flash Player is installed on 98% of network connected desktop. Flash video just works with loading the web page on any platform or format and can be played in different way. It is very simple to watch and modify.

A flash video file consist of a header, and interleaved audio, video, metadata tags. Video tags takes the most of the file size. Flv files are transferred on TCP. If there is any missing tags, player stop play until receiving enough number of tags.

Frame Type	Size (Byte)	Play Time (ms)	Received Time (ms)	Buffering Time (ms)
0x1	2437	0	215.735912323	215.735912323
0x2	215	40	215.735912323	175.735912323
0x2	75	80	224.388122559	144.388122559
0x2	73	120	224.388122559	104.388122559
0x2	73	160	224.388122559	64.388122559
0x2	73	200	224.674940109	24.674940109
0x2	73	240	224.674940109	-15.325059891
0x2	73	280	224.674940109	-55.325059891
0x2	73	320	410.731077194	90.731077194
0x2	73	360	410.731077194	50.731077194

Table 7.2: An Example of FLV Video Tag

Table 7.2 shows the example of video tags. Each video tag include the frame type, frame size, play time relative to first frame. In frame type, 0x01 means key frame and 0x02 means inter frame. The ratio of key frame and inter frame is different with the moving picture's characteristics and encoding method.

To measure the quality of flash video, we measure the total buffering time from receiving the first packet of flash video until receiving the last packet. Since FLV doesn't skip any video or audio tag, we didn't consider about the audio and video's quality. Shown as Table 7.2, We also record the each frame's received time relative to first packet's. To play each packet, we have to wait the difference time between received time and play time. We define this difference as buffering time. We only consider the buffering time of video tags, as video tags is the most biggest component of FLV files.

If the time for playing frames is faster than time for receiving frames, the buffering time increase

and playing is interrupted. But if time for receiving frames is faster than time for playing frames, the buffering time will decrease. We define the buffering time of a FLV file playing as the maximum buffering time of video frames. If we wait the buffering time for receiving packets before starting the play, the play will not interrupted. However, we couldn't expect the buffering time before playing.

Video ID	1	4	7	9
Resolution	320*240	320*240	320*240	306*240
Avg.Bitrate (Kbps)	314	246	256	271
Play time (s)	126.000	79.880	85.219	319.187
Contents	2D animation	Game trailer	Home Video	Music Video

Table 7.3: FLV Videos in Experiments

We select 14 videos in Youtube[8]. And selected 4 video's are in Table 7.3. Average bit rate for these video is small enough to being served by WiBro. We downloaded these video's in different location with different CINR levels.

CINR (dB)	Video 1	Video 4	Video 7	Video 9
23	251	1688	620	567
17	216	158	221	5
15	216	159	221	4
6	316	125	2000	17
-2	293	45285	195	10

Table 7.4: Max. buffering time (ms)

Table 7.4 shows the maximum buffering time of each FLV play. Most case, the buffering time is normally less than 1s. This means that if we wait 1s before downloading a FLV file, the FLV play is not distrusted. But sometimes, the buffering times can be increase tens of second. This is from bad signal quality and other flows' effect. In addition, our experiment is conducted on WiBro Wave1 and it has packet overwriting bug we explained in Section [**?**]. We found that lots of packets received in this experiments has this bugs. It is the one reason of big buffering time. However, we conclude that in normal case, flash video need only 1 second for start up delay to seamless play of video files.

8. Performances over WiBro with Mobility

This chapter is best viewed in color.

In this chapter, I focused on the performances over WiBro with mobility environment. WiBro supports up to 120km/h mobility. To support mobility, WiBro system have to support handoff between RASes or ACSes. The performance of mobility depend on the quality of handoff, signal quality of area and the characteristics of wave. I'll show the service area's signal quality distribution and compare the performances between with and without mobility.

8.1 Distribution of Signal Quality

Figure 8.1 shows the experiment environment. The experiment is conducted in North side of Seoul, above Han river. The course is about 15km and takes 10 $\sim$ 15 minutes by car. The speed of car is from 0km/h $\sim$ 45km/h Our experiment is conducted from early morning, 12am to 6am, to minimize the other user's interrupt and to avoid traffic jam. The large 'X' marks show the handoff between ACR and small 'x' marks shows the handoff between RAS. In this course, there are 3 handoff between ACR and 4 handoff between RAS.

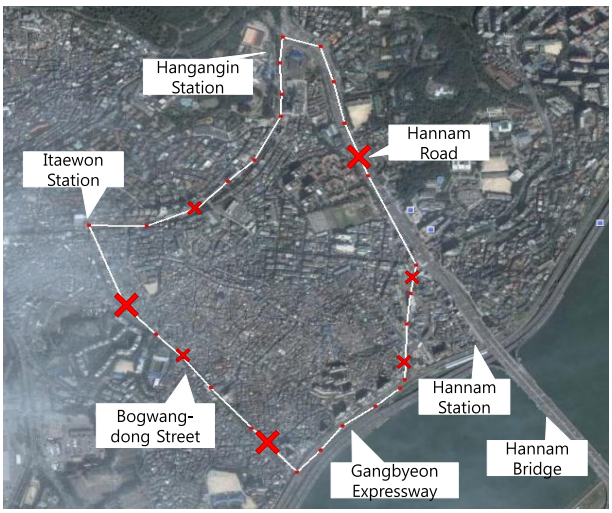
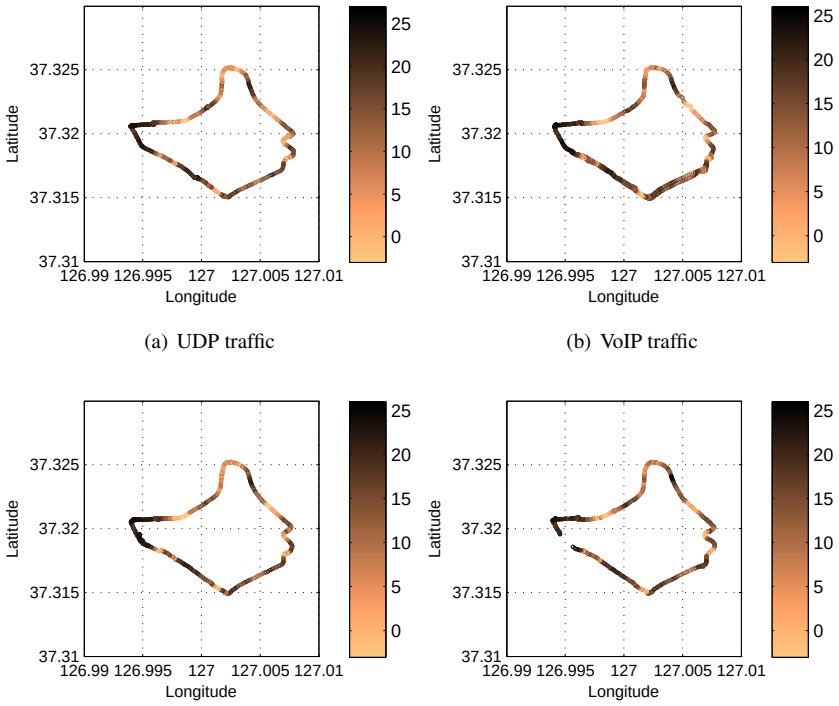


Figure 8.1: Mobility Map



(a) UDP traffic

(b) VoIP traffic

(c) TCP traffic (down)

(d) TCP traffic (up)

Figure 8.2: CINR variation

The other experiments setting is same as before. We use Iperf to generate traffic and synchronization techniques we made. The only difference is mobility. Our client labtop is in moving vehicular. In this experiments, we test only UDP, TCP and VoIP traffic.

Figure 8.2 shows the CINR values in each cycle. The dark color indicate good signal quality and vice versa. The distribution of CINR value is same in every cycle. And we can see that location with weaker color is same where handoff is occurred. In handoff area, CINR value is lower than 5. This graph shows the allocation of ACR, RAS and repeater by KT, the WiBro service company. They set the repeater for quality of CINR more than 5dB.

Figure 8.3 shows the cdf of CINR distribution. We have shown that WiBro shows more than 2Mbps with more than 10dB of CINR in Wave 1. In our experiment location, 60% $\sim$ 70% of street has more than 10dB of CINR. It means that 60% of area assure more than 2Mbps of bandwidth in WiBro Wave 1. 20% of area has less than 5dB CINR value. With this CINR value, the bandwidth is less than 1Mbps.

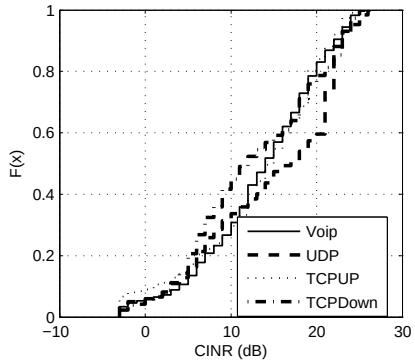


Figure 8.3: CINR variation (CDF)

8.2 Quality of Traffic over WiBro

In this section, we will show the quality of traffic in mobility environment. Each experiment is conducted in unit of one trip of the course. At each cycle, one kind of traffic is tested. We examine the udp and tcp's quality over downlink and uplink. Also, VoIP traffic's quality over downlink is provided. All traffic's quality is plotted by time.

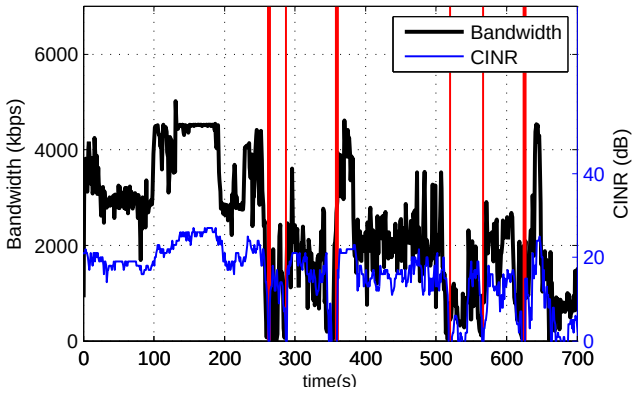


Figure 8.4: UDP bandwidth (downlink)

At first, bandwidth of UDP for downlink and uplink is in Figure 8.4 and 8.5. The thick black line indicates the bandwidth and blue line indicates the CINR. Compare the bandwidth of this traffic with the stationary experiment result from section 5.1, the peak bandwidth with good signal quality is same, about 5Mbps, 2Mbps each. This graph shows that the bandwidth is in proportion to

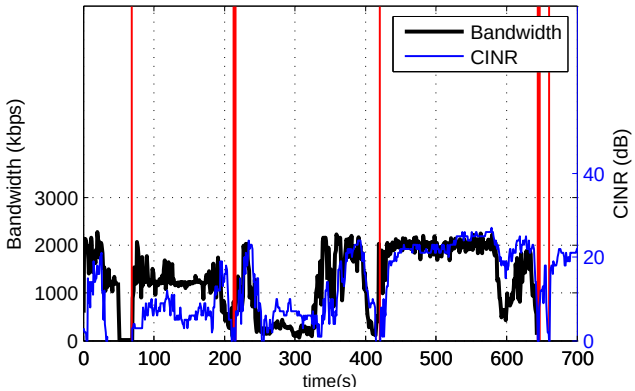


Figure 8.5: UDP bandwidth (uplink)

CINR values. The most notable thing is that the bandwidth become almost zero when handoff is occur. Handoff not only between ACR, but also between RAS makes bandwidth very low. The low bandwidth is from overhead of handoff and also from low signal quality.

Bandwidth of TCP for downlink and uplink is in Figure 8.6 and 8.7. Since our experiment is conducted based on Windows XP, the bandwidth of TCP also can not exceed 1Mbps as we explained in Section 5.1. TCP's Bandwidth is also has similar tendency as UDP'. When handoff is occurred, Signal condition is very low and TCP's bandwidth fall into below.v

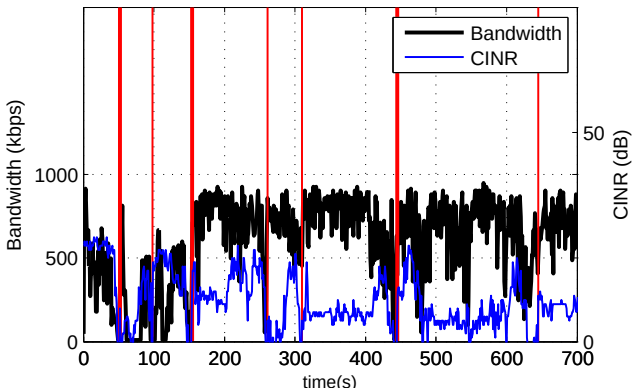


Figure 8.6: TCP bandwidth (downlink)

The interesting thing in above 4 graphs for udp and tcp's bandwidth is that bandwidth is always fall down whether ACR or RAS handoff. The handoff delay in WiBro is less than 150ms. The

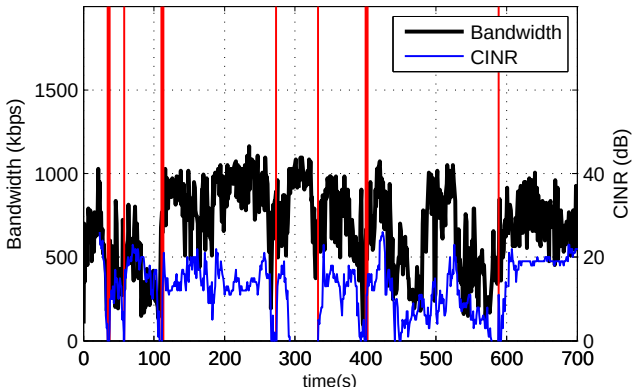


Figure 8.7: TCP bandwidth (uplink)

bandwidth plotted in graph is averaged every minute. The bandwidth is starting drop before handoff. Therefore, the low bandwidth with handoff is affected not only by handoff overhead, but also by bad signal condition.

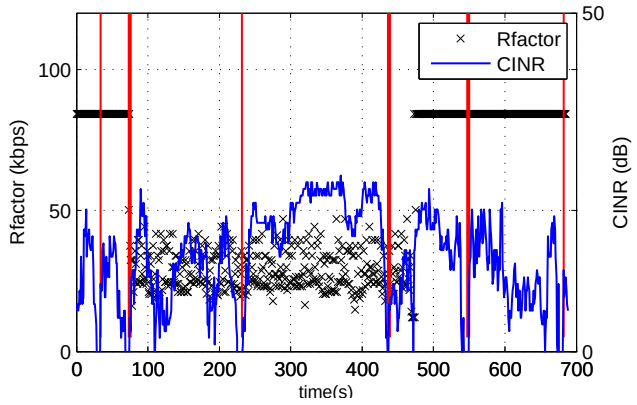


Figure 8.8: VoIP Rfactor

Figure 8.8 shows the R-factor for VoIP traffic. The first and last part of trace has good R-value of 84.2 regardless of signal quality. VoIP traffic need very small bandwidth, so it can support good quality even when signal quality is bad. However, the middle of trace has very low R-value even the signal quality is good. The reason for low quality is long delay. The middle of trace has one-way delay longer than 1s. Although VoIP need very small bandwidth, it is very sensitive to one-way delay as it is real-time application.

WiBro currently support only BE service. If one user send lots of packet at once, the other user's delay who sharing queue also increase. This make about 400s of very bad quality VoIP service. To support real-time application's quality, minimizing delay and supporting another kinds of QoS is crucial.

9. Conclusion

WiBro is high-bandwidth mobile wireless data services which compatible with 802.16 standard. WiBro is developed within Korean indoor technology by ETRI and Samsung. Since 2006, KT have started the world's first commercial WiBro services in Seoul, the capital of Korea and it's satellite cities. The United State, Japan, East Europe is developing and preparing commercial WiBro services. Therefore, I study the characteristics of WiBro compared with 3G networks and WiFi and measure the performance of WiBro user's experience.

At first, I participated developing a small and cheap GPS synchronization module to synchronize two end node. We could get a synchronization techniques of sub-millisecond accuracy.

Second, I measured the best-case performance of single flow in ideal WiBro Wave2 environment. I used KT's testbed to assure best signal strength and no competition and interaction from other flows. WiBro has high bandwidth-delay product compared to Ethernet, WiFi and cellular network. The default socket buffer size of Windows XP, 64KB, is not enough to support TCP over WiBro. Therefore, the network which has high bandwidth-delay product have to configure the TCP setting for Windows XP users.

Third, I measured the WiBro user's experience on WiBro Wave1 in Seoul and its satellite cities. I examined the relationship of CINR between bandwidth and delay using UDP and TCP traffic. The results shows that more than 60% of area has more than 10dB CINR. In this condition, both WiBro uplink and downlik can support 2Mbps of bandwidth. I also measure the service quality of the most frequently used application such as Web, VoIP and VoD. At last, I measure the performance of UDP and TCP traffic and service quality of VoIP in vehicular mobile environment. These results proof the high quality of WiBro service.

요 약 문

상용 와이브로 망에서 UDP/TCP 플로우와 어플리케이션의 성능 측정

와이브로는 802.16 와 상호 호환성이 있으며, ETRI와 삼성에 의해서 순수 국내 기술로 만들어진 무선광대역인터넷이다. 2006년 KT가 서울 및 수도권에 세계 최초의 상용 와이브로 서비스를 시작한 이래, 미국, 일본, 동유럽 등의 국가에서도 모바일 와이맥스 서비스의 개발 및 상용화를 진행하고 있다. 이에 기존의 3세대 이동통신 및 와이파이와는 다른 와이브로의 특징을 연구하고, 실제로 사용자들이 체감 성능을 측정하여보았다.

첫째로, 실험을 위해, 두 단말을 시간동기화시킬 필요성을 느끼고 실험에 사용하기 위한 작은 GPS동기화 모듈의 개발에 참여하였고, 결과적으로 서브밀리세컨드 이하의 정확도를 가지는 동기화기술을 얻을 수 있었다.

둘째로, KT에서 Wave2 와이브로 테스트베드를 지원받아서 플로우간의 경쟁 및 간섭이 없고, 가장 좋은 신호 세기를 가지는 이상적인 환경에서의 기저성능을 측정해보았다. 와이브로는 랜, 와이파이 및 이동통신 서비스에 비해서 높은 대역폭-지연 곱을 가지고 있기 때문에 기존의 Windows XP에서 기본적으로 제공하는 64KB TCP 소켓 크기로는 충분한 TCP성능을 얻다. 따라서, 고 대역폭-지연 곱을 가진 네트워크에서는 Windows XP 사용자들을 위해 모뎀드라이버 설치시 TCP 환경설정을 해줘야 한다는 점을 시사하였다.

셋째로, 서울 및 수도권지역의 상용 Wave1 와이브로 서비스에서 사용자들의 체감성능을 측정해보았다. 신호대잡음비와 대역폭 및 지연의 관계를 UDP와 TCP 트래픽을 사용해서 조사해본 결과 와이브로 서비스 지역의 60% 이상의 지역이 10dB이상의 신호대잡음비를 가지고 있었고, 이 조건하에서는 와이브로의 성능을 업, 다운링크 모두 2Mbps 이상의 대역폭을 보여주었다. 상용 와이브로 서비스 위에서 사용자들이 많이 사용하는 어플리케이션인 Web, VoIP 및 VoD의 사용자들의 체감 성능을 조사하였고, 특히 와이브로의 세일즈포인트인 이동성 환경에서 실제 성능을 평가하였다.

References

[1] 4s. www.akamai.com/4seconds.

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

[3] Kreonet. http://www.kreonet.re.kr/english/.

[4] kreonet-kt-ix-day. http://noc.kreonet.net/sub02/s03_2.htm.

[5] Ping. http://ftp.arl.mil/ mike/ping.html.

[6] Rankey. http://www.rankey.com/.

[7] Tracert. http://www.tracert.com/.

[8] Youtube. http://www.youtube.com/.

[9] ITU-T standard g.107. the e-model, a computational model for use in transmission planning, May 2000.

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

[11] R. G. Cole and J. H. Rosenbluth. Voice over ip performance monitoring. *SIGCOMM Comput. Commun. Rev.*, 31(2):9–24, April 2001.

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

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

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

[15] G. M. V. Keon Jang, Sangman Kim and S. Moon. Implementation and evaluation of a mobile planetlab node. In *SOSP 2009 Workshop on Real Overlays and Distribute Systems (ROADS '09)*.

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

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

[18] C. Liao, M. Martonosi, and D. W. Clark. Experience with an adaptive globally-synchronizing clock algorithm. In *SPAA '99: Proceedings of the eleventh annual ACM symposium on Parallel algorithms and architectures*.

[19] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39:1482–1493, 1991.

[20] D. L. Mills. RFC2030: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, 1996.

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

[22] 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*.

[23] W. Wei, C. Zhang, H. Zang, J. Kurose, and D. Towsley. Inference and evaluation of split-connection approaches in cellular data networks. In *Passive and Active Measurement Conference*, Adelaide, Australia, March 2006.

감 사 의 글

쌩유

이 력 서

이　　름 : 우 신 애

생 년 월 일 : 1986년 3월 1일

출 생 지 : 서울특별시 구로구 번동 ..

본 적 지 : 서울특별시 구로구 번동 ..

주　　소 : 대전 유성구 구성동 한국과학기술원 다솜관 501호 ...

E-mail 주 소 : shinae@an.kaist.ac.kr

학　　력

2002. 3. – 2004. 2.　대전과학고등학교 (2년 수료)

2004. 3. – 2008. 2.　한국과학기술원 전산학과 (B.S.)

2008. 3. – 2010. 2.　한국과학기술원 전산학과 (M.S.)

경　　력

2007. 7. – 2007. 8.　한국과학기술정보연구원 (KISTI) 인턴

학 회 활 동

1. Keon Jang, **Shinae Woo**, Sue Moon, *Design Considerations for a Mobile Testbed*, In Proceedings of the International Conference on Future Internet Technologies (CFI), June 2008, Seoul, Korea.
2. **Shinae Woo**, Keon Jang, Sangman Kim, Soohyun Cho, Jaehwa Lee, Youngseok Lee, and Sue Moon, ACM Workshop on Mobile Internet through Cellular Networks: Operations, Challenges, and Solutions (MICNET), October 2009, Beijing, China.