TCP over WiMAX: A Measurement Study

Emir Halepovic, Qian Wu, Carey Williamson, Majid Ghaderi
Department of Computer Science
University of Calgary, Canada
{emirh, qianwu, carey, mghaderi}@cpsc.ucalgary.ca

Abstract

We present active measurement results from a commercial IEEE 802.16/WiMAX-based network, with primary focus on TCP performance. We compare four TCP variants, namely New Reno, Cubic, Vegas and Veno, using throughput, round-trip time (RTT), and retransmission rate metrics. While all TCP variants achieve similar throughput, they do so in different ways, with different impacts on the network performance. We identify adverse effects of TCP window auto-tuning in this environment and demonstrate that on the downlink, congestion losses dominate wireless transmission errors. We reveal several issues for this WiMAX-based network, including limited bandwidth for TCP, high RTT and jitter, and unfairness during bidirectional transfers. Such a network environment may be challenging for many wireless Internet applications, such as remote login, VoIP, and video streaming.

1. Introduction

Wireless Internet access is an essential requirement for today's tech-savvy users, including students, consumers, travelers, and business users. However, users often have to make the choice between fixed high-speed local access (e.g., WiFi networks), or low-speed mobile wide-area access (e.g., cellular data networks). There is a strong desire to bridge this gap and offer high-speed mobile access, with performance comparable to wired broadband.

One promising technology is based on the IEEE 802.16 family of standards, commonly referred to as WiMAX (Worldwide Interoperability for Microwave Access). While WiMAX is commonly viewed as a metropolitan-area technology (WMAN), it can offer a service range of up to 50 km, under ideal conditions. Currently, the major market area of WiMAX is fixed Internet access for residential and small business users, as an alternative to cable and DSL service.

The purpose of our study is to investigate the performance characteristics of WiMAX-based networks, and to assess their potential impact on the user-perceived experience with Internet applications. This paper focuses on transport-layer behavior, specifically the Transmission Control Protocol (TCP). To the best of our knowledge, this is one of the first detailed studies of TCP performance on a commercial WiMAX network.

The first goal of our study is to characterize a WiMAX network and identify any important effects of the new technology on performance. In the particular network that we study, the network path has sub-optimal routing, with high round-trip time (RTT), and high RTT variability, which may adversely affect interactive Internet applications such as Web browsing, as well as delay-sensitive applications such as media streaming and VoIP. The available bandwidth in the network is limited by the vendor's configuration of the wireless modem, and the achievable throughput varies, depending on the deployment location.

The second goal is to compare four TCP variants on the WiMAX network. While all TCP variants tested can achieve similar throughput, they do so in different ways, with different impacts on the network performance. Our measurement results demonstrate that TCP segment retransmissions on the downlink are mainly triggered by congestion losses induced by TCP, rather than transmission losses due to wireless channel errors. The Automatic Repeat Request (ARQ) mechanism employed by WiMAX successfully shields TCP from transmission losses, allowing all TCP variants to function reasonably well. However, we find that bidirectional TCP transfers degrade network performance, which may be an obstacle for users running file-sharing or video conferencing applications that require bidirectional transfers.

The rest of the paper is organized as follows. Section 2 provides background on WiMAX and TCP, and reviews related work. Experimental methodology and network characterization are discussed in Section 3. Section 4 presents the measurement results and analysis for unidirectional transfers. Additional experiments and results are discussed in Section 5. Conclusions and future research directions are given in Section 6.

978-1-4244-2818-2/08/$25.00 ©2008 IEEE

2. Background and related work

2.1. WiMAX

IEEE 802.16 is a family of standards for local and metropolitan area networks that define physical (PHY) and medium access control (MAC) layers of the air interface. The fixed and mobile components of the standard have recently been consolidated under the 802.16e™-2005 document [6].

The WiMAX MAC layer supports point-to-multipoint (PMP) and mesh topologies, both of which contain elements for efficiently dealing with shared access medium. In PMP topology, a WiMAX network is divided into cells and sectors consisting of one base station (BS) and many subscriber stations (SS), similar to a cellular telephone network. This architecture naturally lends itself to PMP operation in the downlink direction, from BS to SS, where time-division duplex (TDD) or frequency-division duplex (FDD) is used. In practice, TDD is typically used, where BS dynamically adjusts the duration of the downlink and uplink portions of the data frame, depending on the requirements. Uplink access is usually TDMA, with scheduling fully controlled by the BS.

The 802.16 standard explicitly supports Quality of Service (QoS) differentiation. SS's are assigned access slots on a demand basis. An SS may request access continuously or upon user request, depending on the class of service. QoS policy implementation is not specified in the standard, but rather left for vendors to decide.

The MAC layer is connection-oriented. All service flows are mapped to connections between BS and SS, with one MAC connection per service flow fashion. For example, one TCP connection would be mapped to two MAC connections, one for each direction of the TCP transfer. MAC connections map the requirements of service flows to bandwidth requests on the uplink. Each MAC connection typically uses a timer-based ARQ mechanism.

Adaptive modulation is used to achieve the highest possible data rate for a given link quality. Modulation can be adjusted at very short time intervals (within each frame that is typically 5 ms long), allowing for robust transmission links and high system capacity.

The OFDM mechanism uses many individual carriers to transmit user data, and effectively adapts to multi-path fading in the wireless environment. Combined with TDD and adaptive modulation, it enables high channel utilization, capacity, and data rate when deployed in a non-line-of-sight (NLOS) environment, such as urban areas. Depending on the frequency range and modulation used, WiMAX can theoretically achieve a data rate of over 120 Mbps, and 50 km in range.

While often seen as an evolutionary extension of WiFi, WiMAX has several important differences. Wireless channel access is controlled by the BS in PMP mode, in contrast to WiFi where the access point contends with mobile nodes for channel access. WiMAX is intended for infrastructure deployment as a long-range access technology rather than for short-range home and office networking. Licensed spectrum is predominantly used for WiMAX, and QoS is explicitly supported, unlike in WiFi.

2.2. TCP

TCP is a reliable, connection-oriented data transfer protocol, with well-established mechanisms for flow and congestion control based on the sliding window. In this study, we use TCP variants that include only sender-side modifications. This section outlines the basic operation of four TCP variants, and justifies their inclusion in this study. Interested readers are referred to the original literature for more details.

TCP New Reno is a well-studied and prevalent TCP variant in the Internet [3]. Its main components are the slow start algorithm and congestion avoidance phase. Slow start allows the sender to grow its window rapidly to capture bandwidth. The congestion avoidance phase following the slow start grows the congestion window linearly, promoting fairness and stability. Loss is detected either by a triple duplicate acknowledgment (ACK), in which case the congestion window is halved, or a retransmission timeout (RTO), in which case it is reduced to one segment. The congestion window evolves in Additive Increase Multiplicative Decrease (AIMD) fashion, which is appropriate for congestion-induced loss, but not for random loss due to wireless transmission errors. TCP New Reno is evaluated in this study as a baseline TCP variant.

Cubic is a TCP variant for networks with high bandwidth-delay product (BDP). It is characterized by an aggressive (cubic) window growth function that is independent of RTT [16]. Upon loss, it reduces the window by a factor of 0.8. Cubic is the default TCP variant in newer Linux kernels, and therefore makes a good choice for this study.

TCP Vegas is a modified New Reno that detects loss proactively earlier; it can retransmit before receiving the third duplicate ACK or RTO timer expires [2]. The congestion avoidance mechanism keeps the appropriate amount of data in the network, and congestion window increases and decreases in a linear fashion. Congestion window stability and low retransmission rate make Vegas a good choice for wireless networks.

TCP Veno specifically targets wireless networks, by combining Reno and Vegas mechanisms to deduce whether a network is congested or random loss is more

likely [4]. Congestion window evolution is determined by throughput estimates, and reduction factors are 0.5 for congestion loss and 0.8 for random loss. Veno's efficient bandwidth utilization and low retransmission rate make it attractive for wireless links.

Regardless of the TCP variant employed, a recently introduced improvement for dealing with high BDP links is **TCP window auto-tuning** [18]. Auto-tuning is the default Linux kernel feature that dynamically adjusts the sender's and receiver's window to better utilize available link capacity. TCP window adjustment is accomplished by resizing the socket buffer size. The kernel monitors TCP throughput and when it estimates that more bandwidth is available than the default window sizes are able to use, it then increases the default window size, which is then advertised to the other endpoint of the connection. This mechanism is effective for high bandwidth links, since it reduces the time required for the TCP sender to utilize the available link capacity.

2.3. Related work

Wireless links are inherently lossy and subject to errors due to path loss, interference, channel fading, and handoffs. TCP was originally designed for wired links with low packet error rate (PER), where packet loss happens predominantly due to congestion. Wireless links pose a challenge to TCP, which cannot distinguish between different causes of loss. This results in lower throughput, and underutilization of a wireless link [17].

Several approaches have been proposed as a remedy for wireless TCP problems, including end-to-end and link-layer solutions, as well as connection splitting [1, 17]. While the proposed solutions attempt to address the fundamental issue of the high bit-error rate (BER), the heterogeneity of wireless networks may still require different mechanisms for each type of network. Not all wireless environments are equal; therefore, TCP's performance problems may be more acute in some types of wireless networks than in others.

In *cellular networks*, the important factors affecting TCP performance are handoffs, temporary signal interruptions by obstacles, high delay, and delay variance [10]. *WiFi networks* based on IEEE 802.11 standards, with high data rates and low mobility, cause problems for TCP through contention-based access, hidden and exposed nodes, and dynamic rate-adaptation [5, 17].

WiMAX networks are a new kind of wireless network. Often, they are evaluated using analysis and simulation of PHY and MAC layers [15, 19, 20], since live measurements are lacking or preliminary [14], mostly due to sporadic deployment. The reported analytical and simulation results indicate that improvements in TCP performance would result from different modulation schemes between data and ACK channels, proper ARQ settings, a TCP-friendly MAC scheduling policy, and appropriately tuned ACK handling [15, 20].

The fact that different wireless networks require different mechanisms for optimizing TCP performance provides motivation for our work. It is important to characterize WiMAX-based networks in order to understand their behavior and their impact on TCP. Furthermore, we need to understand which parameters affect the TCP performance, and why.

3. Experimental methodology

3.1. Network environment

The network under consideration is an early commercial offering of a fixed WiMAX service deployed across Canada by two network providers. The indoor wireless modem we use is Motorola Expedience RSU-2510F, operating inside the licensed 2496-2690 MHz band. Expedience technology uses TDD/OFDM combination with 4/16/64 QAM modulations and 3 - 6 MHz channels. The modem connects via Ethernet to the user computer. The MAC layer protocol and scheduling policy are not publicly available. Therefore, we treat the wireless modem as a black box.

The service provider limits the maximum data rates to 1.5 Mbps on the downlink and 256 Kbps on the uplink. Nomadic movement between base stations is fully supported, while mobility during a session is not (though it is still possible at low speeds).

The experimental test-bed consists of two commodity laptops, one connected to the wireless modem and another to the University of Calgary campus network using a 100 Mbps Ethernet LAN (Figure 1). Two different test locations for the wireless end of the test-bed are used; one on campus and one in a residential setting.

Both laptops are running Linux with kernel version 2.6.20, which supports the pluggable on-demand TCP congestion control algorithms used in this study. This kernel's default TCP settings are Cubic variant with auto-tuning, timestamp option, and SACK enabled.

We use *netperf* [12] for TCP transfers and *iperf* [12] for UDP streams. We collect the traffic traces using *tcpdump* [12] and post-process them with *tcptrace* [12], *tstat* [12], and other custom tools.

Measurements were made between April 2007 and March 2008. The length of the study allowed us to observe long-term changes in WiMAX network behavior.

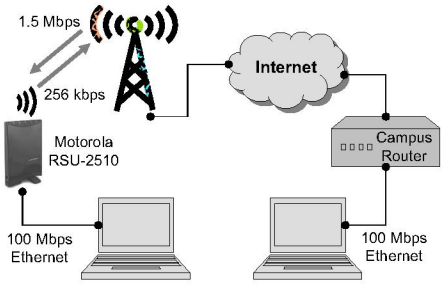

Figure 1: Experimental setup

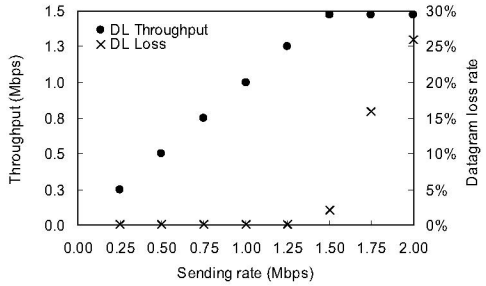

Figure 2: Throughput of UDP CBR transfers

3.2. Network characterization

This experimental study measures performance of the network as seen by the user equipped with the WiMAX modem. Since it was not possible to measure the WiMAX link in isolation, we conduct network characterization measurements to understand the effects of the WiMAX link on the overall performance and to better interpret the TCP-related measurements.

To measure available bandwidth, RTT, and loss on the path, we use the *pchar* tool [12] with UDP probes. The path measurements are summarized in Table 1.

Table 1: Network path characteristics

Measurement	Downlink		Uplink	
Date	Jun 8, 2007	Jun 26, 2007	Jul 4, 2007	Feb 15, 2008
Hops	16	16	13	14
μRTT (ms)	118.2	125.0	124.4	107.3
σRTT (ms)	0.9	3.2	6.4	3.1
Wired μRTT (ms)	88.2	88.1	65.3	75.0
Wired σRTT (ms)	0.07	0.12	n/a	n/a
Wireless μRTT (ms)	30.0	37.0	59.1	32.3
Wireless σRTT (ms)	n/a	n/a	30.0	3.8
Bottleneck BW (kbps)	385	497	87.5	162.8
Packet loss	9%	11%	1%	8%

The network path between the test endpoints consists of 13 to 16 hops, some of which change over time, which is expected in today's Internet. The overall mean RTT (μ_{RTT}) is very high, consistently exceeding 100 ms. Closer analysis reveals that the major contributors to the RTT are cross-continental hops on the backbone network from Calgary to Toronto (about 3,000 km) and back. In the downlink direction, the total μ_{RTT} of 118.2 ms is composed of 88.2 ms on the wired and 30.0 ms on the wireless portion of the path. It is unusual that the wired portion accounts for almost 75% of the total RTT. This shows inefficient routing, with all traffic routed through Toronto. High delay can affect latency-sensitive applications such as telnet and VoIP. Performance will differ depending on the geographic location of users.

We next consider the standard deviation (σ_{RTT}) of RTT. The results in Table 1 show the μ_{RTT} and σ_{RTT} of the wired and wireless hops, as computed by *pchar*. Wireless portion consists of a single wireless hop over the WiMAX link. For the downlink, σ_{RTT} for the total path exceeds that of the wired portion by an order of magnitude. This indicates higher RTT variations on the wireless hop. For the uplink, the measured σ_{RTT} for the wireless hop is again larger than the overall σ_{RTT}, sometimes by an order of magnitude. It is reasonable to expect higher RTT variation on the wireless hop due to higher PER and the ARQ employed by WiMAX [20]. Note that *pchar* reports σ_{RTT} only for the partial path and not for individual hops, hence several entries in Table 1 are marked "n/a".

3.3. MAC layer characteristics

Before addressing TCP behavior over WiMAX, we first conduct several experiments with UDP constant-bit-rate (CBR) streams, to gain basic insights into the wireless scheduler behavior without TCP's flow and congestion control effects. We show downlink throughput and loss in Figure 2. We note that throughput closely follows the sending rate with minimal loss until the sending rate reaches the link limit of 1.5 Mbps with a 2.6% datagram loss rate. Further increasing the sending rate causes the throughput to saturate the link capacity, with loss rates increasing. The uplink throughput results are similar. The only difference is that more losses occur; these reach 4.9% when the sending rate equals the link capacity.

From these results we can conclude that the scheduler allocates as much bandwidth as the data backlog in the buffers requires up to the link capacity. Trying to send more data than the available capacity, even up to 4 Mbps, does not appear to degrade the throughput or penalize the sender in any way, unlike the phenomenon observed in CDMA networks [11].

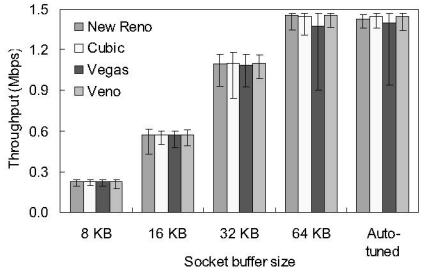

Figure 3: Downlink throughput

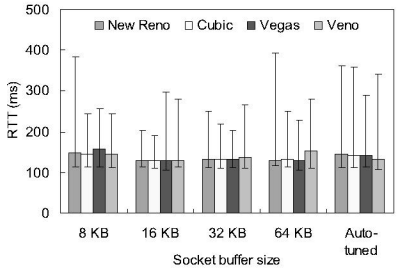

Figure 4: Downlink RTT

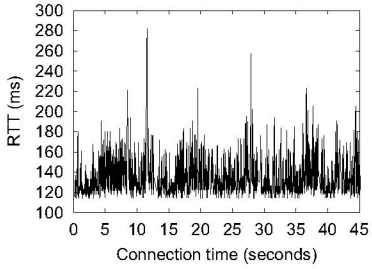

Figure 5: RTT variability

4. Results for unidirectional transfers

4.1. Downlink performance

Throughput. We compare throughput and other metrics with respect to socket buffer size and TCP variant. Each experiment consists of a series of 1-minute data transfers. Sender's and receiver's socket buffer sizes, which set the limit on the TCP congestion window, are set to 8, 16, 32 and 64 KB. Auto-tuned case is considered the largest size because it initially uses receiver's default advertised window of 85.33 KB. Downlink experiments are repeated five times per day, for 7 days.

Throughput is reported based on *netperf* output, which measures goodput, defined as user data successfully transferred per unit time. Figure 3 shows the mean and range of throughputs measured as the socket buffer size grows from 8 KB to auto-tuned. Throughput of all TCP variants is window-limited when using smaller buffer sizes from 8 to 32 KB. Larger buffer sizes clearly benefit each TCP variant and allow them to utilize the full link capacity of 1.5 Mbps. In this case, throughput is loss-limited.

All TCP variants achieve similar throughput for each of the buffer sizes tested, though Vegas has slightly lower throughput (about 96% of that achieved by the other TCP variants). For the two largest buffer sizes, Vegas' throughput ranges widely from 0.9 to 1.5 Mbps. About 20% of Vegas flows achieve no more than 1.3 Mbps. The trace analysis confirms that the conservative window growth by Vegas causes lower throughput, as expected.

Overall, it appears that, even though very different in their congestion control mechanisms, all four TCP variants utilize the available bandwidth equally well.

RTT. The second metric of interest is RTT. Figure 4 shows the mean and range of RTT values of one representative test connection for each socket buffer size and TCP variant. All TCP variants experience similar mean RTT values for each buffer size. However, RTT jitter is very high. Figure 5 shows that RTT is highly variable throughout the connection. The minimum RTT is consistently around 120 ms. This distinctive lower bound is due to the high path propagation delay.

Retransmissions. After investigating TCP throughput and RTT, we turn to segment retransmission rate, expressed as the percentage of retransmitted segments during each connection.

We find some significant differences between TCP variants. Figure 6 shows the average retransmission rates corresponding to each TCP variant and socket buffer size. The overall trend is that larger buffer sizes result in higher retransmission rates. This is not surprising, given that 64 KB and larger buffers produce congestion windows exceeding the BDP, leading us to conclude that most retransmissions are due to congestion losses.

The auto-tuned case bears more consideration. There is a clear jump in retransmission rates, which approach 3% for Cubic. Retransmissions for New Reno, Cubic, and Veno more than doubled with auto-tuning compared to 64 KB buffer, with Cubic suffering the four-fold jump. Vegas is least affected by large buffer sizes and auto-tuning.

4.2. Uplink performance

We present uplink measurements of throughput, RTT and retransmission rate for 16 experiments performed over an 8-day period, twice per day. We use buffer sizes of 8, 16 and 32 KB, as well as auto-tuned. The 32 KB buffer size appears adequate to utilize the uplink capacity fully.

Throughput. In Figure 7, we show mean throughput and range for all TCP variants for different buffer sizes. The trends are similar to downlink results. As buffer size increases, throughput increases. The differences between TCP variants are more pronounced for uplink than downlink flows. TCP Veno achieves the highest throughput for 32 KB and auto-tuned buffer, with low variability. This is one indication that Veno performs well in a bandwidth constrained environment.

RTT. The mean and range of RTT values from one experiment are plotted in Figure 8. We observe that there are both similarities and dissimilarities with downlink results. Most importantly, uplink RTT is much higher, nearly double on average. Mean RTT is similar across buffer sizes and lies in the 200 to 300 ms range. This very

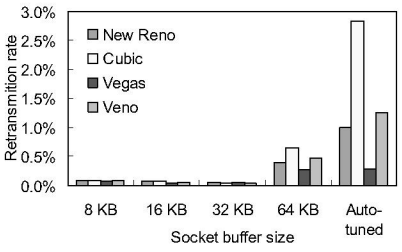

Figure 6: Downlink retransmissions

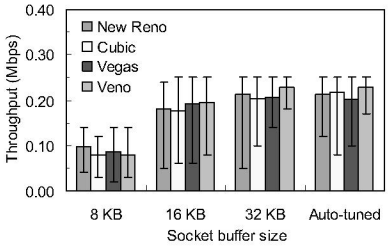

Figure 7: Uplink throughput

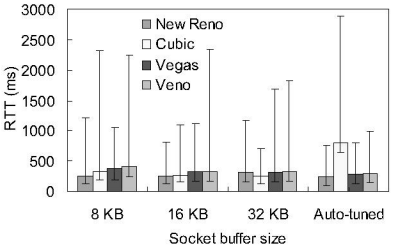

Figure 8: Uplink RTT

high RTT poses potential performance problems for delay-sensitive applications, such as telnet and VoIP. We further note that minimum RTT is mostly stable around 200 ms. However, maximum RTT is excessively high, reaching nearly 10 times the minimum. Therefore, higher delay spikes are to be expected on uplink. We further notice some oddities, such as unusually high RTT range for smaller buffer sizes, but also extremely high RTT for a combination of Cubic and auto-tuning.

Retransmissions. Compared to the downlink, retransmission rates are much higher on the uplink (Figure 9). Average retransmission rate increases with buffer size for New Reno and Cubic, but not for Vegas, which is qualitatively the same behavior as on the downlink. We find evidence of random wireless losses in uplink traces that can be attributed to lower transmission power of the modem, uplink scheduling mechanism, and ambient interference. Veno's adaptation to wireless links explains its improved uplink loss performance.

To utilize the available link capacity fully, TCP requires at least the 32 KB buffer, at which all TCP variants record comparable retransmission rates. With auto-tuning, however, retransmissions significantly increase over the 32 KB buffer rates for New Reno and Cubic, by factors of 1.34 and 1.37, respectively.

4.3. Loss process and effects of auto-tuning

To analyze the dynamics within each TCP connection, we employ time-series plots of congestion window values, focusing on the downlink. Congestion window is tracked using outstanding data calculated by *tcptrace*, as a good approximation. We divide the results into two groups. The first group includes smaller buffer sizes of 8, 16 and 32 KB, which do not allow a single flow to utilize the link bandwidth fully. The second group, 64 KB and auto-tuned buffers, allows full link utilization. These two groups show different behavior in congestion window evolution and retransmissions.

Figure 10 shows a typical evolution of a congestion window when smaller buffer size is used. We use Cubic with 16 KB buffer to show that congestion window remains steady throughout the connection. There is only an occasional segment retransmission, if any, experienced by each TCP variant using each of the three smaller buffer sizes. The observed behavior pattern further confirms that all TCP variants perform equally when they are limited by small buffer size.

The second group of results, using larger buffer sizes, 64 KB and auto-tuned, illustrate the effect of larger buffers on retransmissions and some specific effects of auto-tuning. We discuss only Cubic and New Reno due to limited space. Vegas and Veno exhibit the same qualitative behavior as New Reno.

For easier understanding and comparison, we plot the congestion window evolution for Cubic as observed on a (wired) cable modem access link with 5 Mbps downlink and auto-tuned buffer (Figure 11). The 7-hop route is inside the Calgary metropolitan area, with *ping* RTT in the order of 20 ms, but the RTT for New Reno, Cubic, and Veno with auto-tuning exceeds 100 ms, comparable to the WiMAX network. Cubic very quickly enters its standard pattern, without any detrimental effect of auto-tuning.

Figure 12 shows that with 64 KB buffer, both New

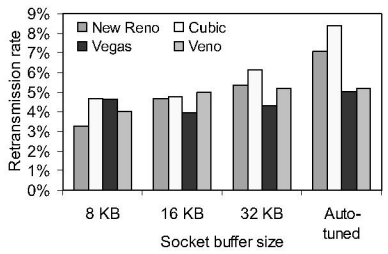

Figure 9: Uplink retransmissions

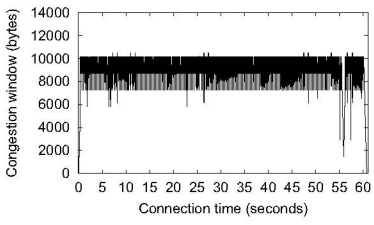

Figure 10: Congestion window evolution (16 KB)

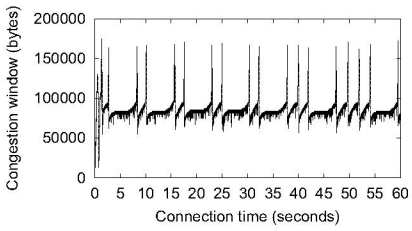

Figure 11: Cubic operation on cable modem

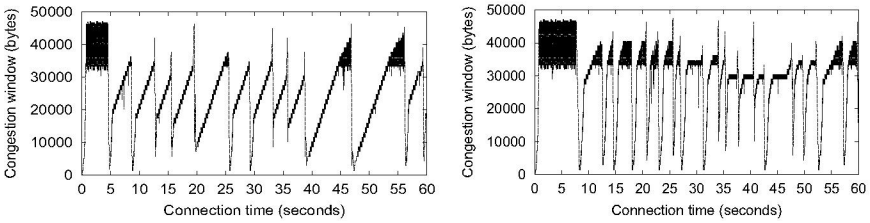

Figure 12: New Reno (left) and Cubic (right) with 64 KB buffer on WiMAX link

Reno and Cubic reach approximately the same window size before they experience loss. It takes both under 10 seconds to enter the congestion avoidance phase and start seeing losses, followed by periods of window reductions and growth. New Reno follows the well-known AIMD window evolution pattern.

We can also see that most retransmissions are caused by timeouts, since the congestion window size is reduced to one segment. What is not visible in the plot is that many retransmissions occur in pairs. These pairs of retransmissions occur at the end of the congestion avoidance epoch. The first retransmission is due to a triple duplicate ACK, followed by RTO triggering the second retransmission. The segments retransmitted within the pair are usually different. The natural question is whether the second retransmission is caused by a spurious timeout [11].

Spurious timeout occurs when RTT suddenly increases, such that it exceeds the pre-determined RTO timer. TCP interprets this as congestion, but in wireless networks, RTT spike can be caused by link-layer ARQ or hand-off. If RTT quickly returns to normal, actions triggered by RTO, which are congestion window reduction and retransmission, are unnecessary. In our case with paired retransmissions, indicators of spurious timeout would be:

1. The ACK for the second segment arrives in less than minimum RTT after the retransmission, meaning that the segment was only delayed in transit, not lost.

2. The ACK for the first retransmitted segment (due to triple duplicate ACK) cumulatively includes the second segment, meaning that the second segment was in transit and delayed or even received when RTO occurred.

Detailed examination of three traces reveals that neither of the indicators occurs. This suggests that RTO was inevitable and caused by real segment loss and not delay variations due to ARQ on the link layer. Congestion window size exceeding the BDP at time of loss also supports this argument. Therefore, we find no evidence that spurious timeouts regularly occur. Timestamp option is used for TCP, which also helps avoid spurious timeouts.

We next explore the effects of TCP window auto-tuning in more detail. The initial indicators that auto-tuning may adversely affect TCP performance are high retransmission rates (Figure 6).

Congestion window evolution with auto-tuning is shown in Figure 13 for New Reno and Cubic. The plot of the Cubic connection shows very erratic adjustments of the congestion window. We see that only in the late portion of the flow does Cubic manage to enter its steady-state operating pattern, where the window oscillates around the BDP. Auto-tuning clearly confuses the Cubic algorithm, by making it attempt to capture bandwidth far beyond the available limit. New Reno and Veno show similar initial behavior, but they reduce the window growth much sooner, and then enter into their normal operating pattern. TCP Vegas appears to be largely unaffected by auto-tuning, and does not attempt to increase the congestion window like other variants, regardless of the available buffer size.

4.4. Multiple connections

We run multiple connections to investigate fairness in TCP throughput between connections. We start all connections from the same end machine at approximately the same time and let them run concurrently for 60 seconds over downlink. Figure 14 shows the average and range of throughputs for 4 and 10 concurrent connections.

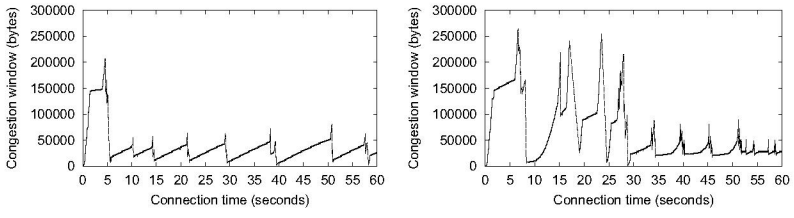

Figure 13: New Reno (left) and Cubic (right) with auto-tuning on WiMAX link

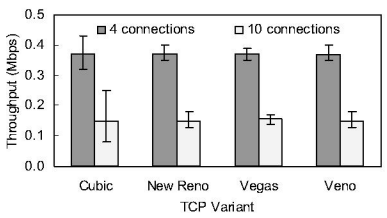

Figure 14: TCP connections fairly share bandwidth

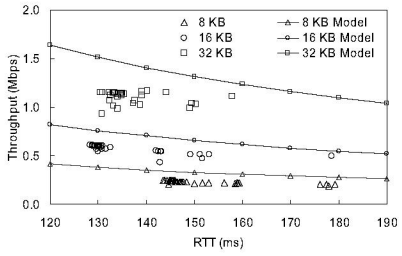

Figure 15: TCP throughput model (residential)

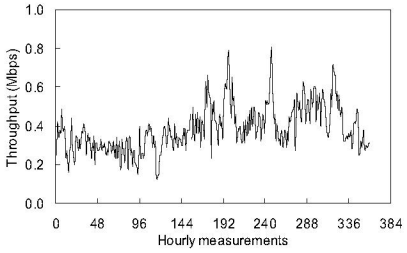

Figure 16: Long-term throughput (campus)

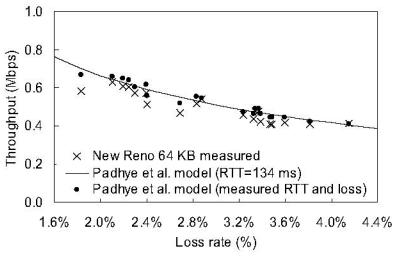

Figure 17: TCP throughput model (campus)

Mean throughput is inversely proportional to the number of concurrent connections, i.e. 4 connections achieve 25% of total throughput each, and 10 connections achieve 10% of total throughput each, on average. This confirms that multiple connections fully utilize the available bandwidth. The range indicates fairness among connections and we see that fairness is very high for New Reno, Vegas and Veno. Cubic registers higher range, but it does not cause any connection to starve.

For more formal fairness evaluation, we use Jain's Fairness Index (JFI) [7]. JFI is always in the interval [0, 1]; where value of 1 means perfect fairness, where all flows achieve their fair share of bandwidth. JFI is calculated as:

$$f(x_1, x_2, ..., x_n) = \frac{(\sum x_i)^2}{n \sum x_i^2},$$

where $x_i = O_i/T_i$, O_i is observed throughput of flow i, and T_i is fair throughput of flow i. Fair throughput is calculated by any appropriate means, and in this scenario it is the total available throughput divided by number of concurrent flows n.

In our test case, JFI exceeds 0.91 even for Cubic with 10 connections, and exceeds 0.98 in all other cases. This result confirms the design objective of WiMAX MAC layer, where BS-controlled access facilitates fairness for same-class traffic in the same direction, in contrast to CSMA/CA in WiFi, which allows unfairness to occur [5].

4.5. TCP throughput model

We compare the New Reno throughput measurements to known models from the literature. We are interested in a fit to a model when throughput is window-limited, i.e. for buffer sizes of 8, 16, and 32 KB. As shown in Figure 15, we find that the nearest match, although not very good, is to the simplest model: $B = 0.75W/RTT$; where B is the throughput, W is the maximum advertised congestion window size and RTT is the average RTT of the connection [9]. It appears that the model predicts the trend of the measurements well, but overestimates the throughput. Further exploration of the appropriate model is left for future work.

5. Additional experiments and results

5.1. Throughput at campus location

We repeated measurements about three months after the first set of results were collected at campus location and noticed that the network does not offer maximum throughput to TCP anymore. To establish that we are not observing a transient state, we continued to monitor network performance at this location for the next 8 months, and determined that low throughput persists.

In Figure 16 we plot the long-term observations of downlink throughput, taken from one-minute New Reno connections on each hour for 2 weeks. We observe that throughput varies and does not reach even 1 Mbps. These results also confirm that time of day does not play a role.

It is very difficult to compare TCP variants under these conditions, given the fluctuations in achievable throughput. We conducted two sets of experiments with 1-minute connection using each TCP variant back-to-back, repeated 20 times, so that each of 20 measurements for each variant comes from 4-5 minute intervals. Throughput fluctuated, but TCP Veno consistently recorded highest throughput. Vegas performed slowest of the four variants. Further comparison of TCP variants under these conditions is left for future work.

We do not have PHY layer measurements or access to the base station to determine user load, therefore we cannot positively identify the cause of lower throughput. However, we have a plausible explanation using other measurements. We eliminate poor channel conditions because moving the modem to the location where the strongest signal is registered by the modem does not help the throughput. Even more surprisingly, UDP throughput is near the maximum, completely unaffected. Uplink throughput remained stable near the maximum, with occasional drops, but not drastically like on the downlink. UDP streams and uplink TCP connections were run immediately before and after the downlink tests.

The findings suggest that some traffic classification scheme may be in effect on the downlink, and that the

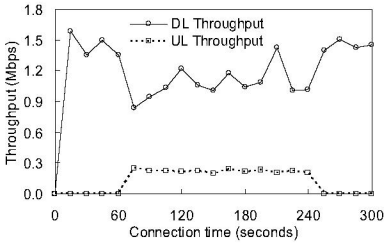

Figure 18: Bidirectional throughput

Table 2: Bidirectional performance

Location	Residential		Campus	
Measurement	**Downlink**	**Uplink**	**Downlink**	**Uplink**
Before/after t-put (Mbps)	1.37	0.24	0.40	0.21
Throughput (Mbps)	1.13	0.22	0.14	0.18
Average RTT (ms)	178	189	321	334
Retransmission rate	0.50%	7.03%	5.18%	3.41%

best-effort class is under-provisioned for the user demand, causing TCP throughput to suffer and fluctuate over intervals of 1 minute. We continue to investigate the causes for the reduced throughput at campus location.

5.2. TCP throughput model

In contrast to the residential location, we observe higher loss rates under low-throughput regime on campus, from 1.8% to 4.2% for New Reno connections. TCP throughput is now loss-limited rather than window or bandwidth-limited. In Figure 17 we show a good match of the TCP throughput model from Padhye et al. [13]. We use detailed trace analysis to obtain loss rate as defined in [13].

5.3. Bidirectional performance

Another important scenario in modern networks is concurrent uplink and downlink flows, as occurs in some peer-to-peer file sharing applications. This scenario reveals unfairness towards the downlink flow.

At residential location, where achievable throughput is stable and near full link capacity, we use 5-minute New Reno download and 3-minute New Reno upload. The upload starts about 1 minute after the download, such that the download connection runs alone for 1 minute before and after the upload. We use 64 KB buffer size that allows full link capacity utilization in both directions. Throughput results of two chosen experiments in Figure 18 clearly show that downlink throughput falls and then fluctuates while uplink connection is active. Overall results for the overlap period are shown in Table 2. Download recovers to near maximum link throughput after the upload completes. Uplink flow throughput remains stable throughout the connection, as if it were running alone. Correlation coefficient of -0.71 indicates significant negative correlation between uplink and downlink throughput.

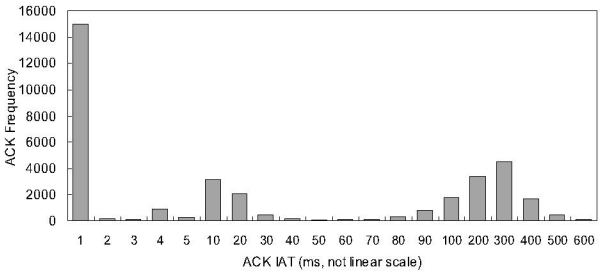

Figure 19: ACK IAT histogram

The unfairness is further exacerbated under low-throughput conditions at campus location. Due to throughput fluctuations, we use one 1-hour long New Reno connection in each direction. To get an approximate available bandwidth, we run five single-connection downlink tests before and after the experiment.

Table 2 shows the overall results for 1-hour trace. Throughput on uplink is higher than on downlink, and retransmission rate is higher on downlink than on uplink. Reduction in downlink throughput is much higher than on uplink, assuming that the available link throughput before and after the experiment is a good approximation of the average available link capacity throughout the experiment. Downlink RTT is very high, more than double RTT when full downlink capacity is available. Therefore, concurrent connections penalize the downlink flow significantly more than the uplink flow, in throughput, RTT, as well as loss rate.

After detailed analysis of the campus trace, we find the cause for the unfairness is ACK compression, which is a known problem affecting asymmetric links [8]. The trace inspection reveals that ACK arrivals occur in groups with gaps between them. The frequency histogram of ACK inter-arrival times (IAT) in Figure 19 shows that majority of ACK's arrive within 1 ms of each other. Gaps between arrivals last mostly between 90 and 400 ms, and between 5 and 20 ms, indicated by spikes in frequencies. This is in sharp contrast to the single TCP connection whose ACK arrivals are predominantly within 6-8 ms.

6. Summary and conclusions

In this paper, we present the measurement results from an early commercial deployment of a WiMAX-based

broadband wireless access network. We examine the network path characteristics and compare performance of uplink and downlink bulk data transfer for four TCP variants: New Reno, Cubic, Vegas and Veno.

The main results indicate that the WiMAX link does not adversely affect the intended operation of TCP variants, but that it is not well-suited for the aggressive Cubic and window auto-tuning. Therefore, the default settings of newer Linux kernels, namely TCP Cubic and auto-tuning are not recommended for connections traversing WiMAX links. While all TCP variants achieve similar throughput, they do so in different ways, from the very aggressive Cubic, to conservative and efficient Vegas. New Reno and Veno occupy the middle ground with robust performance and moderate loss. ARQ mechanism successfully shields TCP from random transmission losses.

Several performance issues identified can be attributed to causes other than WiMAX physical channel. Very high RTT is an artifact of inefficient routing involving transcontinental links. Unfairness during bidirectional transfers happens due to the known problem of ACK compression, but it could be reduced with better channel management that would eliminate congestion of TCP flows. The WiMAX link, however, is responsible for high variability in RTT during the connections, to which Vegas and Veno adapt most successfully.

While the results we present are specific to this particular WiMAX-based network, they are useful as an initial report on TCP performance obtained in this type of environment, which reflects the state-of-the-art for a commercially deployed WiMAX service. To generalize results for other WiMAX networks, additional measurements are needed from the latest deployments of the residential outdoor and mobile WiMAX access links. Our future work will continue by measuring TCP performance over new WiMAX deployments and for other Internet applications, including VoIP and multimedia streaming, as well as effects of TCP window auto-tuning on other network technologies, such as WiFi.

7. Acknowledgments

The authors thank the anonymous reviewers for their constructive feedback. This research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC), Alberta Ingenuity Fund, Killam Trusts, and Informatics Circle of Research Excellence (iCORE).

References

[1] H. Balakrishnan, V. N. Padmanabhan, et al., The Effects of Asymmetry on TCP Performance, Proc. of MobiCom, Budapest, Hungary, 1997.

[2] L. S. Brakmo, S. W. O'Malley, et al., TCP Vegas: new techniques for congestion detection and avoidance, Proc. of SIGCOMM, London, United Kingdom, 1994.

[3] S. Floyd, T. Henderson, et al. RFC 3782: The NewReno Modification to TCP's Fast Recovery Algorithm, IETF, http://www.ietf.org/rfc/rfc3782.txt.

[4] C. P. Fu and S. C. Liew, TCP Veno: TCP enhancement for transmission over wireless access networks, *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 216-228, 2003.

[5] M. Heusse, F. Rousseau, et al., Performance Anomaly of 802.11b, Proc. of INFOCOM, San Francisco, CA, 2003.

[6] IEEE Std 802.16e(TM)-2005 and IEEE Std 802.16(TM)-2004/Cor 1-2005, IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems.

[7] R. Jain, A. Durresi, et al. Throughput Fairness Index: An Explanation, http://www.cse.wustl.edu/~jain/atmf/ftp/af_fair.pdf.

[8] L. Kalampoukas, A. Varma, et al., Improving TCP throughput over two-way asymmetric links: analysis and solutions, Proc. of SIGMETRICS, Madison, WI, United States, 1998.

[9] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 4 ed: Pearson Education, Inc., 2008.

[10] Y. Lee, Measured TCP Performance in CDMA 1x EV-DO Network, Proc. of PAM, Adelaide, Australia, 2006.

[11] K. Mattar, A. Sridharan, et al., TCP over CDMA2000 Networks: A Cross-Layer Measurement Study, Proc. of PAM, Louvain-la-Neuve, Belgium, 2007.

[12] Network Monitoring Tools, SLAC, http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html.

[13] J. Padhye, V. Firoiu, et al., Modeling TCP Throughput: A Simple Model and its Empirical Validation, Proc. of SIGCOMM, Vancouver, BC, Canada, 1998.

[14] J. A. Perez, B. Donnet, et al., Preliminary Analysis of the TCP Behavior in 802.16 Networks, Proc. of WEIRD Workshop on WiMAX, Wireless and Mobility, Coimbra, Portugal, 2007.

[15] S. Ramachandran, C. W. Bostian, et al., A link adaptation algorithm for IEEE 802.16, Proc. of WCNC, New Orleans, LA, 2005.

[16] I. Rhee and L. Xu, CUBIC: A New TCP-Friendly High-Speed TCP Variant, Proc. of PFLDnet, Lyon, France, 2005.

[17] Y. Tian, K. Xu, et al., TCP in Wireless Environments: Problems and Solutions, in *IEEE Communications Magazine*, vol. 43, 2005, pp. S27-S32.

[18] E. Weigle and W. Feng, Dynamic right-sizing: a simulation study, Proc. of ICCCN, Scottsdale, AZ, 2001.

[19] J. Wu, D. Kim, et al., TCP Performance over the WiBro Compatible 802.16e Systems, Proc. of ICACT, Gangwon-Do, Korea, 2007.

[20] X. Yang, M. Venkatachalam, et al., Exploiting the MAC layer flexibility of WiMAX to systematically enhance TCP performance, Proc. of Mobile WiMAX Symposium 2007, Orlando, FL, 2007.