Enhanced Bandwidth Allocation for TCP Flows in WiMAX Networks

Eun-Chan Park[1], Chunyu Hu[2], and Hwangnam Kim[3]

[1] Department of Information and Communication, Dongguk University, Korea
[2] Wireless Networking Business Unit, Broadcom Corporation, CA, USA
[3] School of Electrical Engineering, Korea University, Korea

Abstract. In this paper, we propose a *bidirectional* bandwidth-allocation mechanism to improve TCP performance in the IEEE 802.16 WiMAX networks. According to the IEEE 802.16 standard, when serving a downlink TCP flow, the transmission of the uplink ACK, which is performed over a separate unidirectional connection, incurs additional bandwidth-request/allocation delay. Thus, it increases the round trip time of the downlink TCP flow and results in the decrease of throughput accordingly. First, we derive an analytical model to investigate the effect of the uplink bandwidth-request/allocation delay on the downlink TCP throughput. Second, we propose a simple, yet effective, bidirectional bandwidth-allocation mechanism that couples the bandwidth allocation for uplink and downlink connections by using either *proactive bandwidth allocation* or *piggyback bandwidth request*. The proposed scheme reduces unnecessary bandwidth-request delay and the relevant signaling overhead due to proactive allocation; meanwhile, it maintains high efficiency of uplink bandwidth usage by using piggyback request. Moreover, our proposed scheme is quite simple and practical; it can be simply implemented in the base station without requiring any modification in the subscriber stations or resorting to any cross-layer signaling mechanisms. The simulation results ascertain that the proposed approach significantly increases the downlink TCP throughput and the uplink bandwidth efficiency.

Keywords: IEEE 802.16e MAC, bandwidth request & allocation, TCP performance.

1 Introduction

The emerging broadband wireless access (BWA) network based on the IEEE 802.16e [1], called *Mobile WiMAX*, is one of the most promising solutions for the last mile broadband wireless access to support high data rate, high mobility, and wide coverage at low cost. The International Telecommunication Union (ITU) approved Mobile WiMAX as an International Mobile Telecommunication (IMT) advanced technology in October 2007. According to the WiMAX forum, the number of Mobile WiMAX users in the world is expected to grow up to 93 millions by 2012. On the other hand, TCP has been widely used in most

N. Bartolini et al. (Eds.): QShine/AAA-IDEA 2009, LNICST 22, pp. 363–379, 2009.

communication networks since the late 1980s, and it is still the most popular transport-layer protocol for reliable transmission in the Internet. Therefore, it is imperative to study and optimize the performance of TCP in Mobile WiMAX networks.

In this paper, we propose a solution for enhancing the TCP performance in Mobile WiMAX networks by means of efficient bandwidth allocation in the medium access control (MAC) layer. First, we show that the bandwidth-request delay, which is incurred in transmitting uplink TCP acknowledgements (ACKs), degrades the performance of the downlink TCP flow. Since the ACK packets are served with a separate uplink connection in Mobile WiMAX network, they require bandwidth-request/allocation procedure. This procedure incurs additional delay; therefore, the round trip time (RTT) of the downlink TCP flow is increased and the throughput is remarkably decreased. Moreover, we derive an analytical model for evaluating the effect of the bandwidth-request delay on the throughput of the downlink TCP flow. The numerical results based on the analysis model reveal that the downlink throughput decreases by about $20\% \sim 30\%$ under a typical configuration due to bandwidth-request delay.

In order to resolve this problem, we propose a framework of *bidirectional* connection that couples the bandwidth allocations for two unidirectional connections (one for downlink TCP data and the other for uplink TCP ACK). Within this framework, we propose a simple and effective bandwidth allocation mechanism that combines *proactive bandwidth allocation* with *piggyback bandwidth request*. The former allocates the bandwidth for the TCP ACK in a proactive manner; when a base station (BS) serves a downlink TCP data packet, the BS grants the bandwidth for the corresponding TCP ACK without any explicit request from the subscriber station (SS). The latter lets SS request bandwidth for the TCP ACK in a piggyback manner; SS carries the bandwidth-request for the subsequent ACKs in the header of on-going packet as long as there is ongoing uplink transmission.

The proposed approach decreases the bandwidth-request delay for the TCP ACK packets and reduces the overhead that is incurred in the bandwidth-request process. Implementing our proposed scheme is simple and practical, it is achieved by monitoring bandwidth-request queues managed by the BS without requiring any information or modification in the SS. This approach is a MAC-layer solution to improve the TCP performance; thus, it does not require any change in the TCP sender or receiver. Also, it can be incrementally deployed and widely extended to any centralized scheduling framework with a reliable transport protocol employing ACK mechanism. The OPNET [2] simulation results show that the proposed bidirectional approach increases the downlink TCP throughput up to about 40% compared with the conventional unidirectional bandwidth allocation, and it maintains high efficiency of the uplink bandwidth allocation.

There have been several proposals for efficient bandwidth request and allocation mechanisms in the IEEE 802.16 BWA networks in the literature [3, 4, 5, 6]. They mostly focused on QoS scheduling algorithm and architecture, but they did not consider the TCP characteristics. TCP-aware uplink scheduling scheme

was recently proposed in [7], [8] to assure fair resource allocation among the competing uplink TCP flows. Also, the study in [9] dealt with the collision in the contention-based bandwidth request process, which may occur during the transmission of uplink TCP ACK. Our study differs from previous studies as follows: (*i*) we investigate the interaction between TCP and 802.16 MAC, and analyze the performance degradation in the downlink TCP flow resulting from the bandwidth allocation for the uplink TCP ACK, (*ii*) we propose the bidirectional bandwidth allocation aiming at increasing the throughput of the downlink TCP flow without decreasing the efficiency of the uplink bandwidth allocation, (*iii*) the proposed approach is transparent to a scheduling algorithm, i.e., any advanced downlink/uplink scheduling algorithm can be incorporated into the proposed framework to improve efficiency or QoS.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the QoS scheduling framework of the IEEE 802.16, and we state the problem related to the bandwidth request and allocation for the TCP ACK. Next, we derive the analytical model of the TCP throughput by considering the bandwidth-request process in Section 3. In Section 4, we propose the framework and algorithm for the bidirectional bandwidth allocation. In Section 5, we evaluate the performance of the proposed approach via simulations. Finally, we conclude this paper in Section 6.

2 Problem Statement

2.1 IEEE 802.16 Scheduling Framework

This study considers the point-to-multipoint architecture of the IEEE 802.16 networks, where the communication between BS and SS is controlled by the BS. The transmissions are all made over unidirectional connections that are either downlink (DL) (from BS to SS) or uplink (UL) (from SS to BS). When a connection is established with the specific QoS requirements, the connection admission control comes into play at the BS based on the information of the advertised QoS requirements and the available resource. Once the connection is admitted, the BS schedules both DL and UL connections in a centralized way. The BS maintains two types of queues for scheduling, *data transmission queues* for DL connections and *bandwidth request queues* for UL connections. Based on the QoS requirements specified for each connection (e.g., the tolerable delay and the minimum reserved rate), the BS schedules the DL connections with the transmission queues and UL connections with the request queues, independently. Unlike the DL connections, the bandwidth for the UL connections is allocated on a reservation-basis or on a request-basis depending on the scheduling class. After completing the scheduling process, the BS generates and broadcasts DL/UL MAP messages that contain two dimensional (time and frequency) resource allocation information. When receiving the DL/UL MAP, SS decodes a DL frame and transmits a UL frame in the specified time and frequency of the OFDMA/TDD (Orthogonal Frequency Division Multiple Access with Time Division Duplex) frame.

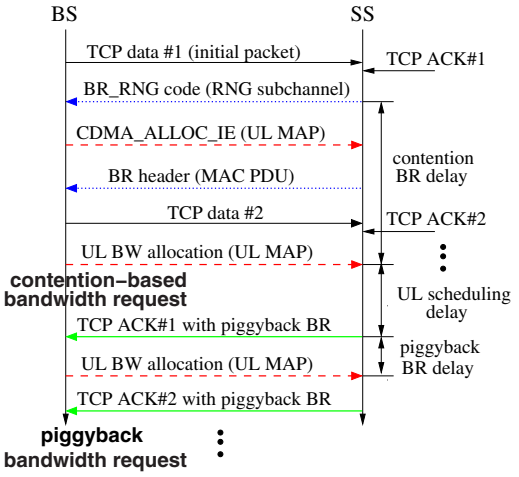


Fig. 1. Bandwidth-request procedure for TCP ACK packets; contention-based request and piggyback request

2.2 Bandwidth Request for TCP ACK

Considering TCP ACK packets are served with a best-effort (BE) connection, there are two standardized bandwidth-request mechanisms for serving them [1]: the *contention-based request* and the *piggyback request*, which are referred to as *contention* and *piggyback* hereafter, respectively.

In the *contention* method, the SS takes the following four-step request-response procedure for transmitting a TCP ACK, as illustrated in Fig. 1[1]; (*i*) the SS picks a random bandwidth-request ranging (`BR_RNG`) code, which is modulated into a dedicated contention-free ranging channel and it is delivered to the BS; (*ii*) the BS detects the `BR_RNG` code, and then sends a `CDMA_Allocation_IE` message in the UL MAP to inform SS of the transmission region for a BR message; (*iii*) the SS sends a stand-alone BR MAC header as a MAC protocol data unit (MPDU) that specifies the required amount of bandwidth; (*iv*) the BS allocates the required bandwidth by sending the UL MAP back to the SS. Finally, the SS can transmit its TCP ACK packet by using the allocated bandwidth. This procedure inevitably incurs a processing delay that is approximately two tenths of a millisecond. Sometimes the delay may increase up to a few hundred milliseconds due to the collisions and the subsequent backoff/retransmissions (when two or more SSs choose the same `BR_RNG` code and simultaneously send them). In this case, the delay can cause TCP-level time-out and retransmission, which drastically decrease the TCP throughput.

On the other hand, the SS can deliver the ACK packets via the *piggyback* method. For the MAC frames backlogged in the transmission queue of the SS, the corresponding BR message can be piggybacked in the sub-header of the

[1] For simplicity, we do not consider the delayed ACK mechanism [10] or the fragmentation/packing of MAC service data unit (MSDU) in Fig. 1.

on-going MAC frame. Fig. 1 shows that the second TCP ACK is delivered by the *piggyback*, while the first ACK is delivered by the *contention*. Compared to the *contention* method, the *piggyback* method neither requires contention for the BR opportunity, nor incurs long delay. Moreover, the *piggyback* method reduces signaling overhead; specifically, the size of the BR MAC header for the *contention* is 6 bytes, while the size of the sub-header for the *piggyback* is 2 bytes [1]. Consequently, it is more desirable to use the *piggyback* method for delivering TCP ACKs.

However, the *piggyback* method is only available when there exists at least one backlogged MAC frame in the transmission queue at the instant of generating a new MAC frame that contains TCP ACK. The *piggyback* method is not always available due to the following reasons:

- TCP data packets are generated and delivered in a bursty fashion, so the corresponding ACK packets are not regularly or periodically generated, i.e., the transmission queue in the SS is occasionally empty.
- When packet loss or TCP time-out occurs (which frequently happens in wireless networks), there is no choice but to perform the *contention* method to serve ACK packets because there is no on-going UL frame.
- The first ACK packet of the first data packet within a certain congestion window may not use the *piggyback*.

3 Modeling TCP Throughput with Bandwidth-Request Process

3.1 Model Derivation

We derive the TCP throughput model by considering and analyzing the effect of the BR delay on the throughput. We consider that a TCP connection is established to download an L-byte object. Here, we make several reasonable assumptions; (*i*) the wireless link between the BS and the SS is a bottleneck link and its capacity is constant, (*ii*) the buffer in the BS is properly provisioned to prevent buffer overflow, (*iii*) a retransmission mechanism, hybrid automatic repeat request (HARQ), recovers the wireless channel error and it assures in-order delivery of the MPDUs according to the IEEE 802.16 specification [1]. Let us denote p_e as the final target packet error rate with the HARQ retransmissions and denote W_{th} and $W_{max}(> W_{th})$ as *slow start threshold* and *advertised window size*, respectively. Then, we model the TCP congestion window $w(k)$ at the kth RTT stage as the Markov chain and we represent the state transition probabilities as:

$$\begin{aligned}
&\text{Prob}[w(k+1) = 2^{1/b}w \mid w(k) = w < W_{th}] = (1-p_e)^w \\
&\text{Prob}[w(k+1) = w + 1/b \mid w(k) = w \geq W_{th}] = (1-p_e)^w \\
&\text{Prob}[w(k+1) = W_{max} \mid w(k) = W_{max}] = (1-p_e)^w \\
&\text{Prob}[w(k+1) = w/2 \mid w(k) = w] = 1-(1-p_e)^w
\end{aligned} \tag{1}$$

Here, b denotes the number of packets acknowledged by one received ACK packet (if the delayed ACK mechanism is used, then the TCP receiver sends one cumulative ACK for two consecutively received packets, i.e., b is 2). This stochastic process models three phases of the TCP congestion window: slow-start, congestion avoidance, and fast recovery, but it neglects the TCP time-out due to the assumptions of a large buffer size and the HARQ retransmissions. Next, we calculate the number of RTT stages required for downloading the L-byte object, defined as N. We consider that packet losses are not correlated among the back-to-back transmissions within a congestion window (because the buffer size is large enough to avoid buffer overflow and the packet error in the wireless channel is random and irrelevant to the queuing discipline). We also consider that a lost packet, even after the HARQ retransmissions, is recovered by a single TCP retransmission with probability close to one since $p_e \ll 1$. Thus, we can represent N as

$$N = \text{argmin}_n \sum_{k=1}^{n} w(k) > L' + \sum_{k=0}^{L'} k \binom{L'}{k} p_e^k (1-p_e)^{L'-k}, \tag{2}$$

where $L' = \lceil L/MSS \rceil$ and MSS (byte) denotes the maximum segment size of TCP. The first and second terms on the right side of (2) represent the initial transmission and the retransmission of the TCP packets. The second term on the right side of (2) is equal to the mean of the binomial distribution; so the right side of (2) becomes $L'(1+p_e)$.

Next, we model the RTT at the kth stage, $RTT(k)$. Fig. 2 depicts the TCP timing diagram between the sender (server) and the receiver (SS). Note that the BS is located between the server and the SS but it is not explicitly shown in Fig. 2. We define R (byte/sec) as the effective capacity to process the TCP payload. At the receiver-side, $RTT(k)$ can be considered as the difference between the time when the receiver starts receiving the first packet in $w(k)$ and the time when it does in $w(k+1)$, i.e.,

$$RTT(k) = \frac{w(k)MSS}{R} + t_{idle}(k), \tag{3}$$

where $t_{idle}(k)$ is the idle time between the time when finishing receiving the last packet in the kth RTT stage and the time when starting receiving the first packet in the $(k+1)$th RTT stage. On the other hand, the RTT can be modeled at the sender-side as the difference between the time when the sender starts transmitting the first packet in the congestion window and the time when the sender receives the corresponding ACK packet, i.e.,

$$RTT(k) = \frac{MSS}{R} + t_{br}(k) + t_q(k) + RTT_{min}. \tag{4}$$

Here, $t_{br}(k)$ denotes the BR delay of the ACK for the first data packet in the kth RTT stage, and it can be modeled as

$$t_{br}(k) = \begin{cases} T_{ct} & \text{if } t_{idle}(k-1) > 0, \\ T_{pb} & \text{if } t_{idle}(k-1) = 0, \end{cases} \tag{5}$$

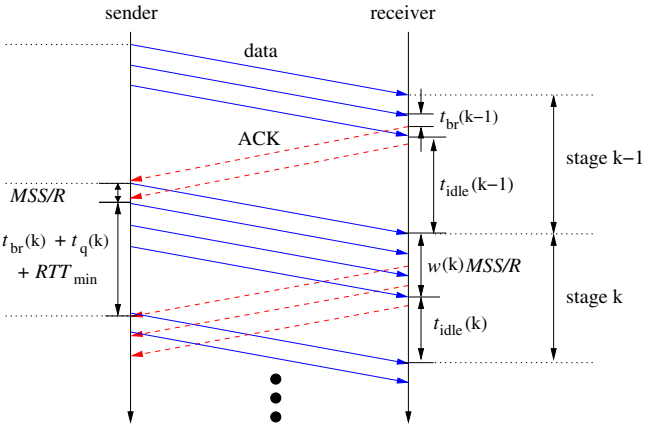


Fig. 2. TCP timing diagram at sender-side and receiver-side

where T_{ct} and T_{pb} are the contention-based and the piggyback-based BR delays, respectively. We define $t_q(k)$ as the queuing delay in the BS for the first data packet in the kth RTT stage and we model it as the accumulated backlog until the $(k-1)$th RTT stage divided by the service rate at the kth stage, i.e.,

$$t_q(k) = \sum_{n=1}^{k-1} \left[w(n)\frac{MSS}{R} - RTT(n) \right]^+, \tag{6}$$

where $[x]^+ = max(0, x)$. In (4), RTT_{min} accounts for several components of the RTT except t_{br} and t_q, e.g., the propagation delay over wired links between the sender and the BS, the DL/UL scheduling delay in BS/SS, the ACK transmission delay, and several other processing delays. Although RTT_{min} may vary, we assume that it is constant in order to focus on the throughput reduction due to t_{br}.

The RTT is matched at both the sender-side and the receiver-side, thus, $t_{idle}(k)$ can be represented from (3) and (4) as:

$$t_{idle}(k) = \left[RTT_{min} + t_q(k) + t_{br}(k) - (w(k) - 1)\frac{MSS}{R} \right]^+. \tag{7}$$

Once $w(k)$ is obtained from the stochastic process described in (1); $t_q(k)$ and $t_{idle}(k)$ can be obtained from (6) and (7), respectively, and $RTT(k)$ is determined by (4). Finally, the average throughput, $\overline{TH}$, is given as

$$\overline{TH} = \frac{\sum_{k=1}^{N} w(k)MSS}{\sum_{k=1}^{N} RTT(k)}, \tag{8}$$

where N is given in (2). Note that the average throughput $\overline{TH}$ is not represented in a closed form, but it can be numerically obtained.

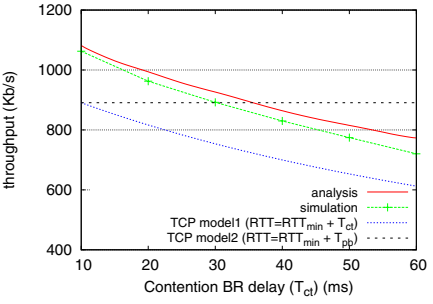


Fig. 3. Effect of bandwidth-request delay on TCP throughput

3.2 Model Validation

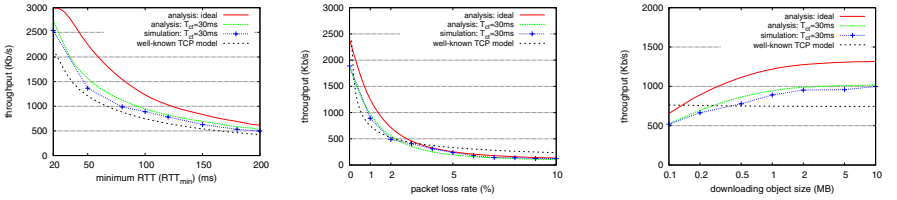
We validate the derived model by comparing its results with the simulation results. We also compare the analysis results with the well-known TCP throughput model [11], which is characterized by the RTT and packet loss rate as;

$$\overline{TH} = min\left(\frac{W_{max}MSS}{RTT}, \sqrt{\frac{3}{2b}}\frac{MSS}{RTT\sqrt{p_e}}\right). \tag{9}$$

Here, we consider b to be one without considering the delayed ACK mechanism for simplicity.

First, we observe the effect of the BR delay on the TCP throughput. We consider a typical configuration such that $W_{max} = 64$ KB, $MSS = 1$KB, $L = 1$MB, $RTT_{min} = 100$ ms, $T_{pb} = 10$ ms, $R = 5$ Mb/s, and $p_e = 0.01$. Figure 3 compares the analysis results with two TCP models, `TCP model1` and `TCP model2`, as well as with the simulation results to validate the analysis model. The results of `TCP model1` and `TCP model2` are obtained from (9) by setting their RTTs such that $RTT = RTT_{min} + T_{ct}$ (`TCP model1`) and $RTT = RTT_{min} + T_{pb}$ (`TCP model2`),[2] respectively. They are intended to represent the cases where the bandwidth is requested by either only the *contention* or the *piggyback*. Compared to the case of $T_{ct} = 10$ ms, the throughput obtained from the simulation and the analysis results is approximately decreased by 32% and 29% when T_{ct} is increased to 60 ms, respectively. The analytical throughput is slightly higher than that of the simulation result, which results from the fact that the model does not consider the delay variation due to the HARQ and the burst behavior of TCP packet loss. However, the throughputs with `TCP model1` and `TCP model2` deviate from the simulation results remarkably. The main reason of this deviation is the assumption made in deriving (9) i.e., once a packet is lost, the subsequent packets are dropped due to the buffer-overflow until the end of the given RTT stage. This assumption is reasonable in wired networks where the routers' buffers are

[2] In this configuration, we observed from the simulation that the queuing delay is negligible compared to RTT_{min} so it is not included in RTT calculation.



(a) effect of RTT (b) effect of packet error rate (c) effect of download object size

Fig. 4. Validation of TCP throughput model with various parameters

managed by the drop-tail queuing discipline; however, it is no longer valid in the wireless networks where packet loss is not highly related to the buffer-overflow, but it occurs randomly. On the other hand, `TCP model2` gives constant throughput, regardless of the change of the contention BR delay, T_{ct}. If $T_{ct} > 30$ms, then `TCP model2` overestimates the throughput compared to the simulation results. However, it underestimates the throughput when $T_{ct} < 30$ ms. These results in Fig. 3 confirm that the existing TCP throughput models cannot capture the effect of the contention-based BR delay but our analysis model is effective to do that.

Our derived analysis model can also evaluate the effect of various system parameters such as RTT (RTT_{min}), packet error rate (p_e), and download object size (L). Fig. 4 shows these effects on the achievable TCP throughput. Here, we set T_{ct} to the typical value of 30 ms, and we compare the analysis results with the simulation results. Additionally, Fig. 4 shows the maximum theoretical throughput that can be obtained from our analysis model by setting $T_{ct} = 0$ and $p_e = 0$. We also compare these results with that of the TCP model [11], where RTT in (9) is set as the average value from the simulation results. We set the default values of the parameters as $RTT_{min} = 100$ms, $p_e = 1\%$, $L = 1$MB, and $R = 5$ Mb/s. From Fig. 4, we observe the following:

- The analysis results agree well with the simulation results for the wide range of various system parameters, which confirms that the analysis model is effective and accurate.
- The existing TCP model underestimates the throughput in most cases. Furthermore, it fails to represent the effect of L, i.e., it gives a constant throughput regardless of the value of L, because the existing TCP model is intended to get the steady-state throughput without considering the slow-start phase of the TCP.
- Compared to the ideal case without the BR delay, the actual TCP throughput is decreased by about $10\% \sim 23\%$ for the whole configuration. Also the relative throughput, defined as the actual throughput divided by the ideal throughput, is decreased as RTT_{min} or L decreases. This implies that the BR delay effect is amplified when the RTT or the object size is small.

4 Bidirectional Bandwidth Allocation

We propose the bidirectional bandwidth allocation for the DL TCP data and the UL TCP ACK to reduce both the BR delay and the overhead. First, we propose a preliminary solution that proactively allocates the bandwidth for the UL TCP ACK when the BS serves the DL TCP data packet. Next, we elaborate on how this mechanism improves the efficiency of the UL bandwidth allocation by combining the proactive allocation with the piggyback request.

4.1 Proactive Bandwidth Allocation

The starting point of bidirectional bandwidth allocation is that a TCP flow essentially involves the bidirectional packet transmission, i.e., the sender transmits the data packets to the receiver while the receiver transmits the ACK packets to the sender. Also, the transmission of the TCP ACK is related to the transmission of the TCP data; no TCP ACK packet is generated until the TCP data packet is delivered to the receiver. However, the process of bandwidth allocation standardized in IEEE 802.16 works in a unidirectional way; the UL bandwidth allocation is completely independent of the DL bandwidth allocation. Under this rationale, we propose a bidirectional connection, where the bandwidth allocation for the UL TCP ACK is associated with the transmission of the DL TCP data. We consider the *proactive bandwidth allocation* (we call it *proaction*) mechanism as a naive approach. The BS scheduler proactively allocates bandwidth for the corresponding UL ACK packet whenever the DL TCP data packet is served by the BS. Thus, it is not necessary for the SS to request bandwidth and the *proaction* can remove the BR delay and the overhead that are caused by transmitting the UL ACK packets. Consequently, the DL TCP throughput can be increased.

However, this approach has two major drawbacks. First, if the delayed ACK [10] mechanism is used it wastes the UL bandwidth. The delayed ACK mechanism, which is implemented in most TCP protocols, lets the TCP receiver not send the ACK packets immediately after receiving the TCP data packets but the receiver waits for the arrival of the next in-order TCP data packet up to 500 ms. Combined with the cumulative ACK, the delayed ACK mechanism can effectively reduce the amount of ACK traffic. Roughly speaking, the receiver sends every other ACK packet on receiving TCP data packets. Consequently, the *proaction* may unnecessarily allocate bandwidth and significantly decreases the UL bandwidth efficiency. Second, the *proaction* cannot determine the accurate amount of bandwidth-request if the MSDU (TCP data packet or ACK packet) is fragmented or packed. In the IEEE 802.16 MAC, the MSDU fragmentation/packing frequently occurs in the scheduling and automatic repeat request (ARQ) operation. If the BS serves the fragmented or packed TCP data packets, then it can hardly predict the amount of bandwidth required for the SS to serve the corresponding ACK packets. For these reasons, the *proaction* mechanism cannot be considered as a practical solution for the bidirectional bandwidth allocation.

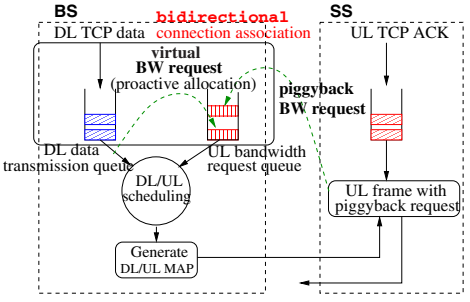


Fig. 5. Schematic diagram of bidirectional bandwidth allocation combining proactive bandwidth allocation with piggyback bandwidth request

4.2 Hybrid Approach

To overcome the drawbacks of the proactive bandwidth allocation, we propose a hybrid approach that combines the proactive bandwidth allocation with the piggyback bandwidth request. The proposed approach basically employs the *piggyback* method, and utilizes the *proaction* method only if the *piggyback* is not available. The *piggyback* does not have any problem involved to the delayed ACK mechanism and the fragmentation/packing since it allocates the bandwidth reactively (the SS first determines the amount of required bandwidth and then it requests the bandwidth). The basic idea is to first limit the usage of the *proaction* for efficient usage of the UL bandwidth, and then to replace the *contention* with the *proaction* for enhancing the DL throughput. The key point is determining the type of UL bandwidth allocation (*proaction* or *piggyback*) and determining the amount of bandwidth request.

Fig. 5 presents how the proposed approach works. Note the UL bandwidth request queue in the BS is managed for each connection. When serving a DL frame, the BS scheduler checks the associated UL request queue. If the UL request queue is empty, the scheduler puts a new bandwidth request to the request queue on behalf of the SS, i.e., the scheduler performs the *proaction*. The amount of bandwidth-request is set to be equal to the amount required to send one TCP ACK including the MAC header. On the other hand, when transmitting an UL frame, the SS checks whether the backlogged traffic is still present in its transmission queue. If the transmission queue is not empty, then the SS requests bandwidth in a piggyback manner at the amount of the backlogged traffic. In this way, the proposed hybrid approach increases the UL bandwidth efficiency due to the *piggyback* while decreasing BR delay due to the *proaction*.

The main strength of the proposed approach is that it does not require any modification of the SS and it can be simply implemented in the BS. Moreover, the proposed mechanism does not have any control parameters on which its performance depends, and thus it is unnecessary to tune the parameters. The proposed mechanism only monitors the state of the UL request queue maintained in the BS to determine the type of bandwidth-request, so it neither requires information about the transmission queue of the SS nor results in additional

signaling overhead between the BS and the SS. In addition to simplicity, this approach can be incrementally deployed because it does not requires any changes of the SS. The proposed approach can also be incorporated with any scheduling algorithm to further improve overall utilization, fairness, or QoS.

5 Simulation

5.1 Simulation Setup

In the simulations, we consider the OFDMA/TDD PHY where the frame duration is 5 msec, the number of DL/UL symbols are 29/18, and the base frequency and the channel bandwidth are 2.5 GHz and 10 MHz, respectively. The wireless channel is modeled by using the empirical COST-231 HATA model [12], log-normal shadowing, and the ITU channel model [13] to consider the multipath fading effect. We implement the adaptive modulation and coding scheme in the simulator. Depending on the signal to interference noise ratio (SINR), the modulation and coding rate are dynamically changed among the followings: QPSK (1/12, 1/8, 1/4, 1/2, 3/4), 16QAM (1/2), 64QAM (2/3, 3/4, 5/6) for the downlink and QPSK (1/12, 1/8, 1/4, 1/2, 3/4), 16QAM (1/2, 3/4) for the uplink. We emulate the HARQ and the ARQ mechanisms such that the target packet error rate of the HARQ is 1 %, the maximum number of the HARQ retransmissions (excluding the initial transmission) is three, and the retransmission delays of the HARQ and the ARQ are 30 ms and 100 ms, respectively. Also, they are modeled so that MSDUs are delivered in-order for reducing TCP retransmissions. The contention-based BR delay is uniformly set between 25 ms to 50 ms; the collision probability of `BR_RNG` code and its timer value are set to 1 % and 100 ms, respectively. We set the maximum TCP segment size to 1500 bytes and we use the delayed ACK mechanism. The minimum RTT, RTT_{min} is set to 100 ms. These configurations are typical operation scenarios for the TCP flows over the mobile WiMAX networks, as recommended by the IEEE 802.16m task group [14]. Note that the simulation results are averaged over ten instances of the simulation with different random seeds.

In the simulation study, we consider the following three bandwidth-request/allocation algorithms and we compare their performance;

(*i*) `UNI`: a conventional unidirectional approach as described in Sec. 2.2,
(*ii*) `blind-BI`: a bidirectional approach using the *proaction* without the *piggyback*,
(*iii*) `adapt-BI`: the proposed bidirectional approach that adaptively switches between the *proaction* and the *piggyback* depending on the state of the BR queue.

Together with simulation results of these algorithms, we will present the theoretical maximum throughput derived from the analysis model in Section 3, it is named as `ideal`. The performance metrics are set as:

– UL MAC-to-MAC delay: the time interval between generating MSDU on the SS and receiving it on the BS.

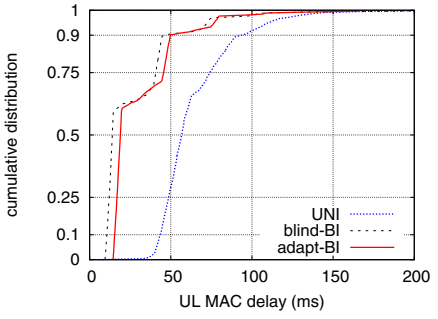


Fig. 6. Uplink MAC-to-MAC delay of UNI, blind-BI, and adapt-BI

- UL bandwidth allocation efficiency: B_{used}/B_{alloc}, where B_{used} is the cumulative amount of UL bandwidth that is actually used to transmit the TCP ACKs and B_{alloc} denotes the cumulative amount of bandwidth allocation (including the MAC header and the bandwidth-request message).
- DL throughput: the average goodput, which is calculated as the download object size divided by the download completion time.

5.2 Tradeoff between Performance and Efficiency

In the first simulation, we focus on the tradeoff between performance and efficiency in the bandwidth allocation process. Here, we consider FTP download and we set the download object size to 1 MB.

First, we observe the UL MAC-to-MAC delay of UNI, blind-BI, and adapt-BI, whose cumulative distributions are shown in Fig. 6. The delay of blind-BI is minimized due to proactive bandwidth allocation, but that of UNI is much larger than the others due to the contention-based request. However, the adapt-BI considerably decreases the delay, which is at least 2 times smaller than that of UNI and is slightly higher than that of blind-BI. Specifically, the median delays of UNI, blind-BI, and adapt-BI are 56.6, 14.0, and 18.9 ms, and the 90th-percentile delays are 94.1, 46.7, and 49.8 ms, respectively. The sudden increase of the delay in Fig. 6 is caused by the HARQ/ARQ retransmissions.

Next, we evaluate the efficiency of the UL bandwidth allocation. Table 1 lists B_{alloc}, B_{used}, as well UL efficiency and DL throughput. The UNI utilizes the UL bandwidth efficiently, i.e., the difference between B_{alloc} and B_{used} is slight. However, in the case of the blind-BI, B_{alloc} is almost two times higher than B_{used}, i.e., almost half of the allocated bandwidth is wasted. The bandwidth wastage of blind-BI is remarkably higher than the other algorithms because it proactively allocates the bandwidth, regardless of whether or not the SS has packets to send. As indicated in Table 1, the bandwidth allocation waste is minimized by the adapt-BI; it is smaller than that of the UNI because the UNI has to send a 6-byte bandwidth-request message in the *contention* phase, which is unnecessary in the adapt-BI.

Table 1. Uplink efficiency and downlink throughput of UNI, blind-BI, and adapt-BI

algorithm	B_{alloc} (Kbyte)	B_{used} (Kbyte)	UL efficiency	DL throughput (Mb/s)
UNI	18.6	16.4	0.88	0.77
blind-BI	32.7	16.1	0.49	1.06
adapt-BI	17.9	16.3	0.91	1.01

In summary, the results in Fig. 6 and Table 1 confirm the following:

- Compared to the UNI, the adapt-BI decreases the average UL MAC-to-MAC delay by more than two times, and so it increases the average DL throughput by about 31%.
- the blind-BI achieves small gain of DL throughput over the adapt-BI at the significant cost of efficiency of the UL bandwidth allocation; almost half of the UL bandwidth is wasted to increase the DL throughput by about 5% compared to adapt-BI.
- Unlike the blind-BI, the adapt-BI maintains high efficiency of the UL bandwidth allocation (higher than blind-BI by 42%); meanwhile, its DL throughput is comparable with that of the blind-BI.

5.3 Effect of RTT

In this simulation, we study the effect of RTT, which is a key factor affecting the TCP throughput. For this purpose, we change RTT_{min}, which can be set to an arbitrary value by excluding the variable components of RTT such as BR delay and scheduling/queuing delay, from 20 ms to 200 ms. Here, L is set to 1 MB.

As shown in Table 2, the UL MAC-to-MAC delay and the UL efficiency are almost insensitive to the change of RTT_{min}. Regardless of RTT_{min}, the adapt-BI considerably reduces the UL delay compared to the UNI, while it remarkably increases the UL efficiency compared to the blind-BI. As RTT_{min} increases, its effect on the throughput surpasses that of the BR delay. Therefore, the effect of the BR delay on the throughput is alleviated, i.e., the throughput increase of adapt-BI/blind-BI over UNI decreases as RTT_{min} increases. For example, if $RTT_{min} = 20$ ms, the adapt-BI gives higher throughput than the UNI by about 39%, but the throughput gain is decreased by 18% if $RTT_{min} = 200$ ms. These simulation results in Table 2 confirm the outstanding performance of adapt-BI in terms of bandwidth request delay, uplink bandwidth efficiency, and downlink throughput.

Next, we compare the ideal throughput calculated from the analysis model with those obtained from simulations. Table 2 shows that the throughput of adapt-BI is close to the ideal throughput; the slight difference between them is mainly caused by the fact that the analysis model does not include the HARQ/ARQ processing delays. Also, we can check the validity of the analysis model by comparing its ideal throughput with the throughput of blind-BI. Apart from the UL bandwidth efficiency, the blind-BI is considered to be the

Table 2. Performance comparison of UNI, blind-BI, and adapt-BI with various values of RTT

RTT_{min} (ms)	UL MAC delay (ms)			UL efficiency (%)			DL throughput (Mb/s)			
	UNI	blind -BI	adapt -BI	UNI	blind -BI	adapt -BI	UNI	blind -BI	adapt -BI	ideal
20	62.0	26.8	31.1	87.6	48.9	91.0	1.19	1.76	1.65	1.79
50	64.9	26.9	30.9	87.9	49.1	90.9	0.98	1.42	1.35	1.47
100	65.5	27.0	31.3	88.1	49.7	91.0	0.76	1.04	1.01	1.11
150	65.8	26.7	30.7	88.3	50.2	91.1	0.60	0.78	0.74	0.81
200	65.5	26.6	31.6	88.2	51.2	91.2	0.51	0.63	0.60	0.67

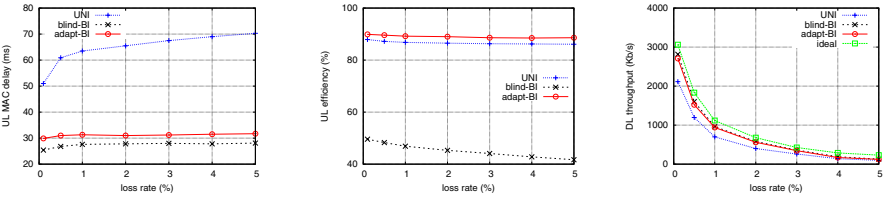
best solution that can maximize the DL throughput by minimizing the BR delay. The result in Table 2 that the analytical throughput is nearly equal to the throughput of blind-BI reconfirms the validity of our analysis model.

5.4 Effect of Packet Loss Rate

In this simulation, we evaluate the performance of the proposed mechanism with various values of packet loss rate. To focus on the effect of packet loss on the TCP throughput, we disable the retransmission mechanisms in PHY/MAC layer (HARQ and ARQ) and we randomly drop packets with probability of p ranging from 0.1% to 5%.[3] Here, we set L and RTT_{min} to 1 MB and 100 ms, respectively.

Fig. 7 compares several performance indices of the three algorithms with various values of p. It is noteworthy from Fig. 7(a) that the UL delay of UNI increases as p increases, but those of blind-BI and adapt-BI change very little with respect to p. The reason is as follows. A packet loss can result in the TCP time-out or increase the burstiness of packet transmission, then the packet loss possibly makes the transmission queue of the SS empty. Accordingly, the probability of the contention-based BR is increased and the UL MAC delay of UNI is also increased. On the other hand, the delays of blind-BI and adapt-BI are nearly irrespective of packet loss because of the proactive bandwidth allocation, i.e., even when the transmission queue of the SS is empty, the bandwidth is allocated without a contention-based request. Next, we can observe the UL efficiency from Fig. 7(b). As p is increased from 0.1% to 5%, the efficiencies of UNI and adapt-BI are both decreased by 1.8%, however, that of blind-BI is decreased by 7.9%. The blind-BI unconditionally allocates bandwidth for the UL TCP ACK, regardless of whether or not the DL TCP data packet is successfully delivered to the SS, so the allocated UL bandwidth may be wasted if the DL packet loss occurs. Consequently, the delay of blind-BI increases in proportion to the packet loss rate. Next, we investigate the effect of p on the throughput from Fig. 7(c). As was expected, the throughputs of all the three algorithms decrease rapidly as p increases. Although the absolute throughput gain of adapt-BI over UNI, which is the throughput difference between them, decreases with respect

[3] If HARQ and ARQ are enabled, most of the packet losses due to the wireless channel errors are recovered, so we cannot arbitrarily set the packet loss rate.



(a) uplink MAC-to-MAC delay (b) uplink bandwidth efficiency (c) downlink throughput

Fig. 7. Performance comparison of UNI, blind-BI, and adapt-BI with various values of packet loss rate

to the increase of p, and the relative throughput gain, which is the throughput of adapt-BI divided by that of UNI, remains within $23\% \sim 40\%$ for the entire range of p. We also observe from Fig. 7(c) that the throughput of blind-BI/adapt-BI is not quite deviated from the ideal throughput.

6 Conclusion

In this paper, we have proven that the DL TCP performance is degraded in the IEEE 802.16 wireless networks due to the bandwidth-request delay for transmitting the UL ACK. Moreover, we have derived the analytical model through which we can quantitatively analyze the effect of the bandwidth-request delay on the TCP throughput. The model is useful for predicting the maximum throughput gain that can be achieved by an ideal bandwidth request/allocation mechanism. To remove the unnecessary bandwidth-request delay and overhead that are involved in transmitting the UL ACK, we have proposed the bidirectional bandwidth allocation framework and the hybrid approach that combines the proactive bandwidth allocation with the piggyback bandwidth request schemes. Due to proactive bandwidth allocation, the proposed approach can reduce both bandwidth-request delay and overhead, and it can increase the DL throughput. At the same time, it can increase the efficiency of the UL bandwidth allocation due to the piggyback request, which is performed in a reactive manner. The simulation results have indicated that the proposed hybrid approach significantly increases the DL TCP throughput (by up to 40% compared to the case without the proactive allocation) as well as the UL bandwidth efficiency (by about two times compared to the case without the piggyback request). The advantages of the proposed scheme are that it is very simple and practical, and it requires neither changes of the SS nor additional signaling mechanism between the BS and the SS.

Acknowledgment

This work was supported in part by the IT R&D program of MKE/IITA [2008-F015-02, Research on Ubiquitous Mobility Management Methods for Higher Service Availability].

References

1. IEEE 802.16 WG, IEEE standard for local and metropolitan area networks part 16: Air interface for fixed and mobile broadband wireless access systems, Amendment 2, IEEE 802.16 Standard (December 2005)
2. OPNET WiMAX Model Development Consortium, "OPNET network simulator with WiMAX model (2007), `http://www.opnet.com/WiMax`
3. Wongthavarawat, K., Ganz, A.: Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems. International Journal of Communication systems 16, 81–96 (2003)
4. Cicconetti, C., Lenzini, L., Mingozzi, E., Eklund, C.: Quality of service support in IEEE 802.16 networks. IEEE Network 20, 50–55 (2006)
5. Liu, Q., Wang, X., Giannakis, G.B.: A cross-layer scheduling algorithm with QoS support in wireless networks. IEEE Trans. on Vehicular Technology 55, 839–847 (2006)
6. Park, E.-C., Kim, H., Kim, J.-Y., Kim, H.-S.: Dynamic bandwidth request-allocation algorithm for real-time service in IEEE 802.16 broadband wireless access networks. In: Proceedings of IEEE INFOCOM (2008)
7. Kim, S., Yeom, I.: TCP-aware uplink scheduling for IEEE 802.16. IEEE Communications Letters 11, 146–148 (2006)
8. Rath, H.K., Karandikar, A., Sharma, V.: Adaptive modulation-based tcp-aware uplink scheduling in ieee 802.16 networks. In: Proceedings of IEEE ICC (2008)
9. Kim, E., Kim, J., Kim, K.S.: An efficient resource allocation for TCP service in IEEE 802.16 wireless MANs. In: Proceedings of IEEE Vehicular Technology Conference (VTC)-Fall, pp. 1513–1517 (2007)
10. Braden, R.: Requirements for Internet hosts – Communication layers. IETF RFC 1122 (October 1989)
11. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP throughput: A simple model and empirical validation. In: Proceedings of ACM SIGCOMM, pp. 303–314 (1998)
12. Blaunstein, N.: Radio Propagation in Cellular Networks. Artech House (1999)
13. ITU-R Task Group 8/1, Guidelines for evaluation of radio transmission technologies for IMT-2000, Recommendation ITU-R M.1225 (1999)
14. IEEE 802.16 WG, Draft IEEE 802.16m evaluation methodlogy, IEEE 802.16 Standard (December 2007)