Bidirectional Bandwidth Allocation for TCP Performance Enhancement in Mobile WiMAX Networks

Eun-Chan Park*, Jae-Young Kim*, Hwangnam Kim†, and Han-Seok Kim*
* WiMAX system Lab, Telecommunication & Network Division, Samsung Electronics Co., LTD., Korea
E-mail: {eunchan.park|jay.m.kim|hs365.kim}@samsung.com
† School of Electrical Engineering, Korea University, Korea, E-mail: hnkim@korea.ac.kr

***Abstract*—We propose a *virtual bidirectional connection* for efficient bandwidth allocation to downlink TCP data and uplink acknowledgment (ACK) in mobile WiMAX networks based on IEEE 802.16. We observed that, in serving a downlink TCP flow, the standard bandwidth-request/allocation mechanism inevitably results in additional delay to transmit uplink TCP ACK, which increases round trip time of downlink TCP flow and decreases its throughput accordingly. To resolve this problem, we propose a hybrid approach combining *proactive bandwidth allocation* with *piggyback bandwidth request*. The proposed framework removes the unnecessary bandwidth-request delay, and also decreases signaling overhead involved in the bandwidth-request. Simulation results show that this approach significantly increases downlink TCP throughput and uplink bandwidth efficiency.**

***Index Terms*—IEEE 802.16e MAC, Mobile WiMAX, bandwidth request & allocation, TCP performance**

I. Introduction

The emerging broadband wireless access (BWA) network based on 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. ITU approved Mobile WiMAX as one of International Mobile Telecommunication (IMT) advanced technologies in October 2007 and more than 30 service providers in the world have plans to launch or launched commercial Mobile WiMAX services (e.g., Korea Telecom in Korea, Sprint in United States, and KDDI in Japan). On the other hand, TCP has been widely used in most communication networks since the late of 1980s. Although TCP has several drawbacks in wireless networks, it is still the common transport-layer protocol for reliable transmission. Thus, it is imperative to study and improve the performance of TCP in Mobile WiMAX networks.

This paper deals with TCP performance enhancement in Mobile WiMAX networks. We reveal that the performance of downlink TCP flow is degraded significantly due to bandwidth-request delay incurred in transmitting uplink TCP acknowledgement (ACK). Whenever a subscriber station (SS) receives TCP data packets, it transmits the corresponding TCP ACK packets to a base station (BS). Since these ACK packets are served with a separate uplink connection in Mobile WiMAX networks, the transmissions require bandwidth-request/allocation procedure in the centralized scheduling framework. This procedure thus incurs additional delay, so that it increases round trip time (RTT) of downlink TCP flow and in turn decreases its throughput remarkably.

In order to resolve this problem, we propose a framework of *virtual bidirectional* connection by allocating bandwidth for two *unidirectional* connections each of which is opposite-directional and used for TCP data packets and its corresponding ACK packets respectively. Within this framework, we also propose a hybrid bandwidth allocation mechanism combining *proactive bandwidth allocation* with *piggyback bandwidth request*. The fundamental operations are as follows: whenever serving a downlink TCP data (*i*) BS allocates the bandwidth for uplink ACK in a *proactive* manner if the uplink transmission queue of SS is empty; (*ii*) as long as there remains backlogged traffic in the uplink transmission queue, SS requests bandwidth in a *piggyback* manner so that the bandwidth-request can be delivered through the MAC frame header. The proposed approach removes bandwidth-request delay for TCP ACK packets and curtails down the overhead incurred in bandwidth-request process. Moreover, this approach requires no modification in SS. It can be simply implemented in BS by monitoring *bandwidth request queue* managed by the BS, without requiring the information of *data transmission queue* in the SS. We implemented the proposed algorithm in OPNET simulator [2]. The simulation results showed that the proposed hybrid approach increases downlink TCP throughput up to 40 % compared to the case without proactive bandwidth allocation, and increases uplink bandwidth efficiency by about 40 % against the case without piggyback bandwidth request.

There have been several proposals for efficient bandwidth request and allocation mechanisms for IEEE 802.16 BWA networks in the literature [3]–[6]. They mostly dealt with QoS scheduling algorithm and architecture, but they do not take TCP characteristics into account. Recently, TCP-aware uplink scheduling scheme has been proposed in [7] to realize max-min fairness based on the estimated sending rate without any bandwidth request. It deals with fair bandwidth allocation among uplink TCP flows, while this study aims at performance enhancement for downlink TCP flows. The study in [8] focuses on the issue of collision in contention-based bandwidth request process, which may occur in the transmission of uplink TCP ACK. Compared to the previous studies, this paper investigates an interaction between TCP and 802.16 MAC and addresses the performance degradation in downlink TCP flow resulted from bandwidth allocation for uplink TCP ACK, and proposes a practical solution to it. Moreover, this approach can be widely extended to any centralized scheduling framework with a reliable transport protocol employing ACK mechanism.

978-1-4244-2644-7/08/$25.00 ©2008 IEEE

The rest of this paper is organized as follows. In Section II, we briefly introduce QoS scheduling architecture of IEEE 802.16 BWA networks and state problems related to bandwidth request and allocation for TCP ACK. In Section III, we propose framework and algorithm for the bidirectional bandwidth allocation. We evaluate the performance of the proposed approach via simulations in Section IV. Finally, we conclude this paper in Section V.

II. Problem Statement

A. IEEE 802.16 scheduling framework

In a point-to-multipoint architecture of IEEE 802.16 networks, a BS controls all the communications between BS and SSs. All the transmissions are made over unidirectional connections, which are either downlink (DL) (from BS to SS) or uplink (UL) (from SS to BS). The BS schedules both DL and UL connections in a centralized manner, it maintains two kinds of queues, *data transmission queue* for DL connection and *bandwidth request queue* for UL connection. Depending on the scheduling class, the bandwidth for UL connection can be allocated on a reservation basis or on a request basis. Based on QoS requirements specified for each scheduling class (e.g., tolerable delay and minimum reserved rate), BS schedules DL connections in the transmission queues and UL connections in the request queues, independently. After scheduling, BS generates and broadcasts DL/UL MAP message containing two dimensional (time and frequency) resource allocation information. When receiving DL/UL MAP, SS can decode DL frame and transmit UL frame in the specified time and frequency. Depending on scheduling class, there are several standardized ways for SS to request bandwidth [1].

B. Bandwidth request for TCP ACK

Considering TCP ACK packets being served with a best-effort (BE) connection, there are two standard ways of bandwidth request mechanisms for them [1]; *contention-based bandwidth request* and *piggyback bandwidth request*, each of which is referred to as `contention` and `piggyback`, respectively, hereafter.

As for `contention` method, SS takes the following four-step request-response procedure as illustrated in Fig. 1 [1]; (*i*) SS picks a random bandwidth-request ranging (`BR_RNG`) code, which is modulated into the contention-free ranging sub-channel and delivered to BS; (*ii*) BS detects the `BR_RNG` code and sends a `CDMA_Allocation_IE` message in the UL MAP, which indicates the transmission region for the bandwidth-request message; (*iii*) SS sends a stand-alone bandwidth-request MAC header as a MAC protocol data unit (MPDU) in which the required amount of bandwidth is specified; (*iv*) BS allocates the required bandwidth by sending the UL MAP back to SS. Finally, SS can transmit its TCP ACK packet by using the allocated bandwidth. This procedure inevitably incurs additional delay, which is typically a couple

[1]For the simplicity, we do not consider delayed ACK mechanism [9] or fragmentation/packing of MAC service data unit (MSDU) in Fig. 1. Their effects will be discussed in the next section.

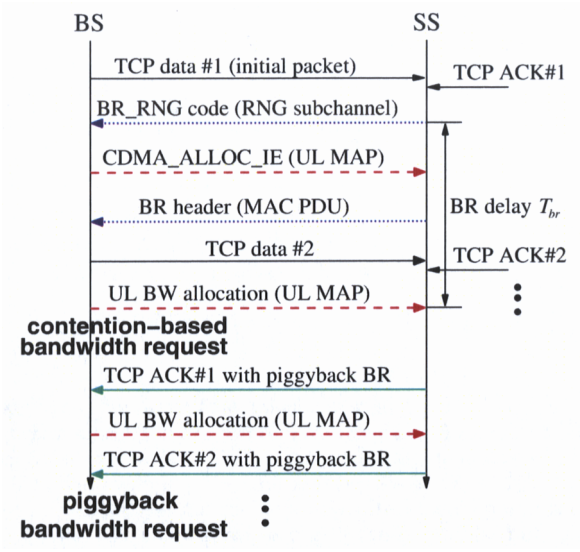


Fig. 1. Bandwidth-request procedure for TCP ACK packets; contention-based request and piggyback request.

of tens of milliseconds. Additionally, the delay may increase up to a few hundreds of milliseconds due to collisions and subsequent retransmissions (when two or more SSs choose the same `BR_RNG` code and send them at the same time). In this case, the delay can cause TCP-level timeout and retransmission, so that it decreases TCP throughput drastically.

On the other hand, SS can deliver ACK packets via the `piggyback` mechanism. The bandwidth-request message for the MAC frames backlogged in the transmission queue of SS can be piggybacked in the sub-header of on-going MAC frame [2]. In Fig. 1, the second TCP ACK is delivered by `piggyback`, while the first ACK is delivered by `contention`. Compared to `contention`, `piggyback` neither requires contention for bandwidth-request opportunity, nor incurs long delay. Moreover, `piggyback` reduces signaling overhead; specifically, the size of bandwidth-request MPDU in `contention` is 6 bytes, while the sub-header size of `piggyback` is 2 bytes [1]. Consequently, it is desirable to make use of `piggyback` in delivering TCP ACK.

However, it is important to note that `piggyback` is not always available. This is because (*i*) TCP data packets are generated in a bursty fashion, so the corresponding ACK packets are not generated back-to-back; (*ii*) the first ACK packet for the first data packet within a certain congestion window cannot be usually served by `piggyback`; and (*iii*) in the case of packet loss or TCP timeout (which frequently happens in wireless networks) there is no choice but to perform `contention` to serve ACK packet because there is no on-going UL frame.

We investigate the effect of bandwidth-request delay in the `contention` mechanism, denoted as T_{br} in Fig. 1, on DL TCP sequence number. For this purpose, we consider three cases of $T_{br} = 0$ (ideal case), 20, and 40 ms. In order to focus on the effect of T_{br}, the transmission rate of physical layer is set to be constant and the packet loss is deliberately not

[2]The on-going MAC frame does not have to be a TCP ACK packet.

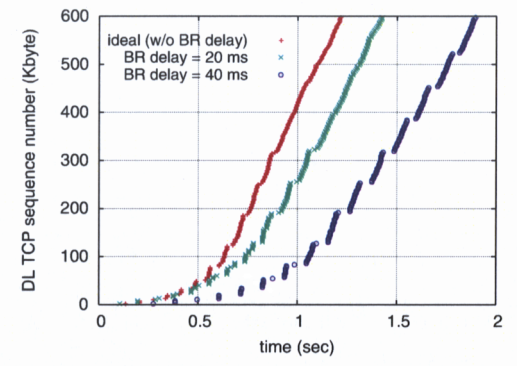


Fig. 2. Effect of bandwidth-request (BR) delay of the contention-based request mechanism.

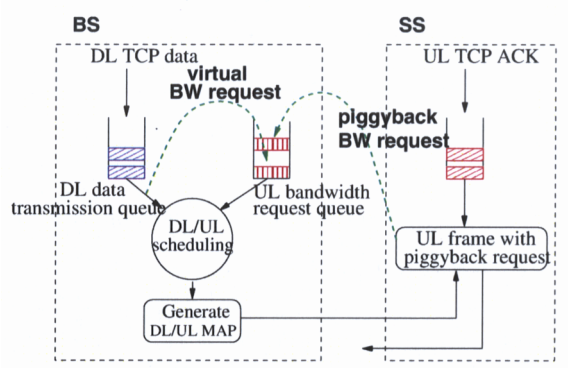


Fig. 3. Schematic diagram of bidirectional bandwidth allocation combining proactive bandwidth allocation with piggyback bandwidth request.

taken into account. In this simulation, TCP ACK packets are delivered through `piggyback` as long as it is available. In Fig. 2, the time gap between bursts of DL sequence number is resulted from T_{br}. We can observe from Fig. 2 that the increase rate of TCP sequence number decreases as T_{br} increases, implying throughput reduction. Also, we observe that the effect of T_{br} on the throughput degradation is more evident in the initial stage (i.e., when TCP is in slow-start phase). For example, in the cases of T_{br} = 0, 20, and 40 ms, the times to download the initial 100 KB are about 0.63, 0.75, and 1.06 sec, respectively, but the times to download another 100 KB starting from 400 KB are reduced to about 0.11, 0.13, and 0.15 sec, respectively. Moreover, Fig. 2 indicates an interesting result that in the case of T_{br} = 20 ms, the time gap does not appear (`contention` is not used any more and `piggyback` becomes available) after 1.2 sec at which the TCP congestion window size reaches its maximum value of 64 KB. In the case of T_{br} = 40 ms, however, the time gap appears periodically even after the TCP window size reaches the maximum value (e.g., after 1.4 sec). These results confirm that `piggyback` is not always available due to TCP burstness. Moreover, we can predict that the throughput reduction due to `contention` becomes worsen as packet loss rate increases or the downloading object size is small (e.g., web pages).

III. Bidirectional Bandwidth Allocation

We propose a bidirectional bandwidth allocation for DL TCP data and UL TCP ACK to reduce bandwidth-request delay and overhead. Since the process of bandwidth allocation standardized in IEEE 802.16 works in a unidirectional way, the bandwidth allocation for TCP ACK is completely decoupled from that for TCP data. However, TCP ACK is correlated to TCP data; no TCP ACK packet is generated until TCP data packet is delivered to the receiver. Thus, we assert that a bidirectional bandwidth allocation mechanism can be effective and efficient to deal with a TCP flow in IEEE 802.16 networks. Under this rationale, we propose a bidirectional connection, where the bandwidth allocation for UL TCP ACK is associated with the transmission of DL TCP data. As a naive approach, we can consider a *proactive bandwidth allocation* (we call this as `proaction`) mechanism. The key idea is to allocate a dedicated bandwidth to transmit an UL ACK packet without request from SS so as to reduce both delay and overhead involved in the aforementioned bandwidth request. Whenever a DL TCP data packet is served, the scheduler in BS allocates bandwidth for the corresponding UL ACK packet in a proactive manner. With this method, `proaction` can remove bandwidth-request delay for UL TCP ACK packets, and hence, it can increase the throughput of DL TCP flow. This is similar to the ideas of [8] and [10]. The bandwidth allocation mechanism proposed in [8] can remove collision of `BR_RNG` code by allocating resource for bandwidth-request MAC header or for TCP ACK. In a similar way, to avoid unnecessary contention for TCP ACK packet transmission, the mechanism in [10] reserves the shared channel for TCP ACK in contention-based channel access mechanism (i.e., IEEE 802.11 distributed coordination function).

However, these naive approaches have two major drawbacks. Firstly, it wastes UL bandwidth if a delayed ACK mechanism [9] is used. The delayed ACK mechanism, which is implemented in most TCP protocols, lets the TCP receiver not send ACK packets immediately after receiving TCP data packets but wait for the arrival of next in-order TCP data packet up to 500 ms. Combined with cumulative ACK, the delayed ACK mechanism can effectively reduce the amount of ACK traffic. Roughly speaking, the receiver sends every other ACK packet, instead of every ACK packet. Note that the scheduler in the MAC layer is usually unaware of the TCP sequence number in the TCP header, which is impossible in some cases where MSDU is encrypted or compressed. Consequently, in the case of delayed ACK, `proaction` may allocate bandwidth unnecessarily, which decreases UL bandwidth efficiency significantly. Secondly, `proaction` cannot determine the accurate amount of bandwidth-request if MSDU (TCP data packet or ACK packet) is fragmented or packed. In IEEE 802.16 MAC, MSDU fragmentation and packing happen frequently in the process of scheduling and automatic repeat request (ARQ) operation. If BS serves the fragmented or packed TCP data packets, it can hardly predict the amount of bandwidth required for SS to serve the corresponding ACK packets. For these reasons, the `proaction` mechanism cannot be a practical solution for bidirectional bandwidth allocation.

TABLE I
IEEE 802.16E OFDMA PHY PARAMETERS AND THEIR VALUES USED IN THE SIMULATIONS.

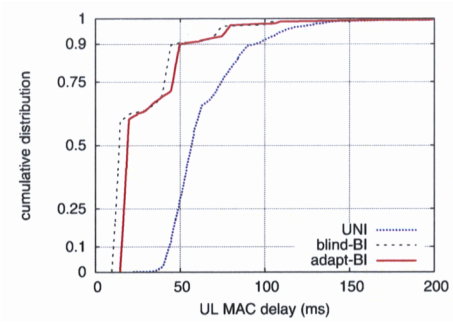
parameter	value
base frequency	2.5 GHz
channel bandwidth	10 MHz
TDD frame duration	5 ms
cyclic prefix duration	11.42 μsec
basic symbol duration	91.43 μsec
Fast Fourier Transform size	1024
number of symbols	29/18 (DL/UL)

To overcome these drawbacks, we propose a hybrid approach combining `proaction` with `piggyback`. This approach basically employs `piggyback`, and utilizes `proaction` only if `piggyback` is not available. Note that `piggyback` does not have any problem involved to the delayed ACK mechanism and fragmentation/packing since it is done in a reactive way (SS first determines the amount of bandwidth required and then it requests the bandwidth). The philosophy beneath the proposed approach is to first minimize the usage of `proaction` for efficient usage in UL bandwidth, and to then replace `contention` with `proaction` for throughput enhancement of the DL TCP flow. The key point is how the scheduler in BS can determine whether the SS can make use of `piggyback` or not.

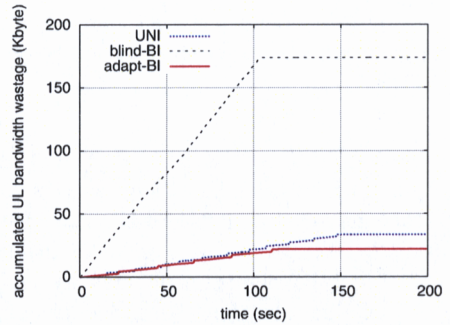
Fig. 3 presents how the proposed approach works. Note the UL bandwidth-request queue in BS is managed for each SS. On serving a DL frame, as shown in Fig. 3, the scheduler in BS checks the corresponding UL request queue. If the request queue is empty, the scheduler puts a new bandwidth-request to it on behalf of the SS, i.e., the scheduler performs `proaction`. The amount of bandwidth-request is equal to the size of a TCP ACK packet including IP and MAC headers. On the other hand, when transmitting an UL frame, SS checks whether there remains backlogged traffic in its transmission queue. If it is not empty, SS requests bandwidth in a piggyback manner at the amount of backlogged traffic. The strong point of this approach is that it neither requires information about the transmission queue of SS in the BS nor results in additional signaling overhead between BS and SS. Therefore, it does not require any modification in SS and can be simply implemented in BS. Moreover, this approach maintains backward compatibility. In this way, the proposed hybrid approach increases the bandwidth efficiency compared to the `proaction` mechanism, and decreases the bandwidth-request delay compared to the `contention` mechanism.

IV. SIMULATION

In this section, we evaluate the performance of the proposed approach with OPNET simulator [2] from the perspective of MAC-to-MAC delay, uplink bandwidth efficiency, and downlink throughput. We compare performance of the following three algorithms; (*i*) `UNI`, a conventional unidirectional approach using `contention` and `piggyback` as described in Sec. II, (*ii*) `blind-BI`, a bidirectional approach using `proaction` without `piggyback`, (*iii*) `adapt-BI`, the proposed bidirectional approach switching between `proaction` and `piggyback` adaptively. The OFDMA physical (PHY) layer parameters and their values are listed in Table I. We model the channel with path-loss, multi-path fading, shadowing, and inter-cell interference and implement adaptive modulation and coding scheme. Also, we emulate hybrid ARQ (HARQ) and ARQ mechanisms such that the target packet error rate of HARQ is 1 %, the maximum number of HARQ retransmissions (excluding initial transmission) is 3, the retransmission delays of HARQ and ARQ are 30 ms and 100 ms, respectively. Here, HARQ and ARQ are modeled so that MSDUs are delivered in order. The bandwidth-request ranging delay is uniformly set between 25 ms to 50 ms and 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 use the delayed ACK mechanism. Also, we consider two types of TCP traffic; FTP traffic and web traffic as modeled in [11]. These configurations are typical operation scenarios for TCP flows. Unless otherwise stated, the simulation configuration is identical to that recommended by IEEE 802.16m task group [11].



(a) Cumulative distribution of uplink MAC-to-MAC delay

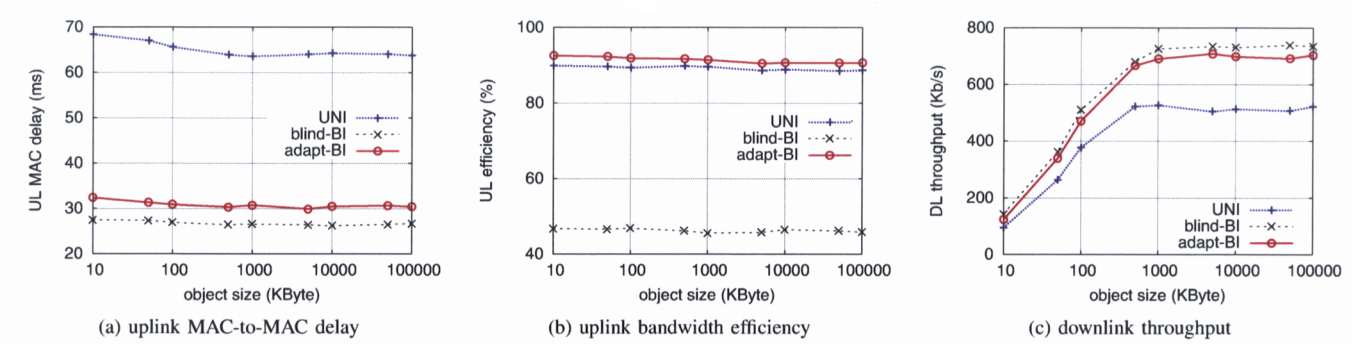


(b) Accumulated uplink bandwidth wastage

Fig. 4. Performance comparison of `UNI`, `blind-BI`, and `adapt-BI` in the case of FTP traffic (10-MB file download).

TABLE II
PERFORMANCE COMPARISON OF `UNI`, `BLIND-BI`, AND `ADAPT-BI` IN THE CASES OF FTP TRAFFIC AND WEB TRAFFIC .

Traffic scenario	Performance index	Algorithm		
		`UNI`	`blind-BI`	`adapt-BI`
FTP traffic	UL delay (ms)	63.9	27.0	31.1
	UL efficiency (%)	88.5	46.3	90.7
	DL throughput (Kb/s)	536.0	769.2	713.5
Web traffic	UL delay (ms)	67.3	27.8	33.0
	UL efficiency (%)	88.0	45.7	91.6
	DL throughput (Kb/s)	98.7	148.3	136.2



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

Fig. 5. Performance comparison of `UNI`, `blind-BI`, and `adapt-BI` for various values of download object size.

First, we observe UL MAC-to-MAC delay, defined as the time interval between the generation of MSDU on the MS and its reception on the BS. Fig. 4(a) compares the cumulative distribution of delay for `UNI`, `blind-BI`, and `adapt-BI`. As was expected, the delay of `blind-BI` was minimized but that of `UNI` was remarkably increased due to contention-based request. However, `adapt-BI` decreased the delay considerably, its average delay was smaller than that of `UNI` by more than 2 times and is slightly larger than that of `blind-BI`. Specifically, the 90th-percentile delay of `UNI`, `blind-BI`, and `adapt-BI` are 94.1, 46.3, and 49.7 ms, respectively. Next, Fig. 4(b) indicates the accumulated uplink bandwidth wastage, defined as $B_{alloc} - B_{used}$, where B_{alloc} denotes the amount of bandwidth allocated for MPDUs (payload with MAC header and bandwidth-request message) and B_{used} is the amount of bandwidth actually used to transmit MSDUs (TCP ACK with IP header). In the case of `blind-BI`, the amount of bandwidth wastage is about 8 times higher than `adapt-BI` because it allocates bandwidth without considering delayed ACK and MSDU fragmentation/packing. However, `adapt-BI` minimized bandwidth allocation wastage; its wastage was slightly smaller than that of `UNI` because `adapt-BI` does not consume bandwidth to send bandwidth-request message in the `proaction` phase, which should be done in the `contention` phase of `UNI`.

Table II lists several performance metrics of three algorithms in the cases of FTP traffic and web traffic. Compared to `UNI`, `adapt-BI` decreased the average UL MAC-to-MAC delay by more than 2 times and increased the average DL throughput by 30 % or more. Furthermore, `adapt-BI` increased the UL bandwidth efficiency, which is defined as B_{used}/B_{alloc}, by 40% at least, compared to `blind-BI`.

We conduct simulations again with various values of object size, L, ranging from 10 KB to 100 MB. As shown in Fig. 5(a) and Fig. 5(b), the effects of L on UL delay and efficiency are insignificant. For the entire range of L, `adapt-BI` outperforms `UNI` and `blind-BI` in terms of UL delay and UL efficiency, respectively. The delay of `adapt-BI` is lower than that of `UNI` by 32.8 ms $\sim$ 35.9 ms and the efficiency of `adapt-BI` is higher than that of `blind-BI` by 44.3 % $\sim$ 45.8 %. Fig. 5(c) shows that as L increases, DL throughput increases accordingly and is saturated as long as L exceeds 1 MB. Regardless of throughput saturation, `adapt-BI` gives higher throughput than `UNI` by about 25.3 % $\sim$ 40.3 %. These simulation results confirm the outstanding performance of `adapt-BI` in terms of bandwidth-request delay, uplink bandwidth efficiency, and downlink throughput.

V. Conclusion

In this paper, we have uncovered that DL TCP performance is degraded in IEEE 802.16 wireless networks due to bandwidth-request delay for transmitting UL ACK. To remove the unnecessary bandwidth-request delay and overhead involved in UL ACK, we have proposed the bidirectional bandwidth allocation framework and hybrid approach combining proactive bandwidth allocation with piggyback bandwidth-request schemes. The simulation results have indicated that the proposed hybrid approach significantly increases downlink TCP throughput as well as uplink bandwidth efficiency. As future research, we would like to derive an analytical model of bidirectional bandwidth allocation and analyze the effect of bandwidth-request delay on TCP throughput quantitatively.

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." http://www.opnet.com/WiMax, 2007.

[3] K. Wongthavarawat and A. Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems," *International Journal of Communication systems*, vol. 16, pp. 81–96, February 2003.

[4] C. Cicconetti, L. Lenzini, E. Mingozzi, and C. Eklund, "Quality of service support in IEEE 802.16 networks," *IEEE Network*, vol. 20, pp. 50–55, March/April 2006.

[5] A. Sayenko, O. Alanen, J. Karhula, and T. Hämäläinen, "Ensuring the QoS requirments in 802.16 scheduling," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems (MSWiM)*, pp. 108–117, 2006.

[6] Q. Liu, X. Wang, and G. B. Giannakis, "A cross-layer scheduling algorithm with QoS support in wireless networks," *IEEE Trans. on Vehicular Technology*, vol. 55, pp. 839–847, May 2006.

[7] S. Kim and I. Yeom, "TCP-aware uplink scheduling for IEEE 802.16," *IEEE Communications letters*, vol. 11, pp. 146–148, November 2006.

[8] E. Kim, J. Kim, and K. S. Kim, "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.

[9] R. Braden, "Requirements for Internet hosts – Communication layers," *IETF RFC 1122*, October 1989.

[10] H. T. Wu, S. Duan, and Cheng, "DCF+: An enhancement for reliable transport protocol over WLAN," *Journal of Computer Science and Technology*, vol. 18, pp. 201–209, March 2003.

[11] IEEE 802.16 WG, "Draft IEEE 802.16m evaluation methodlogy," *IEEE 802.16 Standard*, December 2007.