TCP-Aware Uplink Scheduling for IEEE 802.16

Seungwoon Kim and Ikjun Yeom, *Member, IEEE*

***Abstract*— In IEEE 802.16 networks, a bandwidth request-grant mechanism is used to accommodate various QoS requirements of heterogeneous traffic. However, it may not be effective for TCP flows since (a) there is no strict QoS requirement in TCP traffic; and (b) it is difficult to estimate the amount of required bandwidth due to dynamic changes of the sending rate. In this letter, we propose a new uplink scheduling scheme for best-effort TCP traffic in IEEE 802.16 networks. The proposed scheme does not need any bandwidth request process for allocation. Instead, it estimates the amount of bandwidth required for a flow based on its current sending rate. Through simulation, we show that the proposed scheme is effective to allocate bandwidth for TCP flows.**

***Index Terms*— IEEE 802.16, WMAN, TCP scheduling.**

I. Introduction

IN IEEE 802.16 networks [1], a bandwidth request-grant mechanism is used to accommodate various QoS requirements of heterogeneous traffic such as legacy voice, VoIP (Voice over IP), and Internet data traffic. When a subscriber station (SS) wants to send data, it first needs to send a bandwidth request message to the corresponding base station (BS). Upon receiving the request, the BS grants an appropriate amount of bandwidth to the SS based on an uplink scheduling scheme. There are four service classes defined based on their bandwidth request-grant mechanisms as follows: UGS (Unsolicited Grant Service), RTPS (Real-Time Polling Service), NRTPS (Non Real-Time Polling Service) and BE (Best-Effort). Among them, BE class is allowed to use only contention-based request, that is, there are several shared slots for bandwidth request, and each BE connection contends for sending its request to the BS via the shared slots.

The request-grant mechanism would be effective for QoS sensitive traffic such as real-time multimedia data. However, it may be unnecessary cost for BE TCP traffic in the sense that (a) it needs additional uplink bandwidth for request. As the number of connections in a network increases, the amount of bandwidth for request also increases to resolve request collision; (b) it may also increase latency due to repeated request collision when the bandwidth for request is not enough; and (c) it is hard for each SS to estimate the amount of bandwidth required for its TCP connection due to dynamic changes of the sending rate. A recent study in [2] has addressed a similar observation such that delay and throughput of BE traffic in IEEE 802.16 networks are highly dependent on the offered load due to the bandwidth-request mechanism. To avoid those complexities, in *WiBro* service in South Korea [3] (which is the first commercial service of IEEE 802.16e [4]), bandwidth is equally assigned to each BE connection with round-robin fashion without the request process.

Manuscript received September 19, 2006. The associate editor coordinating the review of this letter and approving it for publication was Dr. Brahim Bensaus. This work was supported by Korea Research Foundation grant KRF 2005-003-D00212.

The authors are with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, South Korea (e-mail: swkim@cnlab.kaist.ac.kr; yeom@cs.kaist.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2007.061509.

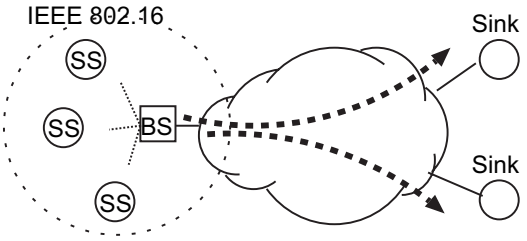

Fig. 1. IEEE 802.16 network.

A simple way for bandwidth allocation without request is to allocate a fixed equal amount of bandwidth to each connection as in *WiBro* service. However, a fixed amount of bandwidth allocation may cause bandwidth wastage due to TCP's variable sending rate. The sending rate of a TCP flow is changed over time due to the AIMD (Additive Increase Multiplicative Decrease) feature in a short-term period and also due to changes of the available bandwidth in a long-term period. Buffering at SSs may be helpful to mitigate short-term oscillations of the sending rate. When a flow maintains its sending rate constantly less than the allocated bandwidth due to external congestion, however, the network observes under-utilization while other flows may suffer from the limited bandwidth of the access link.

In this letter, we propose a scheduling scheme for BE-TCP flows in a IEEE 802.16 uplink. The objective of the proposed scheme is to realize the max-min fairness in bandwidth allocation among them while maintaining high link utilization. The proposed scheme does not need any bandwidth request for grant. Instead, it measures the sending rate of each flow and allocates bandwidth based on the measured sending rate. To evaluate performance of the proposed scheme, we have implemented ns-2 [5] modules for IEEE 802.16 and present extensive simulation.[1] The results show that the proposed scheduling scheme achieves high link utilization without bandwidth request and also effectively deals with dynamic changes of TCP's sending rate.

II. The Proposed Scheme

The network architecture we consider in this letter is illustrated in Fig. 1. SSs are traffic sources and connected to a BS via IEEE 802.16. Traffic is delivered from a SS to the corresponding sink through the BS and the Internet. The proposed scheduling scheme is deployed in the BS and allocates uplink bandwidth to each SS.

To realize the max-min fairness, the scheduler needs to know the demand of each flow. In this letter, we define the demand of a flow as the amount of access link (here

[1]Our modules for IEEE 802.16 are available via http://cnlab.kaist.ac.kr.

IEEE 802.16 link) bandwidth requested for achieving its maximum throughput so as not to be limited by the access link bandwidth.

The proposed scheme estimates the demand of each flow from its sending rate. It is simple to estimate the demand of a flow when the sending rate is measured less than the allocated bandwidth. In this case, the flow observes external congestion or bottleneck links out of the access link, and thus the demand is simply equal to the sending rate.

When the sending rate is equal to the allocated bandwidth, however, it is not straightforward to estimate the demand of the flow from its sending rate since it is hard to distinguish the following two cases: (a) the maximum throughput of the flow is equal to the amount of the current allocated bandwidth and is already achieved; or (b) the access link is the bottleneck of the path currently, and the sending rate is limited by the amount of the current allocated bandwidth.

To distinguish the two cases, in the proposed scheme, the amount of allocated bandwidth is maintained to be slightly higher than the current sending rate. Then, we can expect that the sending rate will be maintained stably in case of (a) whereas it will increase to reach the maximum in case of (b). As a result, the proposed scheme can estimate the demand of each flow as either the current sending rate when it is less than the allocated bandwidth or an amount of bandwidth higher than the current allocated bandwidth when the flow fully utilizes the current bandwidth.

When the demand is expected to be higher than the current bandwidth, it is hard to estimate the exact amount of it. The proposed scheme adaptively increases the bandwidth until the sending rate becomes stable. In Algorithm 1, we present a scheduling algorithm for the proposed scheme.

The proposed scheme adjusts bandwidth allocation whenever detecting any change of flows' demand. To detect the change, it measures the short-term sending rate of each flow and maintains the maximum and the minimum values of it. When the current short-term sending rate is detected to be out of the range between the minimum and the maximum values, bandwidth adjustment is triggered in Line 2-5. To resize the range, the maximum and the minimum values are periodically reset in Line 6-8.

Bandwidth adjustment consists of demand estimation and max-min fair scheduling. The demand estimation procedure is performed as described earlier in this section. Throughout the procedure, note that we use the long-term sending rate for demand estimation rather than the short-term sending rate to avoid frequent fluctuations.

In Line 11-12, for flows with $l[i] \geq B/n$, we set $d[i]$ to be slightly higher ($1/P_1$ times where $P_1 < 1$) than $l[i]$ to provide room for increasing their sending rate even though their sending rate already exceeds the equal share. Note that since the actual bandwidth allocation is performed via the max-min fair scheduling after demand estimation, a demand higher than the equal share does not impact on the bandwidth allocation of flows with a lower demand.

In Line 13, for flows with $l[i] < B/n$, we check if a flow is increasing its sending rate. In the scheme, since $l[i]$ is maintained to be around $P_1 b[i]$ normally, we consider that the sending rate of the flow is increasing when $l[i] \geq P_2 b[i]$ where $P_2 > P_1$. Then, the demand of the flow is set to be higher than $l[i]$ until reaching the equal share (refer to Line 14-16). The increasing rate is determined by r and inc. When $r = 1$, the increasing rate is fixed as the 10% of the sending rate. When $r > 1$, the rate exponentially increases as more increment events happen within $freeze\_time$.

In Line 18-22, we set the demand of a flow with $l[i] < P_2 b[i]$. We first check if $d[i]$ has increased in Line 18 within $freeze\_time$. If so, $d[i]$ is not changed in Line 21. Otherwise, $d[i]$ is set to $l[i]/P_1$ as in Line 12. A TCP flow usually takes several RTTs to inflate its sending rate,and *freeze_time* prevents the increased bandwidth from immediately being reduced.

Once we complete the demand estimation for each flow, the max-min fair scheduling is performed based on the demand. Any algorithm for the max-min fair scheduling such as in [6] can be applicable for the proposed scheduling, and we do not present it here due to the space limitation.

Algorithm 1 Uplink scheduling for IEEE 802.16

```
 : Upon receiving a packet from i^th flow
 1: update s[i] and l[i]
 2: if s[i] > max[i] or < min[i] then
 3:    update max[i] or min[i] with s[i]
 4:    last_update = now
 5:    do Demand Estimation and Max-min Fair Scheduling
 6: else if now > last_update + timeout then
 7:    max[i] = min[i] = s[i]
 8:    do Demand Estimation and Max-min Fair Scheduling
 9: end if
 : Demand Estimation
10: for i = 1 to n do
11:    if l[i] ≥ B/n then
12:       d[i] = l[i]/P1; inc = −1
13:    else if l[i] ≥ P2 × b[i] then
14:       if now > t[i] + freeze_time then inc = 0
15:       else inc++
16:       d[i] = l[i] + 0.1l[i]r^inc; t[i] = now
17:    else
18:       if now > t[i] + freeze_time then
19:          d[i] = l[i]/P1; t[i] = −freeze_time; inc = −1
20:       else
21:          d[i] = b[i]
22:       end if
23:    end if
24: end for
25: if R = Σ d[i] < B then d[i] += R/n for i = 1 to n
26: s[i] and l[i]: short and long-term sending rates of the i^th flow
27: d[i] and b[i]: demand and allocated bandwidth of the i^th flow
28: t[i]: the last time of increasing b[i]
29: B: the total amount of bandwidth for allocation
30: P1 and P2: thresholds for estimating d[i], P1 < P2
31: n: the number of flows for allocation
32: r: the increasing rate (usually 1, 2, or 4)
33: inc: the number of consecutive increases within a freeze_time
```

III. Performance Evaluation

To evaluate the proposed scheme, we have implemented ns-2 [5] modules for IEEE 802.16 networks. TDM (Time Division Multiplexing) is employed for the MAC (Media Access Control) protocol, and the BS allocates slots to each SS.

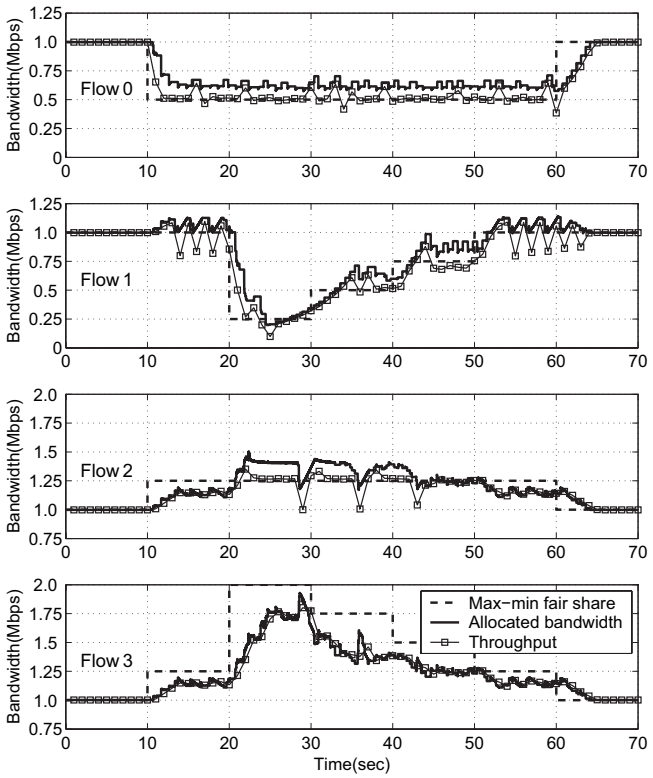

Fig. 2. Allocated bandwidth and throughput.

The proposed scheme is employed in the BS with $\{P_1, P_2, freeze\_time, r\} = \{0.85, 0.9, 0.5, 2\}$.[2] To measure long-term sending rate, we use TSW (Time Sliding Window) [7] with window length = 1 second. The simulation topology is the same as in Fig. 1. There are four SSs connected to a BS, and each SS has one TCP flow. Those four TCP flows share a 4 Mbps IEEE 802.16 link. To make each flow observe different network condition, each flow is transmitted through different wired links from the BS to the corresponding sink host. In Table I, we present the simulation scenario.

In Fig. 2, we present allocated bandwidth and throughput of each flow. We also present reference throughput calculated by the max-min fair algorithm. In the figure, throughput of Flow 0 decreases to 0.5 Mbps at 10 second, and allocated bandwidth also decreases to around 0.6 Mbps. At 60 second, we remove the background traffic, and throughput and allocated bandwidth are recovered to 1 Mbps. During 10 to 60 second, note that we allocate more bandwidth to Flow 0 than its throughput to attempt to increase the throughput since Flow 0 utilizes less bandwidth than the equal share (1 Mbps in this scenario).

During 10 to 20 second, there is about 0.4 Mbps extra bandwidth from Flow 0, and other flows attempt to utilize it. Throughput of Flow 1, however, is limited by the wired link capacity (1 Mbps) while Flow 2 and 3 can increase their throughput to 1.15 Mbps. When we inject 0.75 Mbps background traffic at 20 second, throughput and the allocation bandwidth of Flow 1 decrease to 0.2 Mbps. Then, as we reduce the background traffic from 30 second, throughput is gradually recovered. During 0 to 20 second, and 50 to 70 second, throughput of Flow 2 and Flow 3 is the same since link capacities of them are both larger than the allocated bandwidth. During 20 to 50 second, however, throughput of Flow 2 is limited by the link capacity, and the allocated bandwidth is slightly larger than that to attempt to realize the max-min fairness (compared to Flow 3) while Flow 3 fully utilizes the allocated bandwidth.

[2]We have performed extensive simulation with various sets of the configurable parameters, but the results are not much impacted by them. We do not present the results here due to the space limitation.

TABLE I
SIMULATION SCENARIO

Flow	Wired link bandwidth	Background traffic (Mbps , start time (second), end time (second))
0	1 Mbps	(0.5, 10, 60)
1	1 Mbps	(0.75, 20, 30), (0.5, 30, 40), (0.25, 40, 50)
2	1.25 Mbps	No other traffic
3	2 Mbps	No other traffic

TABLE II
THROUGHPUT COMPARISON WITH CALCULATION

	Flow 0 (Mbps)	Flow 1 (Mbps)	Flow 2 (Mbps)	Flow 3 (Mbps)	Util. (%)
Calculation	0.64	0.79	1.18	1.39	100
Simulation	0.64	0.77	1.14	1.23	94

In Table II, we present comparison of simulated and calculated throughput. Throughput calculation is performed by averaging the max-min fair share in the figure. For example, throughput of Flow 0 is calculated by $(1 \times 10 + 0.5 \times 50 + 1 \times 10)/70$. It is observed that flows with less throughput get closer to their calculated throughput, and the proposed scheduling scheme realizes the max-min fairness.

IV. CONCLUSION

In this letter, we have proposed an uplink scheduling scheme for BE TCP traffic in IEEE 802.16 networks. The proposed scheme does not need any explicit information from senders for bandwidth allocation. Instead, it measures the current sending rate of each flow and allocates bandwidth based on the rate. Through ns-2 simulation, we have shown that the proposed scheme realizes max-min fairness while maintaining high link utilization.

REFERENCES

[1] IEEE 802.16-2004, "IEEE Standard for Local and Metropolitan Area Networks-Part 16: Air Interface for Fixed Broadband Wireless Access Systems," Oct. 2004.
[2] C. Cicconetti *et al.*, "Quality of service support in IEEE 802.16 networks," *IEEE Network*, vol. 20, no. 2, pp. 50-55, Mar./Apr. 2006.
[3] WiBro: Wireless Broadband, available via http://www.wibro.or.kr/
[4] IEEE 802.16e Task Group, "Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands," IEEE Std 802.16e-2005, Feb. 2006.
[5] L. Breslaau *et al.*, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59-67, May 2000.
[6] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[7] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 362-373, Aug. 1998.