Scheduler for IEEE 802.16 Networks

Juliana Freitag Borin and Nelson L. S. da Fonseca

***Abstract*— The IEEE 802.16 standard was designed to support real-time and bandwidth demanding applications with Quality of Service (QoS). Although the standard defines a QoS signaling framework and five service levels, scheduling disciplines for these service levels are unspecified. In this paper, we propose a scheduling scheme for the uplink traffic which is fully standard-compliant and can be easily implemented in the base station. Simulation results show that this scheme is able to meet the QoS requirements of the service flows.**

***Index Terms*— Wireless networks, 802.16 networks, quality of service, scheduling.**

I. Introduction

TO support a wide variety of multimedia applications, the IEEE 802.16 standard [1] defines five types of service flows, each one with different QoS requirements and an uplink scheduling mechanism. The first, Unsolicited Grant Service (UGS), periodically receives fixed size grants without the need to request them. The second, real-time Polling Service (rtPS), provides periodic unicast request opportunities to subscribers; these opportunities ensure delay bound and minimum bandwidth guarantees. The extended real time Polling Service (ertPS) uses a grant mechanism similar to the one used to support UGS connections, except that periodically allocated grants can be used to send bandwidth requests to inform the required grant size. The fourth, non-real-time Polling Service (nrtPS), offers periodic unicast request opportunities, but using more spaced intervals than rtPS, as well as minimum bandwidth guarantee. The final type of service flow, Best Effort (BE), shares contention request opportunities with the nrtPS service flow.

Although these five service levels furnish the basis for QoS provisioning, the core of the task resides in resource allocation, i.e., the scheduling mechanism. An efficient scheduling algorithm is fundamental for the support of QoS requirements, and this has a great influence on network performance.

In this paper, we introduce a BS uplink scheduling algorithm which allocates bandwidth to the SSs. The proposed scheme is fully standard-compliant and it can be easily implemented at the BS. Results obtained through simulation experiments using the ns-2 tool [15] show that the proposed scheme is able to support QoS guarantees of the standardized service classes.

In contrast to previous work [4], [5], [11], the proposed scheme guarantees a delay bound, in addition to minimum bandwidth requirements. The proposals in [3] and [9] also consider these requirements, but they introduce complex scheduling schemes, composed of hierarchies of known schedulers, such as Earliest Deadline First (EDF), Deficit Round Robin (DRR), and Weighted Fair Queueing (WFQ). Simpler solutions are desirable, since the scheduler executes at every frame, which in OFDM-based systems can occur at a frequency of 400 frames per second [1]. On the other hand, other papers [12], [13] have considered either only real-time traffic or only best-effort TCP traffic [14]. The mechanism proposed here supports all five service types defined by the standard. To our knowledge, no other fully standard-compliant mechanism has so far been proposed.

Manuscript received December 15, 2007. The associate editor coordinating the review of this letter and approving it for publication was E. Baccarelli. This research was sponsored by UOL (www.uol.com.br), through its UOL Bolsa Pesquisa program, process number 20060511022200a, and by CNPq, process number 305076/2003-5.

J. F. Borin and N. L. S. da Fonseca are with the Institute of Computing, State University of Campinas (e-mail: {juliana, nfonseca}@ic.unicamp.br).

Digital Object Identifier 10.1109/LCOMM.2008.072110.

II. Proposed Scheduling

The proposed BS uplink scheduler uses three queues, each with a different priority level: low, intermediate and high priority queues. The scheduler serves the queues according to their level of priority. The low priority queue stores BE bandwidth requests. The intermediate queue stores bandwidth requests sent by both rtPS and nrtPS connections. These requests can migrate to the high priority queue to guarantee that their QoS requirements are met. In addition to the requests migrating from the intermediate queue, the high priority queue stores periodic grants and unicast request opportunities that must be scheduled in the following frame. The BS executes the uplink scheduler at every frame, and it broadcasts the scheduling agenda to the SSs in the UL-MAP message.

In each frame, the scheduler generates periodic grants and inserts them into the high priority queue at predefined intervals. The intervals between UGS grants and between ertPS grants are specified at the connection times by SSs. The intervals between request opportunities are defined by the BS. In this way, UGS and ertPS grants are guaranteed, and rtPS and nrtPS unicast request opportunities are provided as specified by the standard. To guarantee delay bound requirement, the BS assigns a deadline for each rtPS bandwidth request in the intermediate queue. Each time the scheduler is executed, the requests with a deadline (minus an adjustable offset δ value) expiring two frames ahead migrate from the intermediate queue to the high priority queue. To determine the request deadline, it is necessary to know the arrival time of the packets at SS queues. Since the BS has no access to this information, it considers the packet arriving in the queue immediately after the last bandwidth request of that connection. Hence, the deadline of a request is equal to the sum of the arrival time of the last request sent by the corresponding connection and the maximum delay requirement of that connection. The value of the offset δ can be adjusted to guarantee that delay requirements are met. If packet deadlines are missed, the offset

ALGORITHM *Scheduling*

1. insert, in the high priority queue, the periodic data grants and unicast request opportunities that must be scheduled in the next frame;
2. CheckDeadline;
3. CheckMinimumBandwidth;
4. schedule the requests in the high priority queue starting from the head of the queue;
5. if intermediate queue is empty and available_slots > 0 then schedule the requests in the low priority queue starting from the head of the queue;

CheckDeadline:
6. for each request i in the intermediate queue do
7. if service[CID] == rtPS then
8. frame[i] = $\lfloor((deadline[i] - \delta) - current\_time) \div frame\_duration\rfloor$;
9. if frame[i] == 3 then
10. if available_bytes $\geq$ BR[i] then
11. migrate request i to high priority queue;
12. granted_BW[CID] = granted_BW[CID] + BR[i];
13. backlogged[CID] = backlogged[CID] $-$ BR[i];
14. available_bytes = available_bytes $-$ BR[i];

CheckMinimumBandwidth:
15. for each connection CID of type rtPS or nrtPS do
16. backlogged_temp[CID] = backlogged[CID];
17. granted_BW_temp[CID] = granted_BW[CID];
18. for each request i in the intermediate queue do
19. if BWmin[CID] $\leq$ granted_BW_temp[CID] then
20. priority[i] = 0;
21. else
22. priority[i] = backlogged_temp[CID] $-$ (granted_BW_temp[CID] $-$ BWmin[CID]);
23. granted_BW_temp[CID] = granted_BW_temp[CID] + BR[i];
24. backlogged_temp[CID] = backlogged_temp[CID] $-$ BR[i];
25. sort the intermediate queue;
26. for each request i in the intermediate queue do
27. if available_bytes $\geq$ BR[i] then
28. migrate request to the high priority queue;
29. granted_BW[CID] = granted_BW[CID] + BR[i];
30. backlogged[CID] = backlogged[CID] $-$ BR[i];
31. available_bytes = available_bytes $-$ BR[i];

is increased; it is decreased as long as packet deadlines are met over a period of time. The dynamic tuning of this parameter is currently under investigation.

This scheduler guarantees the minimum bandwidth requirement of rtPS and nrtPS traffic over a window of duration T. Every time the scheduler is executed, it calculates a priority value for each request in the intermediate queue, considering the following information about the connection: minimum bandwidth requirement, backlogged requests (in bytes), and amount of received bandwidth in the current window of duration T. Requests of connections which have already received the minimum required bandwidth in the current window are assigned low priority values. For the remaining requests, the lower the bandwidth received by the connection, the higher the priority of its requests.

The Algorithm *Scheduling* presents the scheduling scheme. After inserting the periodic grants in the high priority queue, the algorithm checks which rtPS and nrtPS requests should migrate from the intermediate queue to the high priority queue (lines 2 and 3). In line 4, the scheduler serves the high priority queue, and, in line 5, if the intermediate queue is empty and the uplink frame still has available slots, the scheduler allocates bandwidth to the requests in the low priority queue.

In the *CheckDeadline* procedure, for each rtPS request in the intermediate queue, if the request deadline minus δ expires in the frame that follows the next one, and the amount of bandwidth requested by the request (BR[i]) is less than or equal to the number of bytes available [1] in the uplink frame, then the request migrates to the high priority queue. In this case, the algorithm updates the amount of bandwidth allocated to the corresponding connection in the window T (BW_granted[CID] [2]), the number of bytes requested by the backlogged requests sent by the corresponding connection (backlogged[CID]), and the number of available bytes in the uplink frame.

The *CheckMinimumBandwidth* procedure first calculates a priority value for each request in the intermediate queue (lines 15-24). Then, it sorts the intermediate queue according to priority values (line 25). Finally, as long as the uplink frame has available bytes, it migrates requests to the high priority queue and updates the variables BW_granted[CID], backlogged[CID], and the number of bytes available in the uplink frame (lines 26-31).

To derive the expression of the complexity of the mechanism, we introduce the following notation: let k be the number of slots in the uplink subframe, n the number of connections, and r the number of bandwidth requests in the intermediate queue.

In Line 1, the insertion of one periodic data grant or one unicast request opportunity in the high priority queue takes $O(1)$ time. In the worst case, the size of each grant is of one slot, and k grants are inserted into the queue. Therefore, this step takes $O(k)$ time.

In the *CheckDeadline* procedure, each individual step requires $O(1)$ time and they are all executed for all the requests in the intermediate queue (r), thus, the procedure requires $O(r)$ time.

The first loop in the *CheckMinimumBandwidth* procedure (line 6) takes $O(n)$ time, since in the worst case the number of rtPS and nrtPS connections is equal to n. The second and the third loop (lines 18 and 26, respectively) are executed r times, each requiring $O(r)$ time. Finally, the sort operation (line 25) can be implemented by an $O(rlogr)$ algorithm.

Thus, the time complexity of the proposed algorithm is $O(k + n + rlogr)$.

III. SIMULATION EXPERIMENTS

To conduct this study, an ns-2 module for IEEE 802.16 networks [10] was developed. The purpose of the study was to explore the scheduler capabilities and to evaluate its behavior under ideal channel conditions. In the future, the proposed scheme will take into account impairments of the wireless channel.

The topology of the simulated network consisted of a BS, with the SSs uniformly distributed around it. The frame

[1]The number of available bytes in a frame is equal to the number of available slots multiplied by the number of bytes that can be transmitted in each one.

[2]CID represents the connection identifier.

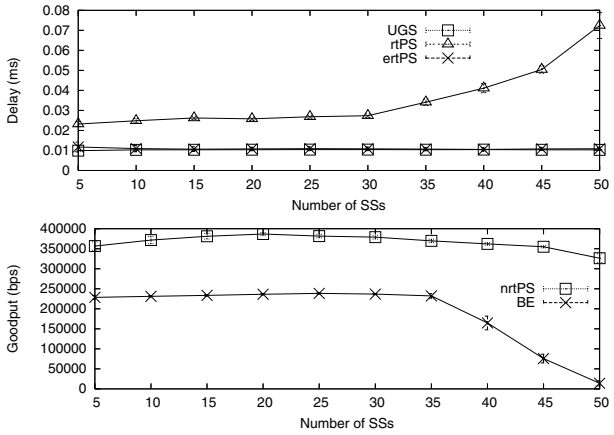

Fig. 1. Delay and goodput of the service flows.

duration was 5 ms, and the capacity of the channel was 40 Mbps, assuming a 1:1 downlink-to-uplink TDD split.

To prevent packet scheduling in the SSs affecting the results, each SS has only one 802.16 service flow. We consider five types of traffic: voice, voice with silence suppression, video, FTP and WEB, which are associated to UGS, ertPS, rtPS, nrtPS, and BE services, respectively.

The voice model used was an exponential "on/off" model with a mean of 1.2 s and 1.8 s for "on" and "off" periods, respectively. During "on" periods, 66-byte packets are generated at fixed 20-ms intervals [6]. The voice with silence suppression model used the Enhanced Variable Rate Codec (EVRC) [2], with packets generated every 20 ms using either Rate 1 (171 bits/packet), Rate 1/2 (80 bits/packet), Rate 1/4 (40 bits/packet) or Rate 1/8 (16 bits/packet). Video traffic was generated by real MPEG traces [8]. The WEB traffic was modeled by a hybrid Lognormal/Pareto distribution, with the body of the distribution corresponding to an area of 0.88 modeled by a Lognormal distribution with a mean of 7247 bytes and the tail modeled by a Pareto distribution with mean of 10558 bytes [7]. FTP traffic was generated using an exponential distribution with a mean of 512 KBytes.

The interval between data grants for the UGS service and for the ertPS service is 20 ms. The interval between unicast request opportunities of the rtPS service is 20 ms and the interval of the nrtPS service 1 s.

For rtPS service, the delay requirement is 100 ms and each connection has its own minimum bandwidth requirement which varies according to the mean rate of the transmitted video. The nrtPS service has minimum bandwidth requirement of 200Kbps, and the BE service does not have any QoS requirement.

In the simulation experiments, the number of SSs increased from 5 to 50 in steps of 5 units (one for each type of service). Each simulation run ten times with different seeds. The mean and the 95% confidence interval are shown in the graph.

Fig. 1 (top graph) shows the average delay of UGS, rtPS, and ertPS uplink connections for each number of SSs. The delay of UGS and ertPS connections was not affected by the load increase, showing that the uplink scheduler is able to provide data grants at fixed intervals as required by these services. Conversely, the delay in rtPS connections increases with offered load, however, it does not surpass the required value. Other simulation experiments, not presented here due to space limitation, indicate that the delay in rtPS connections can be decreased by tuning the value of δ in the scheduling algorithm.

As can be seen in the bottom graph of Fig. 1, the goodput of the nrtPS connections decreased slightly with the increase in offered load. Nonetheless, all these connections guaranteed the minimum bandwidth requirement. The throughput of BE connections, however, decreases sharply when the system is overloaded, as was expected, since the load increase involves the flow of higher priority services.

IV. Conclusions

In this paper, we have proposed an uplink scheduling mechanism for IEEE 802.16 networks. The proposed solution supports the five service levels specified by the IEEE 802.16 standard [1]. Moreover, it considers their QoS requirements in scheduling decisions. It uses a simple approach based on three priority queues and the scheduling algorithm does not require extensive calculations.

The simulation experiments have shown the efficacy of the proposed scheme. Even if there are connections of different service levels in the network, the scheduler will still allocate enough slots to each connection so that the QoS requirements will be met.

References

[1] IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std., Rev. IEEE Std802.16-2004, 2004.

[2] 3GPP2 C.S0014-0, Enhanced Variable Rate Codec (EVRC).

[3] H. Safa *et al.*, "New scheduling architecture for IEEE 802.16 wireless metropolitan area network," in *Proc. IEEE/ACS International Conference on Computer Systems and Applications*, pp. 203-210, 2007.

[4] G. Chu, D. Wang, and S. Mei, "A QoS architecture for the MAC protocol of IEEE 802.16 BWA system," in *Proc. IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions*, pp. 435-439, 2002.

[5] M. Hawa and D. W. Petr, "Quality of service scheduling in cable and broadband wireless access systems," in *Proc. 10th IEEE International Workshop on Quality of Service*, pp. 247-255, 2002.

[6] P. Brady, "A model for generating on-off speech patterns in two-way conversations," *Bell Syst. Techn. J.*, vol. 48, pp. 2445-2472, 1969.

[7] P. Barford *et al.*, "Changes in Web client access patterns: characteristics and caching implications," Technical Report 1998-023, Boston University, 1998.

[8] P. Seeling, M. Reisslein, and B. Kulapala, "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: a tutorial," *IEEE Commun. Surveys and Tutorials*, vol. 6, no. 2, pp. 58-78, 2004.

[9] K. Wongthavarawat and A. Ganz, "IEEE 802.16 based last mile broadband wireless military networks with quality of service support," in *Proc. IEEE MILCOM'03*, pp. 779-784, 2003.

[10] J. Freitag and N. L. S. da Fonseca, "WiMAX module for the ns-2 simulator," in *Proc. IEEE PIMRC'07*, pp. 1-6, 2007.

[11] J. Sun, Y. Yau, and H. Zhu, "Quality of service scheduling for 802.16 broadband wireless access systems," in *Proc. IEEE 63rd Vehicular Technology Conference*, pp. 1221-1225, 2006.

[12] H. Lee, T. Kwon, and D. Cho, "An enhanced uplink scheduling algorithm based on voice activity for VoIP services in IEEE 802.16d/e system," *IEEE Commun. Lett.*, vol. 9, no. 8, pp. 691-693, 2005.

[13] O. Yang and J. Lu, "A new scheduling and CAC scheme for real-time video application in fixed wireless networks," in *Proc. 3rd IEEE CCNC*, pp. 303-307, 2006.

[14] S. Kim and I. Yeom, "TCP-aware uplink scheduling for IEEE 802.16," *IEEE Commun. Lett.*, vol. 11, no. 2, pp. 146-148, 2007.

[15] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/, 2002.