Flexible Resource Allocation in IEEE 802.16 Wireless Metropolitan Area Networks

Spyros A. Xergias, Nikos Passas and Lazaros Merakos

Communication Networks Laboratory
Department of Informatics & Telecommunications
University of Athens
{xergias|passas|merakos}@di.uoa.gr

***Abstract*—IEEE 802.16 is considered today as the most promising technology, to provide extension of wireless broadband communications into metropolitan environments. One of the main design challenges in 802.16 is the introduction of an efficient traffic scheduler that handles resource allocation in the wireless interface. This paper proposes such a scheduler, referred to as the "Frame Registry Tree Scheduler", that aims at providing differentiated treatment to data connections, based on their QoS characteristics. The scheduler uses a tree structure in order to prepare time frame creation and reduce processing needs at the beginning of each time frame. Simulation results show that the proposed scheduler can attain considerable improvement in terms of throughput and packet losses.**

***Index Terms*—IEEE 802.16, traffic scheduling, tree structure**

I. INTRODUCTION

THE evolution in Wireless Local Area Networks (WLANs), mainly through IEEE 802.11, provided flexible and efficient wireless data communications in local area environments. On the other hand, third generation mobile data systems increased the bandwidth provided in wide areas, and extended the set of available services. IEEE 802.16 came to fill the gap between local and wide areas, by introducing an advanced system for metropolitan area networks [1].

One of the main advantages of IEEE 802.16 standard [2] is its ability to support a wide range of data transmissions, with different traffic characteristics and quality of service (QoS) requirements. To do that, it provides a wide set of parameters to allow users to describe their traffic profile and service needs. As usually happens in these cases, the standard does not describe a specific algorithm to use these parameters (i.e., a specific traffic scheduler), leaving that as an implementation differentiator. In the literature so far, only a limited number of such algorithms can be found (e.g., [3], [4]).

In this paper, a traffic scheduling algorithm for IEEE 802.16 networks, referred to as the *"Frame Registry Tree Scheduler"* is proposed and evaluated. The unique characteristic of this scheduler is its ability to prepare in advance the creation of the time frame, and avoid complex processing at the beginning of each frame. To do that, it uses a flexible tree structure that maps the scheduler's decisions.

The paper is organized as follows. Section II presents the main characteristics of IEEE 802.16, and more specifically its QoS capabilities. Section III describes the operation of the proposed scheduler, focusing on the actions required in the Frame Registry Tree. Section IV contains the description of the simulation model used to evaluate the proposed scheduler against a simpler scheduling algorithm. Finally, section V contains the conclusions and plans for future work.

II. IEEE 802.16 BROADBAND ACCESS SYSTEMS

A. Basic Operation

The 802.16 physical layer operates at 10–66 GHz (802.16) and 2–11 GHz (amendment 802.16a [5]) with data rates between 32 and 130 Mbps, depending on the channel frequency width and modulation scheme. The system architecture consists of Base Stations (BSs), each one responsible for a specific area cell, and stationary Sunscriber Stations (SSs). Two operation modes are defined: Point-to-Multipoint (PMP), for communication between the BS and the SSs of its cell, and Mesh mode for direct SS-to-SS communications without the need of a BS. This paper deals only with the PMP mode, leaving Mesh mode for future work. At the PMP mode, each BS regulates all the communication in its cell. The communication path between SS and BS has two directions: uplink (from SS to BS) and downlink (from BS to SS), multiplexed either with Time Division Duplex (TDD) or Frequency Division Duplex (FDD). Transmission parameters, including the modulation and coding schemes, may be adjusted individually for each SS on a frame-by-frame basis.

The scheduler described in this paper applies in the TDD mode. In 802.16, a TDD frame has a fixed duration which may take one of the three values: 0.5, 1 or 2 msec. In any case, the frame is divided into a downlink subframe, and an uplink subframe. The TDD framing is adaptive in that the bandwidth allocated to the downlink versus the uplink direction may vary. Each subframe consists of an integer number of Physical Slots (PSs) representing the minimum portion of allocated bandwidth as shown in Figure 1.

The downlink subframe begins with information necessary for frame synchronization and control. A Frame Start Preamble is used for synchronization and equalization. This is followed by the frame control section, containing the DL-MAP and UL-MAP fields, that state the PSs at which bursts begin in both directions. The following frame portion carries the data, organized into bursts with different burst profiles and therefore different levels of transmission robustness. Each SS

receives and decodes the control information of the downlink direction contained in the DL-MAP and looks for MAC headers indicating data for that SS in the remainder of the downlink subframe.

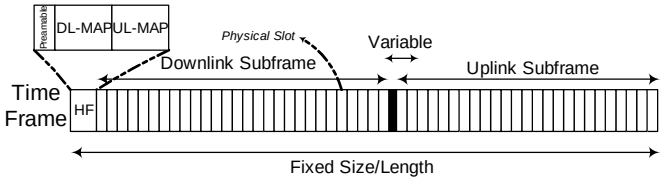

Figure 1. The TDD frame, separated in downlink and uplink subframe.

Through the UL-MAP, the BS determines the transmission opportunities of its subordinates SSs, based on the bandwidth requests of each SS. Bandwidth requests are transmitted through special purpose information elements referred as *BW-Requests*. Each SS having decoded the corresponding control information contained in UL-MAP, knows exactly during which PSs of the uplink subframe it is allowed to transmit and what kind of transmission it can make. Three are the possible kinds of transmission in the uplink:

1. Burst transmission containing variable length MPDUs, or bandwidth requests,
2. Contention-based transmission, where any SS can make requests to the BS, for example requesting more bandwidth,
3. Contention-based transmission of control information, where new-activated or re-activated SSs can initialize their connection.

These different kinds of transmission can occur in any order and any quantity, limited only by the number of the available PSs of the uplink subframe.

In both directions, data bits are randomized, FEC encoded, and mapped to one of the five mandatory (Spread BPSK, BPSK, QPSK, 16-QAM, 64-QAM) or optional 256-QAM signal constellation.

B. Scheduling Services in IEEE 802.16

IEEE 802.16 can support multiple communication services (data, voice, video, etc.) with different QoS requirements. Scheduling services represent the data handling mechanisms supported by the MAC scheduler for data transmission. Each connection is associated with a single data service, and specifies a set of traffic and QoS parameters that quantify its traffic behavior and QoS expectations. The most important parameters, among others, are the following:

1. Minimum Reserved Traffic Rate (in bits/sec)
2. Maximum Sustained Traffic Rate (in bits/sec)
3. Maximum Latency (in ms)
4. SDU Size (in bytes, default=49)
5. Tolerated Jitter (maximum delay variation in ms)
6. Traffic Priority (values 0-7, with 7 the highest)
7. Request/Transmission Policy (values 0-6)

The standard defines four different services: Unsolicited Grant Service (UGS), Real-time Polling Service (rtPS), Non-real-time Polling Service (nrtPS), and Best Effort service (BE). The following text provides a brief description of each of the supported scheduling services, including the mandatory QoS parameters required when a new connection of each service is initiated:

- *Unsolicited Grant Service (UGS)*: This service supports real-time data streams consisting of fixed-size data packets transmitted at periodic intervals, such as Voice over IP without silence suppression. These applications require constant bandwidth allocation, so bandwidth requests are not required and there are no contention opportunities. The mandatory QoS parameters for this scheduling service, based on the previous numbering, are 1, 2, 3, 5, and 7.
- *Real-time Polling Service (rtPS)*: This service supports data streams consisting of variable-sized data packets that are transmitted at fixed intervals, such as MPEG video. These applications have specific bandwidth requirements, as well as a maximum acceptable latency. Late packets that miss the deadline are considered useless. BS provides periodic unicast request opportunities, where each SS can transmit its current queue size. The mandatory QoS parameters for this scheduling service are 1, 2, 3, and 7.
- *Non-real-time Polling Service (nrtPS)*: This service is for non-real-time connections that require better than best effort service, e.g., bandwidth intensive file transfer. These applications are time-insensitive but require a minimum bandwidth allocation. The BS offers unicast polls on a regular basis, which assures that the connection receives a minimum rate of request opportunities even during network congestion. Extra bandwidth is requested through bandwidth requests. The mandatory QoS parameters for this scheduling service are 1, 2, 6, and 7.
- *Best Effort service (BE)*: This service is for best effort traffic with no QoS guarantee. The applications of this kind of service share the remaining bandwidth after allocation to the previous three services is completed. BE uses only contention mode. The mandatory QoS parameters for this scheduling service are 2, 6, and 7.

The traffic scheduler located at the BS decides on the allocation on the PSs in each time frame. In addition to whatever other factors the scheduler may consider, the following items can be taken into account for each active connection:

- the scheduling service specified for the connection,
- the values assigned to the connection's QoS parameters,
- the availability of data for transmission (queue size),
- the capacity of the available bandwidth.

Uplink request/grant scheduling is performed by the BS with the intent of providing each subordinate SS with bandwidth for uplink transmissions or opportunities for bandwidth requests. By specifying a scheduling service and its associated QoS parameters, the BS scheduler can anticipate the throughput and latency needs of the uplink traffic and provide polls and/or grants at the appropriate times.

III. FRAME REGISTRY TREE SCHEDULER

Taking into account the presented scheduling services, as well as the bandwidth request and allocation mechanisms

provided by the standard, it is clear that the required scheduling operation should be performed just before the creation of each time frame, so as to be able to utilize all the available information. This means that even a simple scheduling mechanism requires a minimum number of scheduling tasks, which are translated into considerable computational complexity. Some of the basic actions to be performed by a simple scheduler are the following:

- classification of the connection queues, based on their QoS parameters,
- choice of the packets to be sent,
- ordering of the chosen packets.

A simple scheduling mechanism most probably results in an unfair treatment of lower class services, such as nrtPS and BE in comparison to UGS and rtPS. Even services of the same class can be treated differently, especially during congestion, provided they have different QoS parameters.

The proposed packet scheduler referred to as *"Frame Registry Tree Scheduler"* comes to extend the existing scheduling operations by properly preparing future-transmitted frames. This preparation is performed by distributing the computation complexity during the whole lifetime of each packet. Furthermore, the proposed scheduler provides advanced organization of the transmitted data, facilitating transmission either from the BS to the SSs or from individual SSs to the BS.

Another possible problem that the scheduler has to deal with is the change of one or more basic characteristics of a connection, such as the modulation type of an SS or the service type of the connection. A simple scheduler could hardly treat such changes, increasing packets loss probability, as opposed to the proposed scheduler that conveniently handles such cases, by slightly increasing the computation complexity, and prevents any negative effects in the transmitted frame. For example, if a SS changes its modulation from 256-QAM to 64-QAM, more modulated symbols will be needed for the same data transmission. If there are no empty slots in the current frame, some data should be delayed and may be lost.

A. The scheduler goals

The proposed scheduler is based on a tree structure, referred to as the *Frame Registry Tree*, as shown in Figure 2. All the leaves of this tree belong to the same level and point to the corresponding connection queues where real data are stored. The basic idea is to schedule transmission of each peace of data in the last frame before its deadline. More specifically, the aims of the proposed scheduler are the following:

1. The final time frame should have its transmissions organized in modulation order beginning from BPSK to 256-QAM, in both downlink and uplink subframes (as indicated in [2]).

2. A per QoS service treatment of the transmissions should be possible, based on a specific priority strategy (strict, weighted, portioned, etc.)

3. Data packet transmissions should be based on their deadline (arrival time + latency) and not only on their arrival time.

4. Transmissions should be organized per SS and per connection.

5. The required processing complexity, before the beginning of a time frame transmission, should be minimum.

6. A possible change of a SS's modulation or connection's QoS service should slightly affect the tree structure and add minimum process complexity.

B. Frame Registry Tree Description

The structure in Figure 2 is the heart of the proposed scheduler and is based on the fact that transmissions in both directions are performed through time frames of fixed duration. The tree consists of six levels; root, time frame, modulation, subscriber, QoS service and connection level, and aims at classifying transmissions in a convenient manner, imprinting the whole QoS logic.

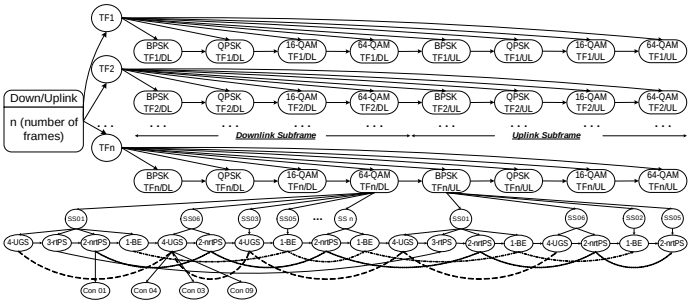

Figure 2. The proposed Frame Registry Tree.

The root, as well as every other node, except from the leaves, can store a number of variables and pointers to their children. For the purposes of the proposed scheduler, these variables include the total number of packets that the connections of each subtree contains, as well as the number of symbols required for their transmission.

The second level represents time frames immediately following the present one, in a sequential order (i.e., TF_1 is immediately after the present frame, TF_2 the next one, etc.). Since the maximum latency parameter of the QoS profile can vary from 0 to 4 bytes (2^{32}), as indicated in the standard, the maximum number of time frames existing in the Frame Registry Tree could be exactly 2^{32}, assuming a frame length of 1ms. But this is a theoretical upper limit; usually some tens of time frames should be considered.

The third level corresponds to the available modulation types. This level can consist of a maximum of twelve nodes, each one representing one of the six possible modulation types (Spread BPSK, BPSK, QPSK, 16-QAM, 64-QAM and 256-QAM) both for uplink and downlink.

In the fourth level, connections are organized per SS. In fact, every SS can have a maximum of two nodes in this level, one for the uplink and one for the downlink, because every SS uses only one modulation each time.

In the next level (fifth), connections are organized according to their QoS service (UGS, rtPS, nrtPS, BE), since

every SS may have many connections, each one belonging to one of the four QoS services.

The last level of the tree structure consists of one leaf for every active connection queue (both for the downlink and the uplink). Every leaf stores one variable and a pointer. The variable's value corresponds to the number of packets stored in the corresponding connection queue and can be transmitted in that time frame, without violating their deadlines. The pointer shows the next leaf and is used so as to facilitate the sweep of the leaves, mainly during the frame creation procedure, which is described in the next subsection.

C. How the scheduler works

The scheduler resides on the BS and its main scope is to decide on the packets to be transmitted and their order in each frame. For every downlink packet arriving into the BS's outgoing queues and every request sent by the SSs for uplink packet transmissions, the required information is inserted into the tree structure, according to the corresponding QoS characteristics (service, modulation, deadline, etc.). The operation of the proposed scheduler can be divided into four main procedures: i) packet/request arrival, ii) creation of next frame for transmission, iii) SS modulation change, and iv) connection QoS service change.

1) Packet /request arrival

The basic idea is to distribute packet transmissions in time frames, according to their deadline, utilizing the fact that frames have fixed duration. Since new packets for the downlink and new requests for packet transmissions from the uplink are treated in the same manner, only the downlink case is described here.

It is assumed that a traffic policing mechanisms is available at the BS (e.g., leaky bucket) to ensure that the incoming traffic is consistent to the connection declarations during setup. There are two possible cases for each packet deadline: i) it can be calculated from the packets' arrival time plus their latency (UGS and rtPS services), and ii) it cannot be calculated because the latency is not given (nrtPS, BE services). In the first case, the subtree of the last time frame that this packet can be transmitted should be updated. In the subtree does not exist at that time (i.e., this is the first scheduled transmission for the specific time frame), it is created. In the second case, the subtree of the last existing time frame, at the time of the packet arrival should be updated.

As an example, let us assume the case of a downlink packet arrival for connection 3 (Con03) that is of type UGS and should be sent to Subscriber Station 6 (SS06), using 64-QAM modulation. If the last frame that this packet can be transmitted, based on its deadline is TF_n, the required path to be updated is TF_n→ 64-QAM → SS06→ UGS→ Con03. If the leaf of the connection that the packet belongs to exists, its counter increases by one. In different case, the required path is created and the leaf counter is set to one. All father nodes of that leaf are updated accordingly.

In the ideal case, the packet/request arrival procedure would result in balanced time frame subtrees, concerning the total number of packets, the downlink/uplink portion and the number of slots left for contention. In different case, some unbalance can be absorbed by the protocols flexibility in terms of downlink/uplink portion and contention slots. Even in heavily unbalanced cases the packet/request arrival procedure does not have to deal with the details of time frame sizes and structure. It simply places packet transmissions to the appropriate time frame subtrees, leaving the frame creation procedure, described later, to decide on the final transmitted frame content and structure.

2) Frame Creation

Before the beginning of each time frame, the frame creation procedure decides on the frame contents and structure based on the information stored in the Frame Registry Tree. Let us assume that at the time instance t_n there are m time frame subtrees in the tree structure (TF_1 … TF_m). The extracted frame will be generated mainly from TF_1 and, in case it has empty slots, they will be fulfilled with packets from the following time frames (TF_2, TF_3, …). The number of slots that the scheduler will provide to each connection depends on the bandwidth distribution policy indicated by the standard, and can be extracted from the Minimum Reserved Traffic Rate and Maximum Sustained Traffic Rate QoS Parameters of each connection. Three cases can be distinguished for TF_1: i) the number of packets of the subtree TF_1 fit exactly into one time frame, ii) the number of packets in the subtree TF_1 are less than the capacity of a time frame and iii) the number of packets in subtree TF_1 are more than the capacity of a time frame.

a) Exactly complete frame

This is the best possible case where the packets for transmission cover exactly the available frame slots. In that case, the frame can be transmitted with no other processing, starting from the first leaf to the last one exactly as they are represented in the last level of the tree. The order of modulation, SS and QoS service with which the frame will be constructed will be the one that the proposed structure depicts.

b) Frame with empty slots

In this case, there are two options for the empty slots: either they will be covered with packets from the next time frame subtrees, or some of them will be left for contention. The exact number of slots that will be fulfilled with packets from the next time frame subtrees depends on a number of factors and needs a specific strategy or algorithm to be followed, which is out of the scope of this paper. It is assumed that this number is known to the scheduler. Many schemes can be used for choosing the packets to fulfill the empty slots. An advanced scheme of choosing packets could be based on a negative exponential percentage. For example, the UGS service could take 50% of the empty slots, rtPS 30%, nrtPS 15% and BE could take the rest 5%.

c) Frame with excess packets for transmission

In this case, if there are BE packets, these can be moved to

the next frame subtree. Using the tree structure, moving is a simple change of pointers and update of variables. If, even after this operation, there are still excess packets for transmission, some of them should be rejected, since it is not possible to be transmitted in a later frame. A number of strategies can apply here, e.g., the greedy or the negative exponential strategy, but is out of the scope of this paper to propose a specific one. In the simulations below, the greedy approach is used, i.e., first all the nrtPS packets are rejected, then all the rtPS, and finally some of the UGS, until the number of packets can exactly fit into one time frame.

3) SS's modulation type change or connection QoS service change

The proposed packet scheduler deals with such changes using a simple process during which every substructure in the tree of a particular SS or connection moves to the right modulation substructure or service substructure. The maximum moves that may be necessary would be two times the numbers of existing time frames. Again, it is noted that moving is a simple operation of pointer changes. In case that such a change occurs just before a frame transmission, the scheduler can decrease the particular substructure moves to those affect only the transmitted frame, finishing the rest of the moves later.

The scheduler operation, as described above, provides continuous transmission of data having the same modulation and coming from or going to the same SS. This reduces data fragmentation and physical overhead due to modulation changes. QoS classification is succeeded by transmitting high priority before lower priority traffic in the same frame. Additionally, the proposed scheduler maintains low computational complexity, without significantly increasing storing requirements. It distributes scheduling process at all phases of a packet's lifetime, instead of gathering all processing needs at the frame creation phase. On the contrary, frame creation is just a simple process of running through the leaves of the next time frames in the tree. Finally, another advantage of this structure is its extension flexibility, based on changes of the standard. For example, it can easily support more QoS services, or more modulation types that may be added in the future.

IV. SIMULATIONS

In order to measure the performance of the proposed scheduler, a simulation program that compares it against a simpler scheduler is constructed. The simple scheduler classifies packets only based on their QoS service, without considering any deadlines or other QoS characteristics. The simulation program was constructed in MATLAB [6] and reflects the full functionality of both the proposed and simple scheduler.

A. Simulation Scenario

A simulation scenario where all SSs had the same modulation (64-QAM) and the same set of connections was considered. More specifically, each SS had the following connections:

- one UGS with constant data rate of 64Kbps (e.g., voice),
- one rtPS with mean data rate of 256Kbps,
- one nrtPS with mean data rate of 128Kbps, and
- one BE with mean data rate of 128Kbps.

The data rate of the physical channel was set to 120Mbps, the time frame length to 1msec and the packet size to 54 bytes. Both algorithms followed the "strict" priority servicing policy, according to which, when the algorithm (simple or FRTS) has to choose between two packets of different service class, UGS has the greatest priority, followed by rtPS, nrtPS and finally BE.

B. Results

Since both algorithms have to serve the same kind of fixed-size packets (because of the same modulation), it is expected that both algorithms will totally serve the same number of packets. This leads to the same data rate. Nevertheless, FRTS is expected to have better throughput performance as a result of less lost packets. This is clearly reflected to the simulation results depicted in Figure 3. The difference between the lost packets by simple scheduler and FRTS corresponds to the difference between the packets being in the memory by the simple and FRTS scheduler.

Four other metrics are expected to differentiate the proposed algorithm compared to the simple scheduler: i)the total packet loss, ii) the packet loss per service class, iii) mean delay per service class and iv) the number of served packets per service class.

As a result of the strict priority policy used by both algorithms, a set of theoretical thresholds can be calculated, where the algorithms are expected to change their behavior:

- Incapable to serve all BE packets, $(BW_{UGS}+BW_{rtPS}+BW_{nrtPS}+BW_{BE})*n > BW_{total}$ <=> (64Kbps+256Kbps+128Kbps+128Kbps)*n>120Mbps <=> (576Kbps)*n>120.000Kbps <=> $n>120*10^3/576$ <=> ***n>208.3***
- Incapable to serve all nrtPS packets and any BE packet, $(BW_{UGS}+BW_{rtPS}+BW_{nrtPS})*n > BW_{total}$ <=> (64Kbps+256Kbps+128Kbps)*n>120Mbps <=> (448Kbps)*n>120.000Kbps <=> $n>120*10^3/448$ <=> ***n>267.9***
- Incapable to serve all rtPS packets and any BE and nrtPS packet, $(BW_{UGS}+BW_{rtPS})*n > BW_{total}$ <=> (64Kbps+256Kbps)*n>120Mbps <=> (320Kbps)*n>120.000Kbps <=> $n>120*10^3/320$ <=> ***n>375***
- Capable to serve only UGS packets, $BW_{UGS}*n > BW_{total}$ <=> (64Kbps)*n>120Mbps <=> $n>120*10^3/64$ <=> ***n>1875***

In Figure 4, 5 and 6 the three of these four points are clearly shown, while the fourth, where UGS packets start not to be served, is omitted for presentation reasons (the results up to 1875 SSs could not be presented in these graphs).

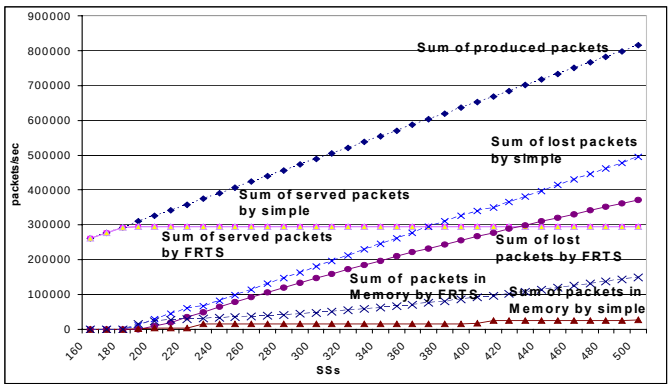

Figure 3. Total produced, served and being in memory packets.

More specifically, in Figure 4 the service percentage of packets per service class is shown. The simple scheduler stops servicing BE packet right after the theoretical limit of 210. On the contrary, FRTS guarantees a minimum bandwidth for BE packets without "steeling" this bandwidth from any other service class. This is because FRTS takes advantage of the latency tolerance of UGS and rtPS service classes. The small period where simple scheduler seems to work better than FRTS for nrtPS service class, is because of packets that are still in memory and have not been serviced yet. The same behavior is repeated for the other three service classes, close to the theoretical numbers of SSs that have been calculated.

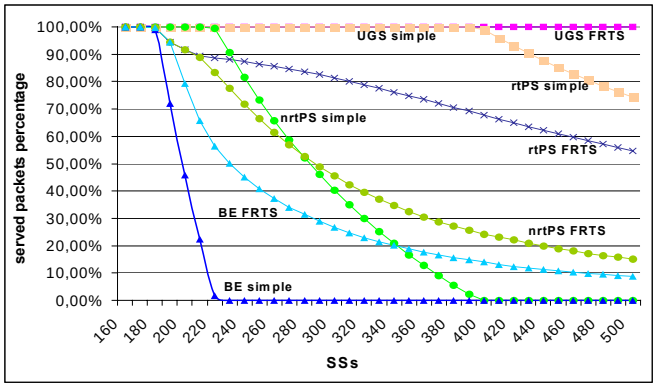

Figure 4. Percentage of served packets per service class.

Figure 4 is supplemented by Figure 5, where the percentage of lost packets per service class is shown. Again it is important to notice that FRTS still services an amount of packets per class, independently from the number of SSs.

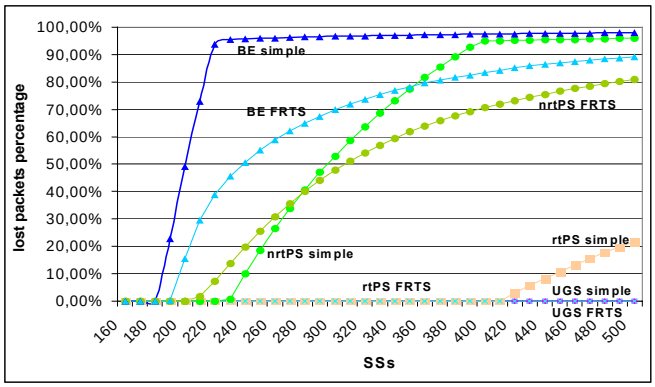

Figure 5. Percentage of lost packets per service class.

Finally, in Figure 6 the mean packet delay per service class is shown. Again, for BE packets, the mean packet delay for simple scheduler becomes almost zero right after 240 SSs, because no packets are served so as to calculate the mean delay. FRTS on the other hand attains a much smoother behavior, as a result of the latency utilization. The same behavior is repeated for the other three service classes, at thresholds, close to the theoretical numbers.

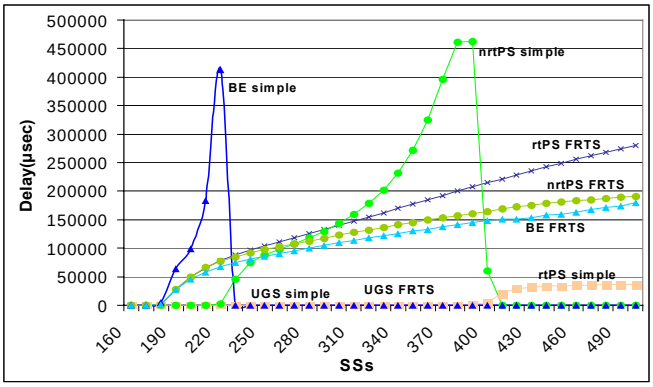

Figure 6. Mean packet delay per service class

V. Conclusions

In this paper, a new traffic scheduling algorithm for IEEE 802.16 networks has been proposed, referred to as the "Frame Registry Tree Scheduler" (FRTS). The unique characteristic of this algorithm is the use of the Frame Registry Tree, a data structure that aims at preparing time frame creation and reducing processing needs at the beginning of each frame. Using this structure, the algorithm schedules each packet at the last time frame before its deadline, in order to allow more packets to be transmitted and increase throughput. Additionally, the scheduler manages to avoid fragmentation of transmissions to/from the same SS or of the same modulation.

Simulation results show that the proposed scheduler can considerably improve packet loss and throughput of different types of traffic, compared to a simpler scheduler. Future plans include time and space evaluation comparison, with more advanced schedulers, and combination with a connection admission control algorithm.

References

[1] C. Eklund, R.B. Marks, K. Stanwood, and S. Wang, "IEEE Standard 802.16: A Technical Overview of the WirelessMAN™ Air Interface for Broadband Wireless Access", IEEE Commun. Magazine, vol. 6, no. 6, pp. 98-107, June 2002.

[2] IEEE Std 802.16-2004, "IEEE Standard for Local and Metropolitan Area Networks – Part 16: Air Interface for Fixed Broadband Access Systems", October 004.

[3] K. Wongthavarawat, and Aura Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems", Int. J. of Commun. Syst., vol. 16, pp. 81-96, 2003.

[4] G. Nair, et al., "IEEE 802.16 Medium Access Control and Service Provisioning", Intel Technology Journal, vol. 8, no. 3, August 2004.

[5] IEEE Std 802.16a, "Amendment 2: Medium Access Control Modifications and Additional Physical Layer Specifications for 2-11 GHz" 2003.

[6] The MathWorks, "*MATLAB User's Guide*", version 7.01, Natick, MA, USA, 2004.