U.P.B. Sci. Bull., Series C, Vol. 70, No. 1, 2008 ISSN 1454-234x

ADAPTIVE PACKET SCHEDULING FOR QoS SUPPORT IN IEEE 802.16 WIRELESS ACCESS SYSTEMS

N. NECULA[1], E. BORCOCI[2]

Articolul prezintă o metodă adaptivă pentru planificarea optimă a accesului la linkul ascendent din standardul IEEE 802.16. Metoda emulează în timp real comportarea unui planificator off-line bazat pe un algoritm genetic. Algoritmul genetic este proiectat astfel încât să coreleze în mod optim cerințele contradictorii de servire prioritară şi servire echitabilă, necesare pentru obținerea calității de servire QoS, definite de standardul IEEE 802.16 pentru toate clasele de serviciu.

An adaptive method is proposed for nearly optimal IEEE 802.16 uplink scheduling. This method emulates in real time the behavior of a Genetic Algorithm-based offline scheduling algorithm that is designed to handle optimally the tradeoff between the priority and fairness, when attempting to meet the Quality of Service requirements defined by the IEEE 802.16 standard for all service classes.

Keywords: IEEE 802.16, QoS, adaptive packet scheduling, genetic algorithms

1. Introduction

The broadband access technology defined by IEEE 802.16 standard [1], and promoted by WiMax Forum, targets to deliver high data rates to a large number of users, especially in rural and developing areas, where there is no available wired infrastructure. It provides high-speed Internet access to home and business subscribers, and it supports quality of service (QoS) for real time applications such as video conferencing, video streaming and voice over IP.

IEEE 802.16 architecture consists of two types of fixed stations: subscriber stations (SS) and a base station (BS). In the point-to-multipoint operational mode (PMP), the SSs communicate only through the BS. The communication path between SSs and BS has two directions: uplink (from SS to BS) and downlink (from BS to SS). Uplink and downlink transmission use time-division multiplexing (TDM) with frames of fixed duration, consisting of a predefined number of time slots. The frames are dynamically subdivided by the

[1] Prof., Dept. of Telecommunications, University “Politehnica” of Bucharest, Romania, e-mail: necula@comm.pub.ro
[2] Prof., Dept. of Telecommunications, University “Politehnica” of Bucharest, Romania, e-mail: eugen.borcoci@elcom.pub.ro

BS into two subframes: one for the downlink transmission and the other one for the uplink transmission. The BS is also responsible for allocating the time slots in the uplink subframes to the subscriber connections, according to their QoS requirements.

While extensive bandwidth allocation and QoS mechanisms are provided, the details of scheduling and reservation management are left un-standardized and provide an important mechanism for vendors to differentiate their equipment.

There are four service classes defined by the 802.16d standard: Unsolicited Grant Service (UGS), real-time Polling Service (rtPS), non-real-time Polling Service (nrtPS) and Best Effort (BE). UGS supports Constant Bit Rate (CBR) services, such as T1/E1 emulation, and Voice over IP (VoIP) without silence suppression; rtPS supports real-time services that generate variable size data packets on a periodic basis, such as MPEG video or VoIP with silence suppression; nrtPS supports non-real-time services that require variable size data grant burst types on a regular basis; and BE services are typically provided by the Internet today for Web surfing. The 802.16e standard has added another class: extended real-time Polling Service (ertPS). This is similar to UGS, but uses dynamic resource allocation, instead of fixed.

IEEE 802.16 media access control, which is based on the concepts of connections and service flows, specifies QoS signaling mechanisms (per connection or per subscriber station) such as bandwidth requests and bandwidth allocation, but the QoS-based packet scheduling algorithms for both uplink and downlink bandwidth allocation are left undefined.

Ideally, in order to maximize the revenue of the system operator, an efficient packet scheduler should be capable of providing the guaranteed service that has been required and paid for accordingly, to as many subscribers as possible.

Several packet scheduling algorithms for broadband wireless networks have been published in the last years, with many of them specifically designed for IEEE 802.16 standard, e.g. [3] and [4].

The simplest way of providing QoS guarantees is to assign different priorities to the different classes of traffic sources, according to their requirements in terms of bandwidth, maximum delay and delay variation, and packet loss probability during the periods of uplink congestion. However, since the priorities are static characteristics assigned to the traffic sources, the scheduler is not capable to adapt its service policy to traffic variation, particularly during simultaneous traffic bursts that may cause unacceptable starvation of the lower priority sources.

The two algorithms mentioned above are both based on static priorities: the first one, described in [3], is a two-level hierarchical algorithm that allocates the bandwidth, in the first level, by following the strict priority order, assigned to the four service classes (without ertPS); the second one, described in [4], is a two-stage algorithm that allocates in the first stage the minimum bandwidth required by each of the five service classes, and, in the second stage, it allocates the unused bandwidth, also by following the strict priority order assigned to these classes. The second level of the algorithm described in [3] allocates the bandwidth within each class of connections, using specific (per class) service disciplines, such as fixed bandwidth for UGS connections, earliest deadline first (EDF) for rtPS connections, weighted fair queuing (WFQ) for nrtPS connections and equal share for BE connections.

Another different approach for bandwidth allocation that can achieve superior performance by monitoring the input traffic variations, and optimally adapting the scheduling algorithm, so that the resource allocation is optimized continuously, subject to service guarantee constraints has also been described in the literature [5], [6].

Optimal resource allocation problems subject to service guarantee constraints have been investigated in various contexts. Specifically, adaptive weighted packet scheduling for premium service in Diff-Serv has been studied in [7], where weights are updated by the estimation of average queue size from exponential weighted moving average.

A Genetic Algorithm (GA)-based adaptive packet scheduler has been described in [8]. It generates nearly optimal time slot allocations once per super-frame, but it is practically impossible to implement it in real-time because of the huge amount of required processing power. However, the behavior of such an offline GA-based scheduler is still worth investigating and emulating in a real-time controller.

This is exactly what this paper is dealing with. The two objectives of the paper are therefore the following:

1. Comparative analysis of the performance obtained for uplink time slot allocation, when using a genetic algorithm vs. conventional algorithms.

2. Real-time emulation of the GA-based algorithm for uplink time slot allocation.

Section 2 describes the offline GA-based implementation of the uplink time slot scheduler. The superior quality of the solution obtained by using a GA, with an appropriately defined fitness function, is then exploited by a process of identification and modeling of the GA-based algorithm behavior.

Modeling can be accomplished either empirically or by using Genetic Programming (GP) for symbolic regression.

The desired behavior is modeled in Section 3 by an empirically determined control function that partitions the set of time slots into two subsets, one for servicing the real-time (RT) sources and the other one for the non-real-time (NRT) sources. This function is then used in a multi-level hierarchical adaptive time slot allocation algorithm.

Simulation results obtained for the Matlab models of different approaches are then illustrated in Section 4 by a number of graphs. These results prove the good or even better quality of the proposed approach, when compared to other more conventional methods.

The final section presents the conclusions and some directions for future work.

2. An offline GA-based scheduler

For the identification and modeling of the behavior of a GA-based scheduler, a basic 802.16 model is assumed, with point to multipoint operation and basic frame organization. Considering one base station and several subscriber stations, a GA has been developed in Matlab for allocating the resources (bandwidth/physical time slots) to several users and services. The time-division multiple access (TDMA) link that connects all subscriber stations to the base station is modeled together with the GA-based control mechanism. Different flows in the four service classes: UGS, rtPS, nrtPS and BE are considered to share the uplink. Since a constant proportion of the link bandwidth is usually reserved for the UGS, the bandwidth allocation for this service class will not be handled by the GA. To simplify the presentation, the frame will be considered as the set of all time slots remaining after the elimination of the reserved UGS time slots.

During every frame, the GA is run to determine the optimal allocation of the time slots in the next frame.

By using an appropriately defined fitness function, the GA is able to allocate nearly optimally all the available time slots, once per frame. The service discipline is therefore not fixed by the class priorities only, but, additionally, it gets dynamically tuned to the current state of all packet queues, thus providing not only priority-based servicing, but also a better fairness for the low priority service classes.

A chromosome, representing a possible per frame time slot allocation, has its fitness defined as in [8]:

$$C_F = \sum_{i=1}^{s} F_i \tag{1}$$

where: C_F = chromosome fitness, s = chromosome length (i.e. the number of time slots per frame), and

F_i = fitness assigned to i*th* time slot = fitness of the traffic source that is being serviced by this time slot, defined below.

$$F_i = \frac{P_i * Q_i}{\sqrt{f_i}} \qquad (2)$$

where: P_i = priority of the traffic source serviced by the i*th* time slot,

Q_i = dynamic queue length of this traffic source, and

f_i = the number of time slots assigned to this traffic source in the current frame.

The GA-based scheduling algorithm is capable of performing better than other algorithms because it adapts dynamically to the current state of all the queues, and it enforces a better service fairness, due to the use of the denominator in (2).

The possibility of real-time implementation of such an algorithm is practically impossible because of the very large amount of computation that needs to be performed by the GA processor in a very short time, normally once per frame. The use of super-frames, called "refreshing frames" is suggested in [8] as a possible way of real-time implementation, by providing a sufficiently long super-frame to allow the complete run of the GA. Unfortunately, the quality of the GA-based solution degrades when the size of the super-frame increases, because the non-stationary characteristics of the incoming traffic, particularly when many bursts are present, cannot be captured and used adequately.

However, the GA-based scheduling algorithm can still provide some useful insights for the development of a real-time adaptive scheduling algorithm, capable to emulate the dynamic behavior of the original GA-based scheduler.

The following steps have been used for identifying such an algorithm, by "reverse engineering" the GA-based solution:

1. First, consider a control function that partitions dynamically the set of time slots per frame into two relevant subsets, e.g. one for real-time (RT) and the other one for non-real-time (NRT) traffic sources:

$$z = F(x,y) \qquad (3)$$

where: z = proportion of time slots per frame allocated to RT sources,

x = total current size of all RT queues, and

y = total current size of all NRT queues.

Queue size is conventionally measured in traffic units, where one traffic unit equals the time slot bandwidth.

This function is defined only in the (x,y) range specified by:

$$x + y > s, \qquad (4)$$

because outside of this range, there are enough time slots per frame to carry all the packets stored in the input queues.

2. Simulate the operation of the time slot scheduler using the GA with fitness function (1), and an appropriate mix of RT and NRT traffic sources, and also calculate and store all per frame triplets $\{x_i,y_i,z_i\}$ for the duration of the

simulation. The resulting three-dimensional array will characterize the dynamic behavior of the GA when handling RT vs. NRT packets.

3. Since the resulting $\{x_i,y_i,z_i\}$ array is very large and z is not uniquely defined for identical (x_i,y_i) pairs, this array needs to be normalized for (x,y) and averaged for z until an n-by-n matrix of z values is obtained, where n is the number of successive value ranges for x and y on a linear or logarithmic scale.

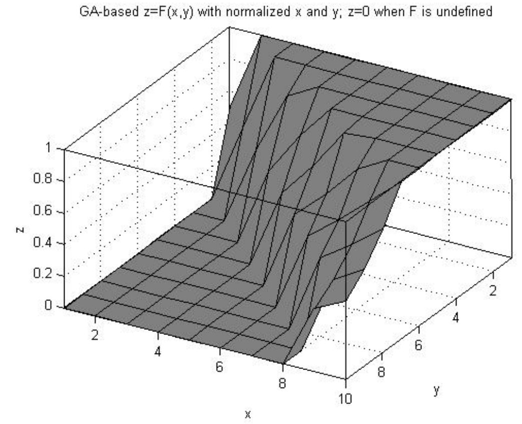

Fig.1. GA-based control function z=F(x,y)

The values of z in the undefined range $x + y \leq s$ are then arbitrarily set to zero, and the unspecified values of z that are still present in the resulting matrix are calculated by interpolation or extrapolation.

Fig. 1 depicts the shape of control function F(x,y).

It can be noticed in this figure that that some fairness is enforced for the NRT traffic even when the RT traffic load is very high, because the proportion of bandwidth allocated to RT traffic is less than 100%, if the NRT traffic load is also very high. This type of behavior results from the optimization performed by the GA by maximizing during every frame the fitness defined by (1). The price paid for this fairness is obviously some degradation in the QoS offered to the RT traffic sources.

3. Proposed adaptive scheduler

The block diagram in Fig.2 illustrates the principle of the proposed algorithm.

It is assumed that he state of the scheduler, consisting of the values O_i for all input queues, is available to the base station at the beginning of each frame. As mentioned in [3], this can be accomplished by sending to BS once per frame the queue size together with the bandwidth request messages for every established connection.

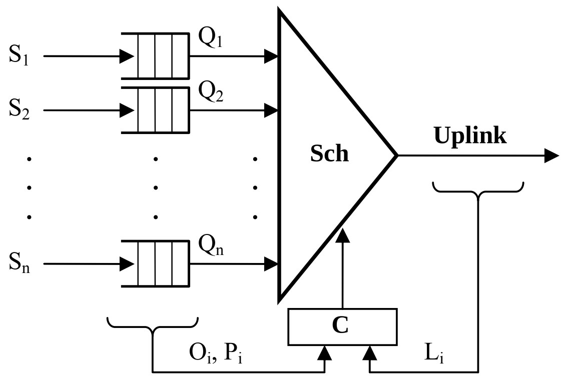

Fig.2. Block diagram of the adaptive scheduler

S_i (i=1,2,...,n) = traffic sources for all SSs
Q_i = per source input FIFO queues
Sch = distributed time slot scheduler (in all SSs)
C = real-time controller (in BS)
O_i = size of Q_i
P_i = priority of S_i
L_i = number of uplink slots allocated to S_i

Per frame multi-level hierarchical bandwidth partitioning among groups of service classes at the top level, and among traffic sources belonging to the same sub-class, at the lowest levels, is a simple and efficient way of implementing the overall adaptive link sharing. A possible such partitioning tree is illustrated in Fig.3.

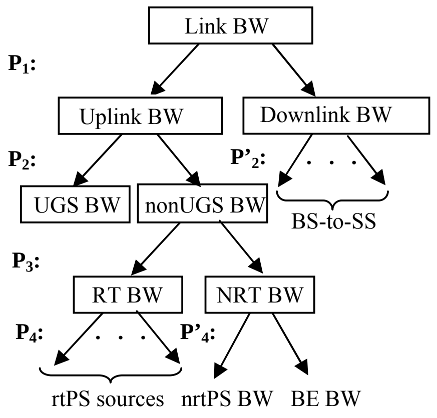

Fig.3. A multi-level hierarchical partitioning of the link bandwidth (BW)

Partition P_1 is adaptive as described in the 802.16 standard, P_3 and P'_4 are adaptive, whereas P_2 is fixed and P_4 could be EDF (earliest deadline first)-based.

Partition P'_2 is not considered in this paper.

Partitioning the available BW into just two partition blocks is advantageous because it is easier to control and simpler to implement. It is also easier to identify the corresponding control function from the behaviour of a GA-based algorithm.

Modeling of function F(x,y) for real-time control of partition P_3 can be achieved in one of the following ways:

- By using an n-by-n lookup table. This works well only when the value of n is small.
- By approximating it with an empirically found algebraic expression.
- By using some form of regression, including symbolic regression of the Genetic Programming [9], to identify an approximate algebraic expression of F(x,y).

A very simple empirical approximation of F(x,y) is given by:

$$\text{IF } (x+y > s) \text{ THEN } z = 1/(1+c*y/x), \text{ with } c < 1 \quad (5)$$

When the logical condition in (5) is not true, the number of available time slots in the current frame can carry all packets waiting in the queues, so z is undefined.

Fig. 4 depicts the shape of the above approximation, where the undefined values of z are arbitrarily set to zero.

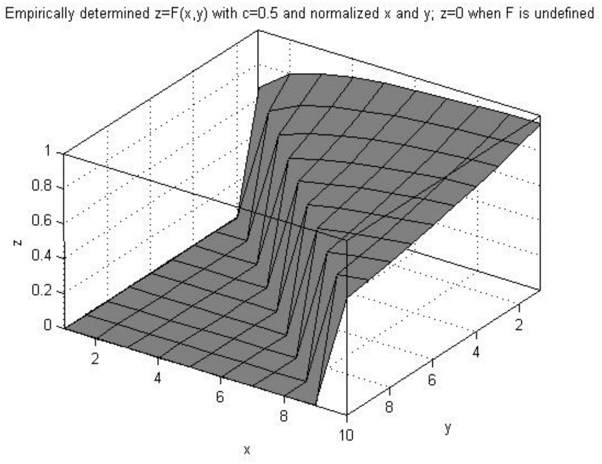

Fig.4. Empirically determined control function z=F(x,y)

The undefined region is specified by $x*n_x+y*n_y \leq s$, where n_x and n_y are the x and y normalization factors.

The two surfaces illustrated in Figs.1 and 4 are acceptably similar. It can be seen that the value of constant c determines the level of fairness that can be achieved when performing partition P_3 using control function (5). For c=1, there is total fairness for the two groups of QoS classes, whereas for values c«1 there is still some fairness in servicing the NRT traffic, as long as $c \neq 0$.

4. Simulation results

A simplified Matlab model of the 802.16 uplink has been implemented, as follows:

- Since the downlink sub-frame has not been considered, the duration of the uplink sub-frame is fixed, instead of being adaptive.
- The polling activities of the BS to collect the SS requests for resources have not been modeled. All information about current requests is considered available at the BS, together with the current queue sizes per frame.
- The frame control section has not been implemented.
- The three data burst profiles for the uplink time slots have not been implemented.
- Well behaved traffic sources have been assumed (with traffic policing if necessary)
- UGS data sources have not been considered for the simulation.

The following parameter values have been considered:

- The duration of every uplink frame = 1 ms. Each frame consists of s=20 time slots with 1 kb of data per slot. This value was used as the traffic unit. Per slot data rate is therefore 1 Mbps and the uplink data rate capacity is 20 Mbps.
- The simulated average uplink load used in the simulation has four possible values: 10%, 60%, 70% and 95% of the uplink capacity.

Simulated traffic sources belong to three service classes:

1. **rtPS** (real-time polling service):

- Seven MPEG2 sources with the maximum value of average BW of 1 Mbps per source
- Six MPEG2 sources with the maximum value of average BW of 0.5 Mbps per source
- These thirteen sources generate traffic that approximates MPEG2 video streams with 20 video frames per second, and 12*50 = 600 ms group of pictures (GOP) patterns, resulting after video compression. A typical GOP pattern consists of twelve bursts that occur once every 50 ms: I,B,B,P,B,B, P,B,B,P,B,B, where size(I) = 5*size(P) = 15*size(B).

2. **nrtPS** (non-real-time polling service):

- Ten FTP sources with the maximum value of average BW of 0.5 Mbps per source
- These sources generate traffic that approximates FTP streams with pseudo-random uniform distribution within given limits of both burst size and inter-burst duration.

3. **BE** (best effort service):

- Four HTTP sources with the maximum value of average BW of 1 Mbps per source

Maximum input load = 19 Mbps = 95% of link BW

Four algorithms have been simulated for performance comparison:

- Strict priority-based scheduler;
- Round robin scheduler.
- GA-based scheduling, using fitness function (1);
- Adaptive scheduling that emulates the behavior of the GA-based scheduler by using the approximation (5) for control function z, with c=0.5;

Simulation of the intra-class scheduling for the adaptive algorithm, at levels 4 and 5 of the hierarchy in Fig.3, are based on the very simple round-robin discipline.

The reasons for comparing the performance of the first two algorithms with the last two are:

- Strict priority-based scheduling provides the best guarantees for high priority classes, but it is the most unfair to the low priority ones.
- Round robin scheduling provides no service guarantees, but it is completely fair to all service classes.

These two algorithms are therefore the two extremes between which the performance of an optimized scheduler should be situated in order to accomplish the trade-off between providing service guarantees and ensuring some fairness to all service classes.

Characteristics of the simulated GA are as follows:

- Population size per generation = 50.
- Every individual represents a possible allocation of the available 20 time slots to up to 20 of the 27 traffic sources.
- The individual “genome” is a “chromosome” with 20 “genes”, one per time slot, where every gene specifies the identity of the traffic source connected to the respective time slot. Each gene can therefore take values between 1 and 27, and the size of the search space used for optimal resource allocation is 27^{20}.
- Each individual is characterized by a chromosome fitness which is calculated with formula (1). This fitness definition allows the QoS (Quality of Service) requirements to be met, and also provides some fairness in handling the link access requests.

- GA is of the elitist type. This means that population in every generation is sorted into three classes, according to the decreasing values of individual fitness. Class A contains the highest fitness individuals. They are transferred into the next generation with no change. Class A also provides the first parent for genetic cross-over, with the second parent selected from Class B or Class C. The two “children” resulting after cross-over are transferred into the next generation until a new Class B is created. Class C is replaced by randomly generated individuals that are transferred into the next generation. The cross-over probability for a gene is 0.7 [10].
- Any gene can also be mutated with a low probability (0.01) [10] by using the operation: ((1+j+r) mod 27), where j = gene’s value, and r = a pseudo-random number between 1 and 27.
- GA is terminated after an empirically determined number of generations are produced and analyzed.
- During every frame, the GA is run to determine the optimal allocation of the time slots in the next frame.

The simulation has been performed for 6000 frames (6 seconds).

Figs. 5, 6 and 7 illustrate the simulation results for the 1 Mbps rtPS, nrtPS and BE sources, respectively. Performance level can be estimated by comparing the average queue length for a given source class as a function of the scheduling algorithm, for link loads above 70%.

It is obvious that the shorter the queue length, the shorter the packet delay and the smaller the packet loss probability. A more detailed performance evaluation is, however, desirable to allow the direct measurement of these scheduler characteristics.

As expected, Fig.5 proves that the best performance for the 1 Mbps rtPS sources is obtained with priority-based scheduling and the worst - with round robin scheduling.

The adaptive method yields better results than the GA-based scheduling, and only slightly worse than the priority-based.

For the nrtPS sources in Fig.6, the GA-based method has the worst results, and the adaptive method – the best.

The priority and Round robin-based methods behave almost identically.

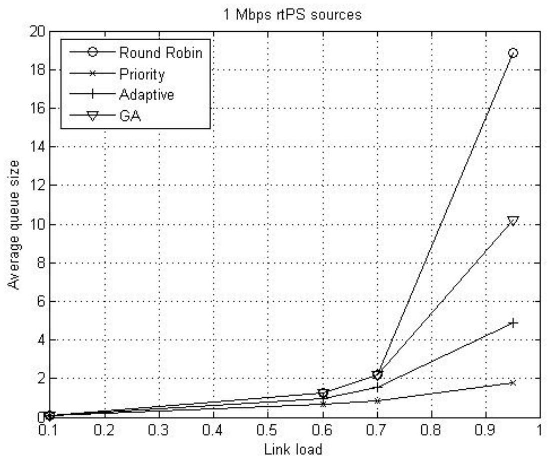

Fig.5. Performance level for the 1 Mbps rtPS sources

Again, as expected, Fig.7 illustrates that the best performance for the BE sources is provided by the Round robin method and the worst – by the priority-based. The adaptive and GA-based methods perform almost identically, and 50% better than the priority-based method.

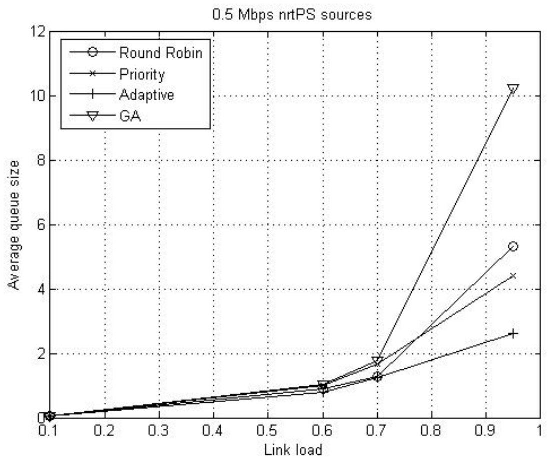

Fig.6. Performance level for the 0.5 Mbps nrtPS sources

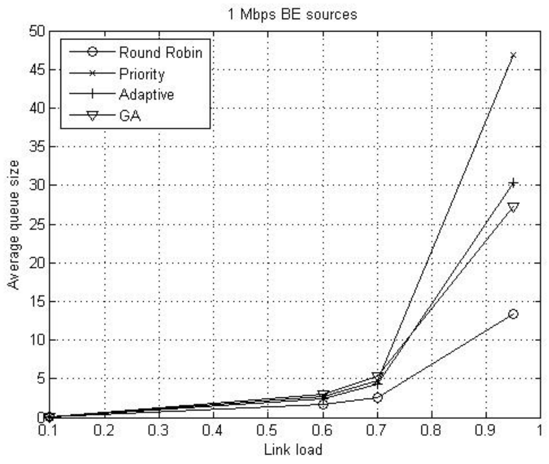

Fig.7. Performance level for the 1 Mbps BE sources

5. Conclusions

The adaptive algorithm proposed in this paper for WiMax uplink time slot allocation could be a useful addition to the existing methods for solving this problem. It combines the advantages of the GA-based approach which dynamically optimizes the slot allocation, once per frame, according to the current state of the system, with a remarkably simple processing that allows an inexpensive real-time implementation.

The basic characteristics of this algorithm are:

- Algorithm behavior is similar to the one manifested by a GA-based scheduler, obtained with an appropriately defined fitness function that allows the optimization of the trade-off between fairness and priority enforced service.
- Time slots are allocated using a multi-level hierarchical procedure with two-block partitionings being performed per level using control functions like (5).

By using this control functions, a very simple real-time implementation is possible.

A drawback of this approach, in comparison with all non-adaptive scheduling algorithms, is the fact that some of the uplink bandwidth needs to be reserved for the queue size transmission from SSs to the BS, once per frame. However, if only partitions 3 and 4’ in Fig.3 are obtained adaptively, then only three numerical values (total queue size for the rtPS, nrtPS and BE classes) per SS have to be sent to the BS at the beginning of every frame, which may be acceptable.

Simulation results described in Section 4 are encouraging and they suggest a number of interesting topics that should be considered for further research and simulation work:

- Using more realistic traffic loads with more source types and better performance evaluation. The very simple traffic source models considered in this paper allowed us to perform just a preliminary evaluation and validation of the proposed concepts.
- Investigating of other, possibly better, types of fitness functions for the GA-based model.
- Finding possibly better control functions either empirically or by (symbolic) regression.
- Using a similar adaptive algorithm for the control of WiMax uplink/downlink sub-frame boundary.
- Modeling the polling-request-grant mechanisms.

R E F E R E N C E S

[1] *Carl Eklund, Roger B. Marks, Kenneth L. Stanwood and Stanley Wang*, IEEE Standard 802.16: A Technical Overview of the WirelessMAN™ Air Interface for Broadband Wireless Access, IEEE Communications Magazine, vol.**40**, Jun 2002, pp. 98-107.

[2] IEEE Std 802.16-2001 Standard, The Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, New York, NY 10016-5997, USA, 8 April 2002.

[3] *Kitty Wongthavarawat, and Aura Ganz*, Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems, Int.J.Commun.Syst., vol.**16**, issue 1, Feb. 2003, pp.81-96.

[4] *Alexander Sayenko, Olli Alanen, Juha Karhula, and Timo Hämäläinen*, Ensuring the QoS requirements in 802.16 scheduling, Proc. 9th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Torremolinos, Spain, Oct. 2006, pp. 108-117.

[5] *P. Xu, M. Devetsikiotis and G. Michailidis*, Adaptive Scheduling using Online Measurements for Efficient Delivery of Quality of Service. Technical Report TR2004-12, March 2004, SAMSI, RTP, NC.

[6] *P. Xu, G. Michailidis and M. Devetsikiotis*, Profit-oriented resource allocation using online scheduling in flexible heterogeneous networks, Telecommun Syst., vol.**31**, 2006, pp. 289–303.

[7] *H.Wang, C. Shen and K.G. Shin*, Adaptive-Weighted Packet Scheduling for Premium Service, Proceedings of the IEEE ICC, 2001.

[8] *S.Thilakawardana, and R.Tafazolli*, Use of genetic algorithms in efficient scheduling for multi-service classes, Proc. 5th European Wireless Conference: Mobile and Wireless Systems beyond 3G (EW2004), Barcelona, Spain, Feb. 2004.

[9] *J.R.Koza, F.H.Bennet III, D.Andre, M.A.Keane*, Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufmann Publishers, 1999.

[10] *M.Ericsson, M.G.C.Resende and P.M.Pardalos*, A genetic algorithm for the weight setting problem in OSPF routing, J.Combinatorial Optimization, vol.**6**, no.3, 2002, pp.299-333.