U.P.B. Sci. Bull., Series C, Vol. 71, Iss. 1, 2009 ISSN 1454-234x

PERFORMANCE ANALYSIS OF AN ADAPTIVE SCHEDULER FOR IEEE 802.16 WIRELESS ACCESS SYSTEMS

Nicolae NECULA[1], Eugen BORCOCI[2]

Articolul prezintă analiza detaliată a performanţelor unei metode adaptive pentru optimizarea alocării canalelor temporale în tehnologia de acces radio de bandă largă IEEE 802.16. Această metodă emulează în timp real comportarea unui planificator off-line bazat pe un algoritm genetic care este proiectat astfel încât să coreleze în mod optim cerinţele contradictorii de servire prioritară şi servire echitabilă, necesare pentru obţinerea calităţii de servire (QoS), definite în standardul IEEE 802.16, pentru toate clasele de servicii.

This paper describes the detailed performance analysis of an adaptive method for optimal timeslot allocation in IEEE 802.16 wideband radio access technology. 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.

Key words: adaptive packet scheduling, genetic algorithms, Matlab simulation, QoS, IEEE 802.16.1

1. Introduction

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 Multiple Access (TDMA) with frames of fixed duration, consisting of a predefined number of time slots. In the Time Division Duplex Mode (TDD), the frames are dynamically subdivided by the 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 bandwidth and QoS requirements.

[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

While extensive bandwidth allocation and QoS mechanisms are provided, the details of scheduling and resource 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.

Optimization of the tradeoff between priority-based service and service fairness during periods of high load with large bursts of data is a challenging problem that is worth investigating and solving.

Conventional solutions allocate statically a fixed number of timeslots per frame to the different QoS (quality of service) classes, and use the spare timeslots for strict priority-based dynamic allocation, when needed [2,3].

The main drawback of this type of solutions is the poor fairness obtained when the sources in the higher priority classes generate large bursts of data at the same time.

A different approach has been described in [1], that accepts a slight degradation of the quality of service for higher priority classes as the price paid for a better fairness offered to the lower priority classes during the congested periods.

A simple real-time adaptive scheduler has been developed that mimics the behavior of a GA – based scheduler (GA=Genetic Algorithm), designed specifically for optimal balancing of the opposite requirements related to both priority and fairness.

This paper describes some recently obtained simulation results regarding the two tasks that have been mentioned in [1] as objectives for further work:

- Multi-level implementation of the adaptive algorithm, where different values of the algorithm parameter need to be determined, one per adaptive level.
- More accurate determination of these parameter values as a function of the input data parameters.

Section 1 describes the adaptive timeslot allocation algorithm. This is a real-time 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, has been exploited in a process of identification and modeling of the GA-based algorithm behavior.

Two WiMax uplink simulation scenarios are described in Section 2: the two-class adaptive timeslot allocation and the three-class hierarchical adaptive allocation.

Simulation results obtained for the Matlab models are then illustrated in Section 3 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.

After a brief section with conclusions, Appendix A presents the GA-based algorithm, and Appendix B describes the models of the VoIP, MPEG2 and HTTP data sources.

2. The adaptive timeslot allocation algorithm

The basic idea of the adaptive scheduling algorithm is to allocate dynamically, once per frame, the available resources to just two QoS classes or groups of classes (e.g.: real-time classes vs. all other classes), according to the following formulas that mimic the behavior of the GA-based scheduler:

```
IF $q_1+q_2>s$ THEN
        $r = q_1/(q_1 + c*q_2)$                (1)
        $n_1=\min(\mathrm{round}(r*s),q_1)$    (2)
        $n_2=\min(s-n_1,q_2)$                  (3)
ELSE $n_1=q_1$, $n_2=q_2$; ENDIF
    $n_3=\max(1,s-(n_1+n_2))$                  (4)
```

where:

- s = the number of un-reserved uplink timeslots,
- q_1 = total current length of all real-time queue(s),
- q_2 = total current length of all other queues,
- q_1 and q_2 are conventionally measured in traffic units (tu's), where 1 tu = the timeslot size in bits,
- r = % of frame timeslots allocated to real-time sources,
- c = an experimentally determined parameter,
- n_1, n_2 = the number of timeslots allocated to the two QoS classes,
- n_3 = the number of timeslots allocated to the BE class.

If the total queue lengths q_1+q_2, corresponding to the number of packets waiting to be dispatched from the queues is less than s, then all sources obtain their required number of timeslots and no resource partitioning algorithm is needed.

Otherwise, formula (1) provides the desired tradeoff between priority-based service and fair service: when $q_1 >> c*q_2$, r is close to 1, and almost all timeslots are allocated to the first class, but when q_1 and $c*q_2$ are comparable, then the second class gets more timeslots allocated to it. Formulas (2) and (3) provide the work conserving characteristic of the algorithm by never allowing the number of timeslots allocated to any non-BE class to exceed the current length of the queue used by the respective class.

Thus, no unused timeslot is wasted, but it is eventually allocated by formula (4) to the "greedy" BE sources. In order to prevent total starvation of the BE sources during the congested periods, one timeslot is reseved to them.

The value of parameter c determines the timeslot allocation policy, as follows:

- c=0 ensures strict priority-based allocation
- As c increases in value between 0 and 1, the allocation fairness between the two QoS classes gets better, but the price paid is the increased degradation of the service offered to the first class.

The value of c can be determined by the approximate formula:

$$c = f*b_2*q_{1max}/(b_1*q_{2max}) \quad (5)$$

where:

- f = the desired "degree" of fairness to be achieved,
- b_1, b_2 = the mean bandwidth required by the two QoS classes,
- q_{1max} and q_{2max} = the maximum queue lengths for these classes. These values are used as approximate maximum burst sizes.

By assuming, for example, that f=1, $b_1=b_2$, and $q_{1max}/q_{2max} = 1/1000$, then, after substituting (5) in (1), when $q_1 = q_{1max}$ and $q_2 = q_{2max}$, the resulting values c=0.001 and r=0.5 provide complete fairness to the two QoS classes, i.e. round robin scheduling.

Partitioning of the available bandwidth (BW) into just two blocks is advantageous because it is easy to control and simple to implement. It also allowed an easy identification of the corresponding control function from the behavior of a GA-based algorithm, as described in Appendix A.

The main characteristics of the described scheduling algorithm are:

- It is adaptive, i.e. all the timeslots are allocated dynamically, as needed, for improving the fairness, while maintaining or slightly reducing the quality of higher priority service.
- It is very easy to implement in a real-time controller.
- A drawback is the overhead caused by the required transmission of q_1 and q_2 values from all SSs to BS once per frame.

Resource allocation to more than two QoS classes of interest requires obviously a repeated application of the above algorithm in a multi-level hierarchical manner.

The adaptive 3-class partitioning is therefore, as follows:

```
IF $q_1+q_2+q_3>s$ THEN
    $r_1 = q_1/(q_1 + c_1*(q_2+q_3))$        (6)
    $n_1=\min(\mathrm{round}(r_1*s),q_1)$        (7)
    $s_1=s-n_1$
    $r_2 = q_2/(q_2 + c_2*q_3)$        (8)
    $n_2=\min(\mathrm{round}(r_2*s_1),q_2)$        (9)
    $n_3= \min(s_1-n_2,q_3)$        (10)
ELSE $n_1=q_1$, $n_2=q_2$, $n_3=q_3$; ENDIF
$n_4=\max(1,s-(n_1+n_2+n_3))$        (11)
```

where:

- s = the number of un-reserved uplink timeslots,
- q_1,q_2,q_3 = total current lengths of all queues for the 3 QoS classes,
- r_1 = % of frame timeslots allocated to the top QoS class,
- r_2 = % of remaining frame timeslots allocated to next QoS class,
- c_1,c_2 = experimentally determined parameters,
- n_1,n_2,n_3 = the number of timeslots allocated to the 3 QoS classes,
- n_4 = the number of timeslots allocated to the BE class.

In the first step, formulas (6) and (7) partition the set of s timeslots into two subsets: n_1 are allocated to the top class, and $s_1=s-n_1$ - to the other classes.

In the second step, formulas (8) and (9) partition the remaining s_1 timeslots into two new subsets: n_2 are allocated to the second class, and n_3 - to the third.

Per frame multi-level hierarchical bandwidth partitioning (fixed or adaptive) 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 WiMax link sharing, as illustrated in Fig.1.

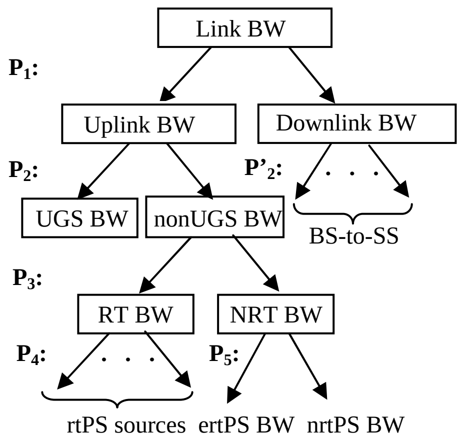

Fig.1. A possible multi-level hierarchical partitioning of the BS-SSs link bandwidth (BW)

Partition P_1 is dynamic, as specified in the 802.16 standard, P_2 is generally fixed, and P_4 is usually EDF (earliest deadline first)-based.

Partitions P_3 and P_5 are the two possible candidates for the multi-level adaptive implementation described above.

2. Evaluation scenarios

2.1. Simulation environment and its characteristics

- Physical layer:
  - 1000 frames per second
  - 16 time slots (TSs) per frame
  - 256 bits per TS = conventional packet size
  - 100% uplink occupancy because of the BE sources
  - One timeslot is reserved for the BE sources
- Performance measurements:
  - Probability of packet loss for the rtPS flows, generated by VoIP sources. VoIP packets are lost when they are not able to meet the chosen 20 ms deadline.
  - Mean number of timeslots allocated to all non-BE classes.
  - Mean service delay for all non-BE classes.
  - Packet delay distribution for rtPS and nrtPS flows. It is assumed that nrtPS flows use large buffers and therefore no packet loss occur. The packet delay is measured in frames: if a packet is received in frame m and it is transmitted in frame $m+k$, then its delay is k frames.

  - All these measurements are obtained for heavy values of the input load: 0.8, 1 and 1.2 of the 15 timeslot capacity (referred to as the relative input load).
- Simulation characteristics:
  - Simulation duration = 100 seconds.
  - In order to smooth statistical fluctuations, all results are averaged over 10 successive runs, without re-initialization of the Matlab random number generator.

2.2. Two-class scenario

Specific simulation environment and characteristics:

- 2+1 service classes:
  - Highest priority (rtPS): VoIP sources
  - Next priority (nrtPS): HTTP sources
  - Lowest priority: BE “greedy” sources
- Input traffic characteristics:
  - Equal BW demand (50% each) from the top two classes for all considered input loads.
  - VoIP and HTTP source models are as described in Appendix B.
- Parameter c values are chosen as follows:
  - c=0 (strict priority for the two top classes)
  - c=0.001 (some fairness for HTTP, relative to VoIP)
  - c=0.002 (more fairness for HTTP, relative to VoIP).

2.3. Three-class scenario

Specific simulation environment and characteristics:

- 3+1 service classes:
  - Highest priority (rtPS): VoIP sources
  - Next priority (ertPS): MPEG2 sources
  - Next priority (nrtPS): HTTP sources
  - Lowest priority: BE “greedy” sources
- Input traffic characteristics:
  - Equal BW demand (33% each) from all top three classes for all considered link loads.
  - VoIP, MPEG2 and HTTP source models are as described in Appendix B.
- Parameter c_1 and c_2 values are chosen, according to the traffic characteristiscs of the three classes, as follows:
  - c_1=0, c_2=0 (strict priority for the three top classes)

- c_1=0.001, c_2=0.5 (some fairness for MPEG2 and HTTP, relative to VoIP, good fairness for HTTP relative to MPEG2)
- c_1=0.002, c_2=0.5 (some more fairness for MPEG2 and HTTP, relative to VoIP, good fairness for HTTP relative to MPEG2)
- c_1=0.004, c_2=0.5 (more fairness for MPEG2 and HTTP, relative to VoIP, good fairness for HTTP relative to MPEG2)
- c_1=0.004, c_2=0.9 (same more fairness for MPEG2 and HTTP, relative to VoIP, very good fairness for HTTP relative to MPEG2)

Since VoIP bursts are much smaller than MPEG2 ones, whereas MPEG2 and HTTP bursts are both very large, the values used for c_1 are, according to formula (5), much smaller than those used for c_2.

3. Simulation results

The graphs included in this section illustrate the scheduler performance for the two scenarios, regarding:

- VoIP packet loss probability (Figs.2 and 9)
- Mean number of VoIP, MPEG2 and HTTP timeslots (TSs) per frame (Figs.3,4 and 10-12)
- Mean service delay (Msd) for the 3 classes of packets (Figs.5,6 and 13-15)
- VoIP packet delay distribution for the 2-class scenario (Figs.7, 8).

Figs. 2, 3 and 4 illustrate the way the adaptive algorithm handles the VoIP and HTTP sources, in the first simulation scenario, for different values of parameter c.

When c=0, the VoIP packets are serviced with highest priority, and therefore no packet loss occurs even for 20% input overload. In this case, the number of VoIP TSs grows linearly with the input load. For c>0, the 20% input overload leads to less than 8% VoIP packet loss, when c=0.001, which is acceptable under such high link congestion. However, when c=0.002, the 18% loss is excessive.

Fig.4 illustrates the better fairness obtained, for higher values of c, in servicing the HTTP flows during periods of high congestion (20% overload).

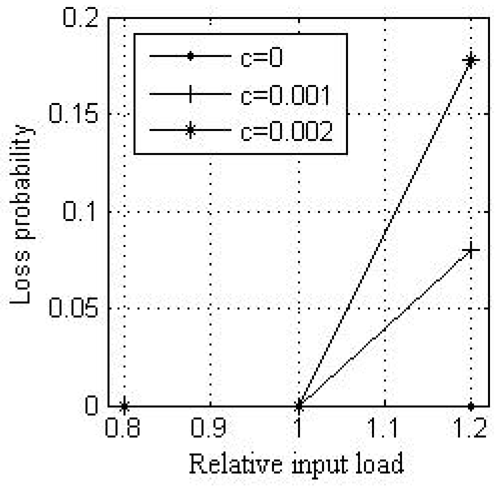

Fig.2. VoIP packet loss probability

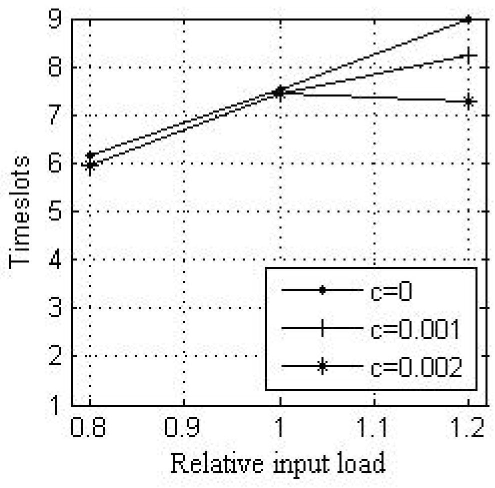

Fig.3. Mean number of VoIP TSs

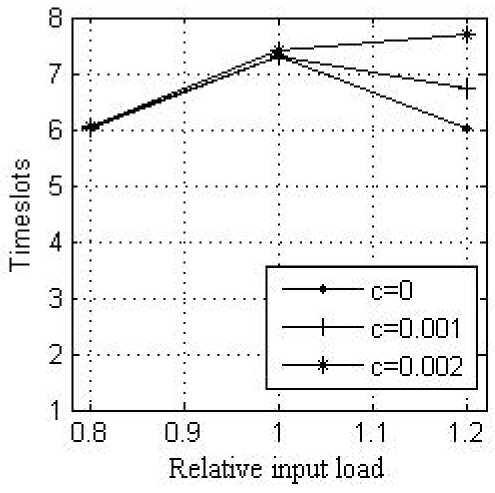

Fig.4. Mean number of HTTP TSs

The mean service delay (Msd) is less than 20 ms for VoIP flows, as illustrated in Fig.5. For HTTP flows, it decreases by approximately 25%, when relative input load =1.2 and c=0.001 (Fig.6).

Packet delay distribution (pdd)is illustrated for VoIP sources only in Figs. 7 and 8, for c=0 and c=0.002.

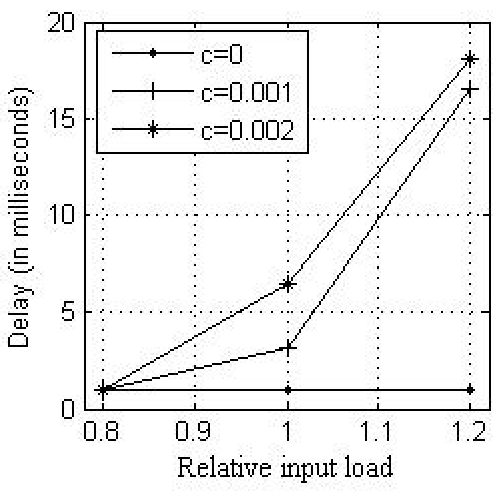

Fig.5. Msd for VoIP packets

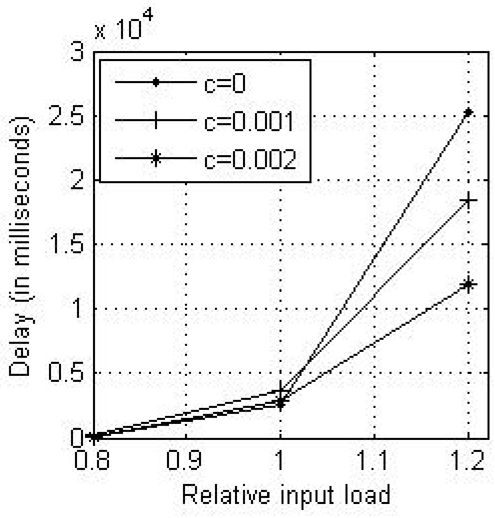

Fig.6. Msd for HTTP packets

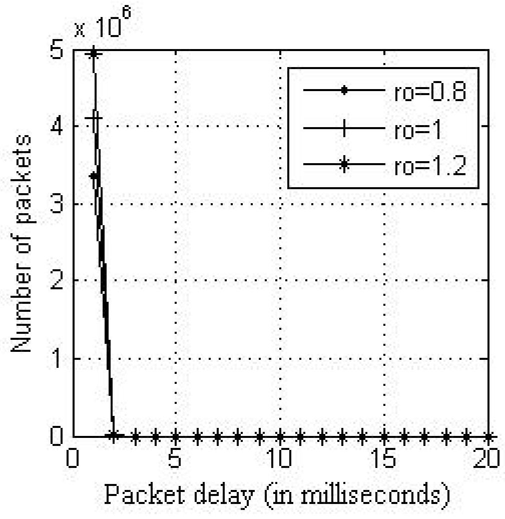

Fig.7. VoIP pdd for c=0

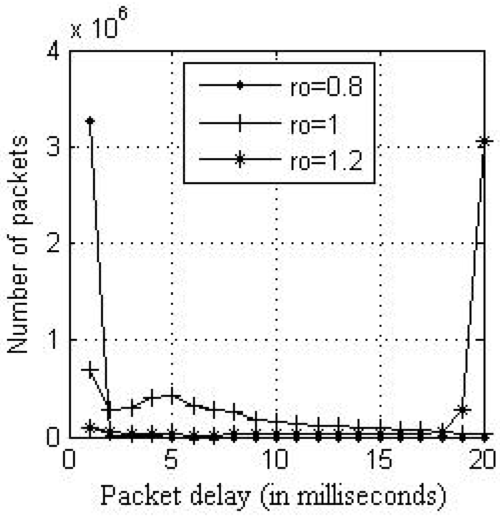

Fig.8. VoIP pdd for c=0.002

The expected behavior of the adaptive scheduler is conclusively illustrated by the above figures.

For c=0, the mean number of VoIP timeslots grows almost linearly with the VoIP input load, reaching a value of 9 (i.e. more than half of the total number of TSs), because c=0 enforces strict priority-based service, so that the VoIP class receives everything it needs. For c=0.001 and greater, the number of TSs saturates in the overload conditions, allowing less channels to be allocated to the

VoIP sources, in order to improve the fairness in servicing the nrtPS class. The price paid for this limitation is the occurrence of lost VoIP packets.

VoIP mean service delay (Msd) is minimal for c=0, as expected, and it increases monotonically towards the deadline value when the input load increases.

For c=0, the mean number of HTTP TSs is severely limited as the input load grows into the overload conditions. However, for c=0.002, a value that enforces more fairness for the two classes, the mean number of HTTP timeslots grows continuously with the input load. By choosing c=0.001, a reasonable fairness for HTTP is obtained in the overload conditions.

Mean service delay of HTTP packets, which is not very relevant for evaluating the scheduler performance, decreses when c increases, during the 20% link overload condition.

Simulation results for the 3-class scenario provide further insights into the adaptive algorithm operation. Figs.9-15 illustrate some of these results, with the legend of Fig.9 being used in Figs.10-15, as well.

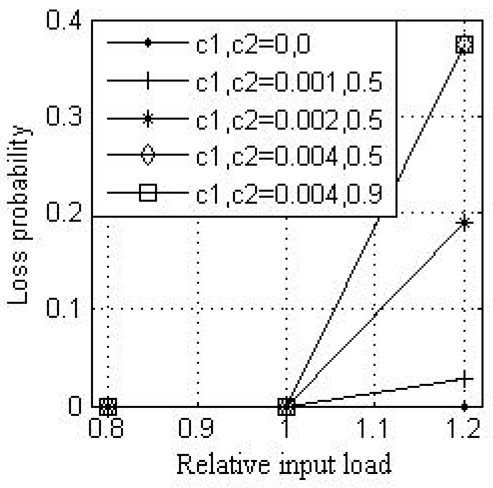

Fig. 9. VoIP packet loss probability

In Fig.9, when $c_1=c_2=0$, the VoIP packets are serviced with highest priority, and therefore no packet loss occurs even for 20% input overload. As the value of c_1 is increased from 0 to 0.001, 0.002 and 0.004, the loss probability, for 20% input overload, grows from 0 to 3%, 19% and 38%, respectively. The c_2 value has no impact on the loss probability, because it does not affect the resource allocation to the VoIP class. Higher enforced fairness leads obviously to unacceptable packet losses.

The next 6 figures can now be used to analyze how the scheduler performance is affected for all the top 3 classes, when choosing a fixed acceptable value for $c_1=0.001$ and different values for c_2.

In Fig.10 it is obvious that the mean number of TSs allocated to VoIP sources for 20% overload is almost the same as in the strict priority case.

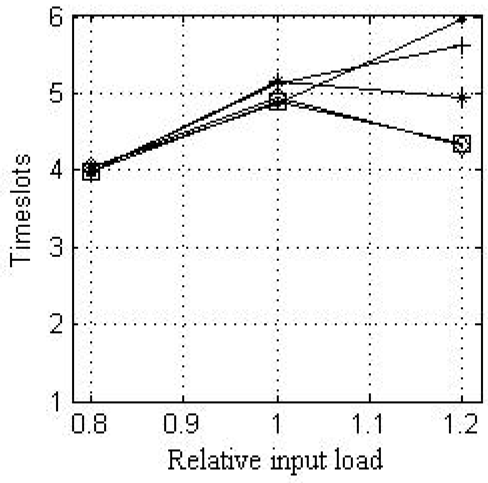

Fig.10. Mean number of VoIP TSs

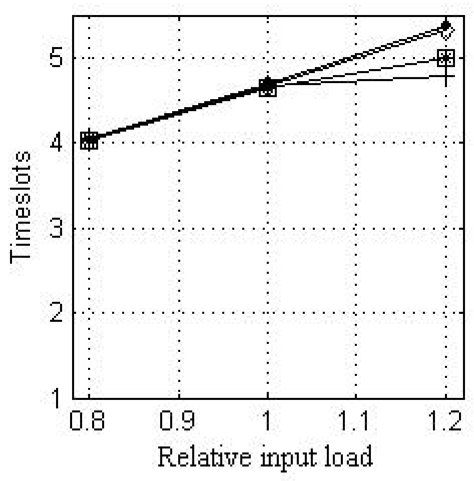

Fig.11. Mean number of MPEG2 TSs

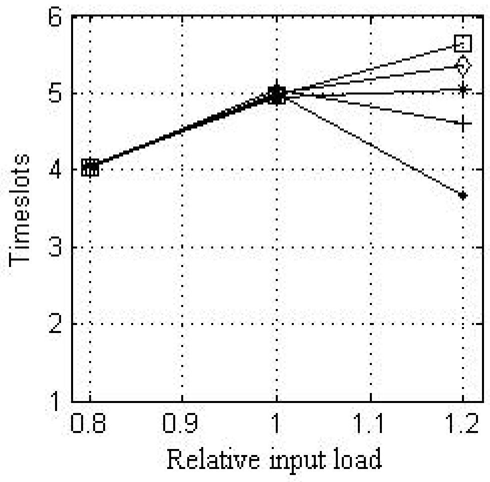

Fig.12. Mean number of HTTP TSs

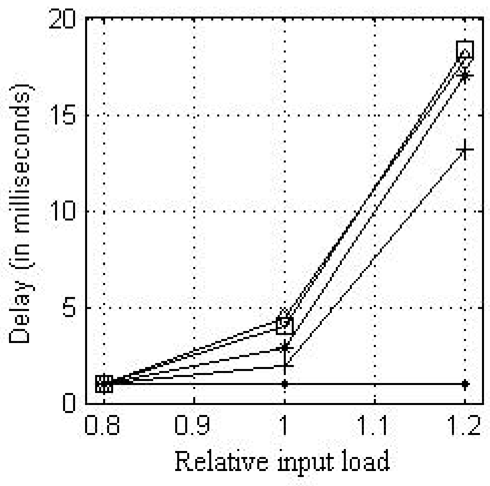

Fig.13. Msd for VoIP packets

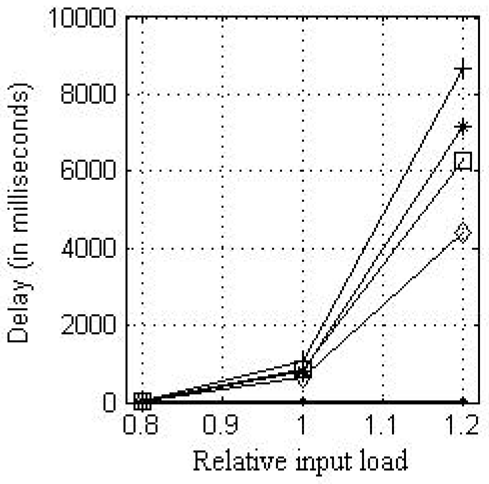

Fig.14. Msd for MPEG2 packets

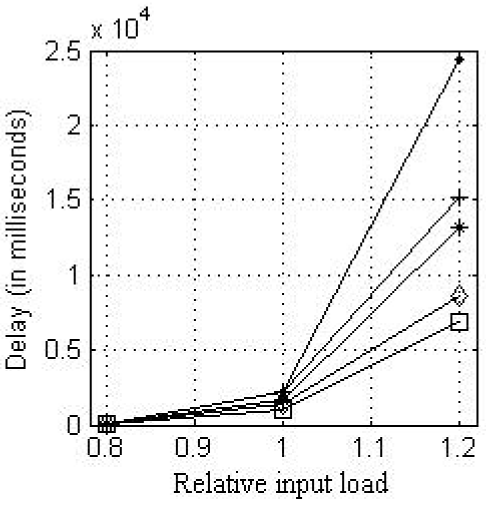

Fig.15. Msd for HTTP packets

Fig.11 shows some saturation for the mean number of TSs allocated to MPEG2 sources for 20% overload, while Fig.12 illustrates that one extra TS is allocated to HTTP sources for this overload, when the strict priority scheduling is replaced by the slight fairness enforcing, due to c_1=0.001.

For the same 20% overload, the above service policy replacement causes the mean packet service delay to increase for both the VoIP and MPEG2 packets (Figs.13 and 14), whereas the HTTP packets experience a decrease by 10 seconds of this mean service delay (Fig.15).

This illustrates that, for the price of small acceptable degradations of the performance for the highest priority class, the quality of service for the other classes can be improved, particularly during heavy overload conditions.

3. Conclusions

The adaptive scheduling algorithm described in this report is a useful addition to the numerous existing scheduling algorithms.

It provides a better fairness in servicing the lower priority classes during (severe) link overload conditions, by enforcing just minor (acceptable) quality degradation in servicing the higher priority classes.

An adaptive packet scheduler, like the one described in this paper, seems to be capable of providing not only the guaranteed service that has been required and paid for accordingly, to as many subscribers as possible, but also a better service fairness to all subscribers.

While the results obtained in this paper are mostly “as expected”, the practical value of the presented approach and study consists in offering both a simple real-time adaptive scheduling algorithm, and quantitative values for delays and loss in high load conditions, under realistic operating scenarios. Such values could be used, together with selection of parameters c_1 and c_2, in a policy driven management context where the operator can define different degrees of fairness and acceptable overload for his/her network, the target being a controllable QoS for real time flows (like VoIP, MPEG video) combined with a good link utilization of WiMAX resources.

R E F E R E N C E S

[1] *N.Necula, E.Borcoci*, Adaptive packet scheduling for QoS support in IEEE 802.16 wireless access systems, UPB Scientific Bulletin, Series C, Vol.70, No.1, 2008, pp.3-16.

[2] *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, pp.81-96, Feb. 2003.

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

[4] *S.Thilakawardana and R.Tafazolli*, Use of genetic algorithms in efficient scheduling for multi-service classes, European Wireless Conference 2004, Barcelona, Spain, 24-27 Feb. 2004.

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

[6] *Jose Manuel Gimenez, Maria Jose Domenech, Jorge Martinez and David Garcia*, Source characterization, in: *N.Necula* (Editor), Resource allocation algorithms for wireless MANs, STEP.D3 Deliverable for IST Project “Design and Engineering of the Next Generation Internet”, Network of Excellence, Nov.2006, www.eurongi.org

[7] *Fred Halsal*, Multimedia Communications, Sub-chapter 4.3: Video compression, Addison-Wesley, 2001.

APPENDIX A: A GA-BASED SCHEDULER

Each chromosome represents a possible per frame timeslot allocation, (UL-MAP). The fitness assigned to a chromosome is calculated, according to [4], by:

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

where:

s = chromosome length = the number of timeslots per frame,

F_i = fitness assigned to the i*th* time slot , with regard to the traffic source serviced by this time slot:

$$F_i = \frac{P_i * Q_i}{\sqrt{f_i}}$$

where:

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

Q_i = dynamic queue length of this traffic source

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

By maximizing the above fitness, the GA attempts to find, once per super-frame, an optimized chromosome capable to assign all the available timeslots in such a manner that achieves a balance between the opposite requirements regarding both priority-based service and fairness [4].

The nearly optimal timeslot allocations, thus obtained, cannot be used in a real-time controller because of the huge amount of required processing power. However, the behavior of such an offline GA-based scheduler is still worth investigating and possibly emulating in real-time.

The following steps have been used in [1] to identify a function that describes how the GA-based scheduler partitions adaptively and (almost) optimally the set of available timeslots into two subsets, corresponding to two QoS classes of data sources:

1. Function definition: $r = F(q_1,q_2)$ partitions dynamically the set of available timeslots per frame into two subsets, e.g. *rt* and *nrt* traffic sources, as a function of the current length of the queues assigned to these subsets. Queue length is measured in conventional traffic units. In our simulations, one traffic unit equals the timeslot bandwidth.

2. Function identification: By using the fitness function defined in Section 1, together with a mix of *rt, nrt and BE* traffic sources, the GA-based scheduler is run once per super-frame, and triplet $\{q_{1i},q_{2i},r_i\}$ found in the i-th super-frame is stored in an array, for the duration of the simulation. The resulting three-dimensional array describes the dynamic behavior of the GA when handling *rt* vs. *all other* packets. Then, a surface r = F(q_1,q_2) that fits best the “cloud” of stored triplets is determined either empirically or by using the symbolic regression method of Genetic Programming [5].

APPENDIX B: TRAFFIC DATA MODELS

B.1. VoIP (Voice over IP) [6]

Real time rtPS services support service flows with variable size data packets, e.g voice over IP **(**VoIP**)** or MPEG video.

A VoIP conversation, like traditional telephony, can be considered an ON-OFF process during which one of the parties is speaking, and the other is listening. The ON duration can be modeled by a hyperexponential distribution with two stages, while the OFF period duration can be represented by a hyperexponential distribution with three stages. The parameters of these distributions are shown in Table B.1.

Table B.1

Parameters of the hyperexponential distributions for VoIP model.

		Weight	**Rate**
ON	Stage 1	0.63	0.85
	Stage 2	0.37	0.40
OFF	Stage 1	0.50	1.83
	Stage 2	0.20	0.25
	Stage 3	0.30	19.68

B.2. MPEG2 [7]

In the simplified MPEG2 source model, a 0.6 second group of pictures (GOP) structure is used, consisting of 12 frames (bursts) of three types: I, P and B. Each frame occurs once every 50 ms, and the frame sequence is: I,B,B, P,B,B,P,B,B,P,B,B, with Mean_size(I)=5*Mean_size(P)=15*Mean_size(B). These three sizes are normally distributed and have a standard deviation is chosen arbitrarilly as approximately one quarter of the respective Mean_size.

B.3. HTTP [6]

The nrtPS is designed to support delay-tolerant data streams consisting of variable-sized data packets for which a minimum data rate is required. Since the traffic from the client to the server is negligible, only the traffic from the server to the client is taken into account. This traffic is modeled as shown in Fig.B1.

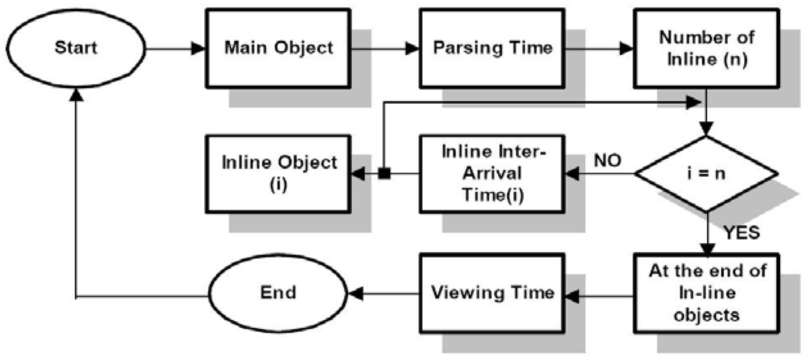

Fig. B.1: State transition diagram for HTTP traffic generation.

At the beginning, the traffic corresponding to the main object is generated, with its size following a lognormal distribution. Just after the reception of the main object, there is a *parsing time* period that comprises the time in which the web browser analyses the main object looking for the number of inline objects. That period of time and the number of inline objects follow a gamma distribution. The first inline object starts after the expiration of the *parsing time*, and the second one starts one *inline interarrival time* after the start of the first one. Frequently, *inline interarrival time* is less than the duration of the connection, and hence the model includes the effect of parallel downloading of inline objects. After all the objects are transmitted, the model remains silent during the *viewing time*, starting again, after that period.

The probability distributions and their respective parameters are summarized in Table B.2.

Table B.2

Distributions and parameters for HTTP sources.

Parameter	**Distribution**	**Mean**	**Standard Deviation**
Main object size (bytes)	Lognormal	10710	25032
In-line object size (bytes)	Lognormal	7758	126168
Parsing time (seconds)	Gamma	0.13	0.187
Number of in-line objects	Gamma	5.55	11.4
In-line interarrival time (seconds)	Gamma	0.86	2.15
Viewing (OFF) time (seconds)	Gamma	39.5	92.6

B.3. Best Effort (BE)

Best Effort services support data streams with no quality of service requirements. This type of service class is useful, for example, for background applications where delay has no impact. The implementation of this type of service is done using an infinite file transfer, which will use all the remaining bandwidth, i.e., the free time slots that are not used by the rest of the services. This elastic traffic source will adapt its rate to the conditions of the link. These sources are “greedy”, trying to occupy all the resources available at any given time.