SWIM: A Scheduler for Unsolicited Grant Service (UGS) in IEEE 802.16e Mobile WiMAX Networks*

Chakchai So-In, Raj Jain, and Abdel-Karim Al Tamimi

Department of Computer Science & Engineering, Washington University in St. Louis
One Brookings Drive, Box 1045, St. Louis, Missouri 63130 USA
{cs5, jain, aa7}@cse.wustl.edu

Abstract. Most of the IEEE 802.16e Mobile WiMAX scheduling proposals for real-time traffic using Unsolicited Grant Service (UGS) focus on the throughput and guaranteed latency. The delay jitter and the effect of burst overhead have not yet been investigated. This paper introduces a new technique called **Sw**apping M**i**n-**M**ax (SWIM) for UGS scheduling that not only meets the delay constraint with optimal throughput, but also minimizes the delay jitter and burst overhead.

Keywords: Scheduling, Resource Allocation, Mobile WiMAX, IEEE 802.16e, Unsolicited Grant Service, UGS, QoS, Delay Jitter

1 Introduction

One of the key features of the IEEE 802.16e Mobile WiMAX is its strong quality of service (QoS). The IEEE 802.16e Mobile WiMAX provides the multiple QoS classes for voice, video, and data applications [1]. To meet QoS requirements especially for voice and video transmissions with delay and delay jitter constraints, the key issue is how to allocate resources among contending users. That is why there are many papers on designing resource allocation algorithms for the IEEE 802.16e Mobile WiMAX [2].

The IEEE 802.16e Mobile WiMAX offers five classes of service: Unsolicited Grant Service (UGS), extended real-time Polling Service (ertPS), real-time Polling Service (rtPS), non-real-time Polling Service (nrtPS), and Best Effort (BE). UGS is designed for Constant Bit Rate (CBR) traffic with strict throughput, delay, and delay jitter constraints. ertPS is a modification of UGS for voice with silence suppression. rtPS is designed for variable bit rate voice, video, and gaming applications that have delay constraints. nrtPS is for streaming video and data applications that need throughput guarantees but do not have delay constraints (the packets can be buffered).

* This work was sponsored in part by a grant from Application Working Group of WiMAX Forum. "WiMAX," "Mobile WiMAX," "Fixed WiMAX," "WiMAX Forum," "WiMAX Certified," "WiMAX Forum Certified," the WiMAX Forum logo and the WiMAX Forum Certified logo are trademarks of the WiMAX Forum.

BE is designed for data applications that do not need any throughput or delay guarantees.

These five service classes can be divided in two main categories: non-real-time and real-time. nrtPS and BE are in the first category and UGS, rtPS, and ertPS are in the second category. For the first category, common schemes can directly apply such as Weighted Fair Queue (WFQ) and a variation of Round Robin (RR) since there are no hard constraints on delay and delay jitter [2]. On the other hand, real-time services have strict constraints on these parameters. This makes scheduling difficult in trying to meet the delay constraint and tolerate the delay jitter with optimal throughput.

UGS is one of the real-time services. Basically, UGS traffic provides a fixed periodic bandwidth allocation. Once the connection is setup, there is no need to send any other requests. UGS is designed and used commonly for Constant Bit Rate (CBR) real-time traffic such as leased-line digital connections (T1/E1) and Voice over IP (VoIP). The main QoS parameters are maximum sustained rate, maximum latency, and tolerated jitter (the maximum delay variation).

As indicated earlier, previous papers on the IEEE 802.16e Mobile WiMAX scheduling have ignored the effect of burst overhead and often ignored the delay and delay jitter constraints [2 to 5]. In this paper, we propose an algorithm for UGS scheduling that includes these features. Although, the discussion in this paper is limited to UGS service class only, we plan to extend this algorithm for other real-time services and for a mixture of users from different service classes.

Scheduling Factors

The scheduler for UGS needs to be designed to meet the four main QoS criteria for the IEEE 802.16e Mobile WiMAX [1, 2]. First, to optimize system throughput, that is, the scheduler should use all available UGS slots if there is traffic.

Second, the scheduler should guarantee the delay constraints or maximum latency guarantees. In this paper, we also use the term “deadline” to mean delay constraint because the allocation is made within the deadline.

Third, the scheduler should minimize delay jitter. The definition of delay jitter is the variability in inter-packet times from one inter-packet interval to the next.

Finally, the scheduler should minimize number of bursts in order to reduce Media Access Control (MAC) and MAP overheads that reduce system throughput.

2 Related Work

There has been some research on delay jitter control for real-time communication in ATM and packet data networks. One way is to introduce a delay jitter regulator or rate regulator at each hop. The regulator delays a packet in order to keep constant delay jitter over the end-to-end path [6, 7, 8]. This method minimizes delay jitter increasing the mean delay and possibly reducing the throughput.

As shown in our extensive survey of the IEEE 802.16e Mobile WiMAX schedulers [2], channel-unaware IEEE 802.16e schedulers have applied two techniques for UGS traffic: Weighted Round Robin (WRR), equally spread the allocation over all Mobile

WiMAX frames (we call this averaging or AVG algorithm) and Earliest Deadline First (EDF) [3, 4, 5]. With the admission control, these techniques can achieve optimal throughput and meet deadlines; however, the delay jitter is not considered. This parameter is one of the required QoS parameters for UGS, that is, the tolerated jitter [1].

In addition, most papers have ignored burst overhead, which directly depends on how many bursts a Base Station (BS) allocates in a Mobile WiMAX frame [2]. Therefore, the delay jitter and the number of bursts are investigated in this paper. In this paper, we introduce a new algorithm, called SWIM (**Sw**apping M**i**n-**M**ax). This algorithm assures deadlines and delay jitter constraints, optimizes the throughput, and also minimizes the number of bursts. We show that along with zero delay jitter, a number of bursts with SWIM are comparable with those of EDF.

The paper is organized as follows: UGS allocation algorithm with assumptions of arrival traffic and parameters is described in Section 3. Then, the performance evaluation and examples are demonstrated in Section 4. Finally, the conclusions are discussed.

3 SWIM Algorithm

In this section, general assumptions are described first in subsection 3.1. Our algorithm can be used for both downlink allocation and uplink allocation. However, the problem is more difficult in the uplink since a Base Station (BS) has no information about the actual traffic at Mobile Stations (MSs), i.e., the arrival traffic process or queue length. The BS only knows about total demand and the period. Then, in subsection 3.2, SWIM algorithm is introduced. The SWIM algorithm basically can be divided into three basic steps that achieve optimal throughput while meeting the deadline, minimal delay jitter (in fact zero delay jitter), and minimal number of bursts.

3.1 Assumptions and Parameter Explanation

Basically for the IEEE 802.16e scheduler, all allocations are integer number of slots. In this paper, the definition of resource is fixed in terms of the number of slots per uplink (or downlink) subframe. This is denoted by the variable *#slots*. The number of bytes corresponding to a slot depends upon the modulation and coding which can vary among users. Without any loss of generality, we use a fixed number of bytes per slot and use bytes as the unit of resource allocation and demand.

For UGS traffic, at connection setup, MSs basically declare the total demand (denoted by *DataSize*) and a *period*. For example, *connection_1* asks 540 bytes every 3 frames. In other words, every 15 ms (WiMAX profiles specify a frame size of 5ms [9]). Due to the periodic nature of UGS traffic, the period is the same as deadline. We use these terms interchangeably; however, we show in Section 4.4 that if the deadline is less than the period, the throughput is not optimized.

MSs can dynamically join and leave the networks. For joining, in order for BS to

admit a connection, the BS needs to verify if there are enough resources. Also, the MS may request to change its service on the fly. Therefore, the scheduler needs to be aware of the quality assurance of all currently accepted connections.

In other words, there is an admission control mechanism. The BS can only admit a connection if and only if the sum of the total number of currently used resources per frame and the new demand divided by the deadline of the connection is less than the total available slots per frame for UGS.

The allocation algorithm is based on *DataSize* which is a MAC Service Data Unit (SDU) size for both deadline and delay jitter calculation. We do not explicitly consider any headers such as fragmentation and packing headers, MAC header, and ARQ retransmission overhead. However, the MS has to include these overheads in its demand at the connection setup time.

Finally, we assume that the MS has data available at the beginning of each period. In other words, the MS has enough buffer space at least for one period. This allows the BS to allocate the resources anytime within the deadline.

3.2 Algorithm Description

Our algorithm has two parts. First, an initialization procedure that starts with optimal throughput and delay. Second, a series of resource swapping steps that leads to optimization of all goals.

Given n users with i^{th} user demanding d_i over a period p_i, optimal throughput can be obtained by taking a Least Common Multiple (LCM) of periods p_i's and allocating resources over this cycle.

To achieve zero delay jitter, the algorithm initializes allocated resources (#slots or #bytes) for each connection by *DataSize/period*, i.e., d_i/p_i.

In each frame, the connection with the maximum resource allocation is called *max-res* connection and the one with the minimum allocation is called *min-res* connection.

To minimize the number of bursts, there is a swapping procedure between *max-res* and *min-res* connections that results in eliminating the *min-res* connection and thereby, reducing the number of connections served in that frame by one. In effect, this reduces the number of bursts in that frame by one. This will become more clear in Section 4, where we provide an example.

The swapping procedure is described as follows: first, the algorithm determines the *min-res* connection (say, ith connection) and the *max-res* connection (say, jth connection). The two connections swap their resources such that ith connection gives up its resources in the current frame while gaining an equal amount of resources in a future frame. Of course, the constraints are that jth connection still needs more resources in this frame and that ith connection's deadline will still be met.

The system manager can set a minimum burst size parameter, *MinBurstSize*. The swapping procedure ensures that each burst is at least this size. In our examples, we use a *MinBurstSize* of 1. However, the procedure can be easily applied for any other values of this parameter. The main effect of this parameter is that the connections whose deadline is in the current frame must have *MinBurstSize* allocation or more. If

their allocation is equal to *MinBurstSize*, they are excluded from swapping. Leaving *MinBurstSize* at a non-zero value ensures that all SDUs are delivered exactly at the deadline and the delay jitter is zero. Setting *MinBurstSize* to zero will result in a reduced number of bursts but non-zero delay jitter. The SWIM algorithm will then produce results similar to EDF.

The new *max-res* and *min-res* connections do the resource swapping. Note that the total allocated resources per frame do not change by this swapping procedure. Also, the total resources allocated to a connection over its period do not change.

```
Preallocation(flows)                                  //1st step
Sorted max to min = Sort (flows)
FOR each max_res in Sorted_max_to_min                 //2nd step
     Sorted_min_to_max = Sort (flows)
     FOR each min_res in Sorted_min_to_max            //3rd step
          Max_Min_Swapping (max_res, min_res);
     END FOR
END FOR
```

Fig. 1. Steps in SWIM Algorithm

There are a few special cases. First, a *max-res* connection cannot accept more resources than it needs and so the *min-res* connection may not get eliminated. In this case, the next *max-res* connection becomes the candidate for swapping for the remaining resources of the *min-res* connection.

Second, if there are more than one *max-res* connections (more than one connection with the same maximum resources allocated in the frame), we choose the connection whose resources are higher in the next frame.

Third, if there are more than one *max-res* connections with the same next frame resources, we select the connection whose deadline is longer. Of course, we exclude the connections whose deadline is in the current frame and which have allocation equal to *MinBurstSize*.

Fourth, if there are more than one *min-res* connections, we select the connection that has earlier deadline. Also, if there are more than one *min-res* connections with the same deadline, we choose the connection with lower resources in the next frame.

4 Performance evaluation and examples

In this section, we evaluate the performance of the proposed algorithm with other two commonly used algorithms: EDF and AVG (allocating *DataSize/period*, d_i/p_i in each frame to the ith user). For all three algorithms, the process is cyclic that repeats after LCM period. We show just one such cycle.

First, we evaluate the performance in terms of throughput, mean delay, mean delay jitter, and number of bursts for each algorithm. Then, the concept of flow admission is discussed. Finally, we show an alternative scenario in which the deadline is less than period. In that case, all resources cannot be allocated to UGS connections optimally.

Some resources are left over and can be used by other service classes.

4.1 Throughput, Mean Delay, Mean Delay Jitter, and Number of Bursts

The throughput, mean delay, mean delay jitter, and number of bursts are investigated in this section. We start with a simple example (Table 1) of static flows by applying all three algorithms: AVG, EDF, and SWIM. The performance comparisons are summarized in Table 6.

Table 1. Example I: Static Flows

	C1	C2	C3	C4	C5
DataSize (bytes)	540	80	900	120	600
Period (frame)	3	4	6	6	12

Table 1 shows a simple example of 5 connections (C1 through C5) and their demands (*DataSize*) in bytes and period in terms of WiMAX frames. The total allocated UGS slots are 420 bytes per frame (540/3) + (80/4) + (900/6) + (120/6) + (600/12). With all three algorithms, within one LCM cycle (12 frames in this example), the throughput is optimal, that is, (540×4) + (80×3) + (900×2) + (120×2) + (600×1) = 5,040 bytes or it is equal to 420×12 = 5,040 bytes.

Tables 2 and 3 show the allocations using AVG and EDF algorithms respectively. In AVG, the resource is allocated equally in every frame, e.g., 180 bytes in every frame for C1, 20 bytes for C2, and so on.

Table 2. Example I: AVG Allocation

Time	C1	C2	C3	C4	C5	Sum
0	180	20	150	20	50	420
1	180	20	150	20	50	420
2	180	20	150	20	50	420
3	180	20	150	20	50	420
4	180	20	150	20	50	420
5	180	20	150	20	50	420
			……..			

In EDF, the resource is allocated to the connection whose deadline is earliest. At the beginning, C1 has the earliest deadline, that is, 3 frames. In the first frame, the EDF scheduler allocates the entire available capacity of 420 bytes to C1. In the next frame, the scheduler allocates the remaining 120 bytes for C1 to meet C1's throughput guarantee (540 bytes). Of the left-over 300 bytes, 80 and 220 bytes are allocated for C3 and C2, respectively, because the deadlines of C3 and C2 are 4 and 6 frames.

Table 3. Example I: EDF Allocation

Time	C1	C2	C3	C4	C5	Sum
0	420					420
1	120	80	220			420
2			420			420
3	420					420
4	120		260	40		420
5		80		80	260	420
6	420					420
7	120		300			420
8			420			420
9	420					420
10			80		340	420
11	120	80	100	120		420

In SWIM, we initialize the allocation table with equal allocation. This results in allocations shown in Table 2 for AVG. The swapping steps of SWIM are shown in Table 4. In the first frame, the *max-res* connection is C1 and the *min-res* connection is C2. Therefore, C2's allocation in the frame is given to C1 and taken back in the second frame. This results in C1 obtaining 180+20=200 and C2 obtaining 20-20=0 in the first frame. C1 obtains 180-20=160 and C2 obtains 20+20=40 in the second frame. The resulting allocations are shown in Table 4a. Thus, swapping has reduced the number of bursts by one (one less burst in the first frame while still meeting all the throughput and delay guarantees for all sources).

In the next swapping step, C1 and C4 swap their allocations in frame 1 and 2 resulting in allocations shown in Table 4b. Next, C1 and C5 swap their allocations in frame 1 and 2 resulting in allocations shown in Table 4c. Next C1 and C3 swap in frames 1 and 2. However, in this case, C1 has only 90 units of allocations in 2nd frame and so the swap is done in two steps. In the first step, 90 units are swapped between C1 and C5 in frames 1 and 2. Then, the remaining 50 units are swapped in frames 1 and 3. This results in allocations shown in Table 4d. At this point, the allocation for the first frame is complete since there is only one burst left in this frame. Continuing these processes for the 2nd frame and other subsequent frames result in the final allocations shown in Table 5.

Table 4. Example I: SWIM Initial Steps

(a)

Time	C1	C2	C3	C4	C5	Sum
0	**200**		150	20	50	420
1	**160**	**40**	150	20	50	420
2	180	20	150	20	50	420

……..

(b)

Time	C1	C2	C3	C4	C5	Sum
0	**220**		150		50	420
1	**140**	**40**	150	**40**	50	420
2	180	20	150	20	50	420
			……..			

(c)

Time	C1	C2	C3	C4	C5	Sum
0	**270**		150			420
1	**90**	**40**	150	**40**	**100**	420
2	180	20	150	20	50	420
			……..			

(d)

Time	C1	C2	C3	C4	C5	Sum
0	**420**					420
1		**40**	**240**	**40**	**100**	420
2	**120**	20	**210**	20	50	420
			……..			

Table 5. Example I: Final Allocations of SWIM

Time	C1	C2	C3	C4	C5	Sum
0	420					420
1			420			420
2	120	20			280	420
3	360	60				420
4			420			420
5	180	60	60	120		420
6	420					420
7		20	400			420
8	120				300	420
9	420					420
10			420			420
11	120	80	80	120	20	420

The mean delays for both AVG and SWIM are the same. These delays are equals to the periods: 3, 4, 6, 6, and 12 frames for connection 1 through 5, respectively. For EDF, the mean delays are {(2+2+2+3)/4}=9/4, {(2+2+4)/3}=8/3, {(5+6)/2}=11/2, {(6+6)/2}=6, and 11 frames for connections 1 through 5, respectively.

Both AVG and SWIM have zero mean delay jitter, i.e., all SDUs are received on the period. For EDF, the mean delay jitters are {(0+0+1)/3}=1/3, {(0+2)/2}=1, 1, 0, and 0 for connections 1 through 5, respectively.

Consider the number of bursts: AVG gives 5 connections × 12 frames or 60 bursts, 24 bursts for SWIM, and 23 bursts for EDF. All four performance metrics are

summarized comparatively in Table 6.

Table 6. Performance Comparisons of UGS scheduling disciplines

	Mean Delay	Mean Delay Jitter	#Bursts	Throughput
AVG	Period	Zero	High	Optimal
EDF	Low	**Variable**	Low	Optimal
SWIM	Period	Zero	Low	Optimal

4.2 Fractional Resource Demands

Since both AVG's final allocation and SWIM's initial allocation are obtained by dividing the resource demand by the period, this can result in fractional allocations. To show this, we change the resource demands of C1 and C4 in the previous example to 500 and 200, respectively. With a period of 3, C1 requires (500/3) bytes per frame. Similarly, C4 needs 200/6 bytes per frame. With fractional allocations, we simply round the allocations in a frame after the frame has been completely allocated. We find that two decimal digit representations ($1/100^{th}$) are generally sufficient to avoid any truncation errors. Table 7 shows the final SWIM allocations for the example. The allocation is still feasible and results in 26 bursts.

Table 7. Example II: SWIM with DataSize/Period is prime

Time	C1	C2	C3	C4	C5	Sum
0	420					420
1			420			420
2	80	7		33	300	420
3		73	347			420
4	420					420
5	80	40	133	167		420
6	420					420
7		40	380			420
8	80			41	299	420
9			420			420
10	420					420
11	80	80	100	159	1	420

4.3 Dynamic Connections

It is common for flows to join or leave the network. For AVG, a newly admitted flow does not affect the current flows as long as the sum of the total pre-allocated resources and the resource demand per frame of the new connection is less than the total available resources per frame.

The above statement also holds for SWIM and EDF. However, the scheduler needs to maintain the flow states such as how many resources have already been allocated to

each connection. We illustrate this with an example. Suppose a new connection C6 joins the network at time 15 with a resource demand of 500 bytes over a period of 4 frames. At 15th frame, the total resource demand changes from 420 to 545 bytes per frame. Table 8 shows the initial allocation process for SWIM. The allocations from time 0 to 10 are the same as that in Section 4.1, Table 5.

At the end of 14th frame, the allocations for the five connections are 540, 20, 420, 0, and 280 bytes. Also, a connection C2 has an allocation of 60 bytes in 15th frame. This was a result of previous swapping. In Table 8, this type of pre-allocation (which has changed from initial value due to swapping) is indicated by enclosing it in parentheses. The pre-allocations for other connections and other frames at the end of 14th frame are also shown in the table by enclosing the allocations in parentheses.

Table 8. Example III: SWIM Initial Steps

(a)

Time	C1	C2	C3	C4	C5	**C6**	Sum
…..							420
11	120	80	80	120	20		420
12	420						420
13			420				420
14	120	20			280		420
15	180	(60)	(180)	(0)	(0)	125	545
16	180	20	150	(70)	(0)	125	545
17	180	20	150	(50)	(20)	125	545
			……				

(b)

Time	C1	C2	C3	C4	C5	**C6**	Sum
15	**305**	60	180	0	0		545
16	**55**	20	150	70	0	**250**	545
17	180	20	150	50	20	125	545
			……				

(c)

Time	C1	C2	C3	C4	C5	**C6**	Sum
15	**485**	60		0	0		545
16		20	**205**	70	0	250	545
17	**55**	20	**275**	50	20	125	545
			……				

To meet their throughput guarantees, in their period containing 15th frame, connections C1 through C5 need 540, 40, 300, 0, and 0 bytes over and above their pre-allocations. So, the new initial allocations are made by equally dividing these remaining values by the remaining period. The final results after C6 joins in 15th frame are shown in Table 9.

Table 9. Example III: SWIM with a new admitted flow C6

Time	C1	C2	C3	C4	C5	**C6**	Sum
…..							420
11	120	80	80	120	20		420
12	420						420
13			420				420
14	120	20			280		420
15	485	60		0	0		545
16			170		0	375	545
17	55	60	310	120			545
18	420					125	545
19		20	525				545
20	120					425	545
21	539				6		545
22			374		96	75	545
23	1	80	1	120	218	125	545

4.4 Deadline less than the Period

If the deadline is less than the period, the total demand before and after a connection's deadline is different and it is possible that some frames may be under-allocated. In other words, it is not possible to achieve full throughput. The unallocated resource can easily be used for non-time critical service classes. If we try to achieve full throughput with UGS traffic only, we may not be able to meet the deadline. This is true for all three algorithms as shown by the examples below.

Table 10. Example IV: Deadline < Period

	C1	C2	C3	C4	C5
Data Size (bytes)	540	80	900	120	600
Period (frame)	**3**	4	**6**	**6**	**12**
Deadline (frame)	**2**	4	**4**	**4**	**6**

In Example IV shown in Table 10, the deadline for connections C1 through C5 has been set to 2, 4, 4, 4, and 6 frames, respectively. If we allocate equal resources over all frames before the period (resource demand divided by the period), the allocation per frame is 420. We have full throughput but are missing the deadlines. If we allocate equal resources over all frames before the deadline (resource demand divided by the deadline), we need (540/**2**) + (80/4) + (900/**4**) + (120/**4**) + (600/**6**) or 645. This is over the available capacity of 420. Of course, if the admission control ensures that no new connections will be admitted if the sum of resources per frame (using resource/deadline) is more than the available capacity, we have a feasible solution and can meet the deadlines in a straightforward manner.

5 Conclusions

In this paper, we have introduced a new algorithm for UGS scheduler for the IEEE 802.16e Mobile WiMAX networks. The algorithm tries to minimize the number of bursts and gives zero delay jitter. Compared to AVG, the number of bursts is much less. Compared to EDF, the delay jitter is zero and the number of bursts is comparable. Although this technique has been designed for UGS service, we believe a simple extension with a polling mechanism can be used for ertPS service.

There is a tradeoff between delay jitter and the number of bursts. Thus, further study is needed to relax the tight delay jitter constraints and reduce the number of bursts.

In this paper, we assumed "Partial Usage of Subcarrier Utilization" permutation. In this case, all slots have the same capacity. With other permutations, such as adaptive modulation and coding (band-AMC) permutation, the slot capacity for each slot is different. We are working on an extension to handle this case. Finally, we have assumed the uplink allocation. The same algorithm can be extended for the downlink allocation with further optimization using extra information such as the actual arrivals, packet sizes, and head of line delays.

References

1. IEEE P802.16Rev2/D2.: DRAFT Standard for Local and metropolitan area networks: Part 16: Air Interface for Broadband Wireless Access Systems. Dec. 2007, 2094 pp.
2. So-In, C., Jain, R., Al-Tamimi, A.: Scheduling in IEEE 802.16e WiMAX Networks: Key Issues and a Survey. In: *IEEE Journal on Selected Areas in Commun.,* vol. 27, no. 2, pp. 156—171, Feb. 2009.
3. Cicconetti, C., Lenzini, L., Mingozzi, E., Eklund, C.: Quality of service support in IEEE 802.16 networks. In: *IEEE Network*s., vol. 20, no. 2, pp. 50--55, Apr. 2006.
4. Wongthavarawat K., Ganz, A.: IEEE 802.16 based last mile broadband wireless military networks with quality of service support. In: *Proc. Military Commun. Conf.*, 2003, vol. 2, pp. 779--784.
5. Sayenko, A., Alanen, O., Karhula, J., Hamalainen, T.: Ensuring the QoS Requirements in 802.16 Scheduling. In: *Proc Int. Conf. on Modeling Analysis and Simulation of Wireless and Mobile Systems.*, 2006, pp. 108--117.
6. Dong, L., Melhem, R., Mosse, D.: Effect of scheduling jitter on end-to-end delay in TDMA protocols. In: *Proc Int. Conf. on Real-Time Computing Systems and Applications.*, 2000, pp. 223--230.
7. Mansour Y., Patt-Shamir, B.: Jitter control in QoS networks. In: *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 492--502, Aug. 2001.
8. Verma, D.C., Zhang, H., Ferrari, D.: Delay jitter control for real-time communication in a packet switching network. In: *Proc. Commun. for Distributed Application and Systems*, 1991, pp. 35--43.
9. WiMAX Forum.: WiMAX System Evaluation Methodology V2.1. Jul. 2008, 230 pp. Available: http://www.wimaxforum.org/technology/documents