Efficient and Fair Scheduling of rtPS traffic in IEEE 802.16 Point-to-multipoint Networks

Zeeshan Ahmed, Salima Hamma

Efficient and Fair Scheduling of rtPS traffic in IEEE 802.16 Point-to-multipoint Networks

Zeeshan Ahmed
LUNAM Université, IRCCyN,
CNRS UMR 6597, Polytech′ Nantes,
rue Christian Pauc - BP 50609 -
44306 Nantes Cedex 3 FRANCE
zeeshan.ahmed@univ-nantes.fr

Salima Hamma
LUNAM Université, IRCCyN,
CNRS UMR 6597, Polytech′ Nantes,
rue Christian Pauc - BP 50609 -
44306 Nantes Cedex 3 FRANCE
salima.hamma@univ-nantes.fr

***Abstract*—IEEE 802.16 standard provides a revolutionary air interface that enables very high data rates over large distances. It incorporates a Quality of Service (QoS) framework to ensure satisfactory transmission of different classes of traffic. However, the actual implementation of QoS mechanisms is not defined in the standard and left out for service providers. One of the five different classes of services supported by the standard is real-time polling service (rtPS). Scheduling rtPS traffic is the most challenging because of its bursty nature and tight delay constraints. In this paper, we provide a new algorithm for fair scheduling of rtPS traffic in uplink direction. Besides the constraint of deadline which characterizes this type of traffic, we also assure fairness by using service ratios for each service flow. Performance analysis of the proposed algorithm proves that the algorithm is able to fairly allocate maximum possible bandwidth among all admitted rtPS connections. Furthermore, it shows improvement of various QoS parameters compare to Earliest Deadline First algorithm, particularly for high speed data networks.**

I. Introduction

Advancements in the Internet and mobile communication has resulted in a tremendous growth in their user base . There is also a continuous trend of increased usage of multimedia services, such as IPTv, video conferencing, VoIP etc. These services require huge resources and put enormous burden on network infrastructure. In fact, current cellular networks would not be able to cope up with the increasing demand within next few years [1]. Therefore, there is a need of faster and more reliable networks to support these services. Furthermore, to maintain an acceptable level of service, a network must also be able treat applications according to their priorities. In this regard IEEE 802.16 [2] (WiMAX) is an ideal choice. It is one of the leading candidates to become the official standard of next generation of cellular networks. It offers very high data rates over large distances. Moreover, it incorporates a well-defined QoS framework. The details of WiMAX QoS architecture are provided in section II. In a WiMAX point-to-multipoint topology, a base station (BS) controls and provides connectivity to multiple subscriber stations (SS).

Scheduling is an essential element of any QoS architecture that has to support real-time services with variable needs. The complex task of scheduling is performed by three schedulers in WiMAX i.e. BS downlink scheduler, BS uplink scheduler and SS scheduler. The functions of these schedulers are defined, however the details are not defined in the standard and left out for service providers [3].

The most complex scheduling is performed by BS uplink scheduler as it does not have complete view of service flows' queues that are maintained at SSs. Furthermore, scheduling rtPS traffic is the most challenging because of its bursty nature and tight delay constraints. In this article we propose an algorithm for BS uplink scheduler to provide fair and efficient allocation of resources to the set of active rtPS connections. The simulation results show that the proposed algorithm is efficient, fair, and practical.

The rest of the paper is organized as follows. Section II introduces the QoS architecture provided by the standard. Section III gives a brief overview of the related work. In section IV we present the details of our proposed algorithm. Section V provides simulation results and some comparisons with the related work and section VI concludes the paper.

II. QoS Architecture of IEEE 802.16 Networks

QoS refers to mechanisms that allow network managers to control the mix of bandwidth, delay, latency, and packet loss in a network in order to deliver an acceptable level of user experience. QoS support is a fundamental design requirement in WiMAX and it is considerably more difficult due to variable and unpredictable nature of wireless links. To support QoS differentiation, the standard provides five classes of services: (i) Unsolicited Grant Service (UGS): specifically designed for constant bit rate services, such as T1/E1 emulation and VoIP without silence suppression (ii) Extended Real-Time Polling Service (ertPS): built on the efficiency of both UGS and rtPS. Suitable for applications such as VoIP with silence suppression (iii) Real-Time Polling Service (rtPS): designed for real-time services that generate variable size data packets on periodic basis, such as MPEG video (iv) Non-Real-Time Polling Service (nrtPS): designed for delay tolerant services that generate variable size data packets on a regular basis (v) Best Effort (BE) Service: designed for applications without any QoS requirements such as HTTP service.

As discussed in section I, there are three schedulers specified in the standard: BS uplink scheduler, BS downlink scheduler, and SS scheduler. While the WiMAX framework provides the details of types of service flows and schedulers that should be supported, but it does not explicitly define the actual packet mechanisms at the schedulers. Therefore service providers are free to choose the scheduling algorithms according to their needs. The WiMAX QoS architecture is shown in figure 1.

The WiMAX MAC is connection oriented and therefore signaling messages need to be exchanged between BS and SS to establish a service flow, which is a MAC transport service that provides unidirectional transport of packets in either direction. All service flows are identified by a 32-bit service flow ID (SFID). Once a flow is admitted and activated, it is also assigned a 16-bit connection identifier (CID). A service flow is characterized by QoS parameters, which include details of how the SS requests uplink bandwidth allocations and how the BS uplink scheduler is expected to work.

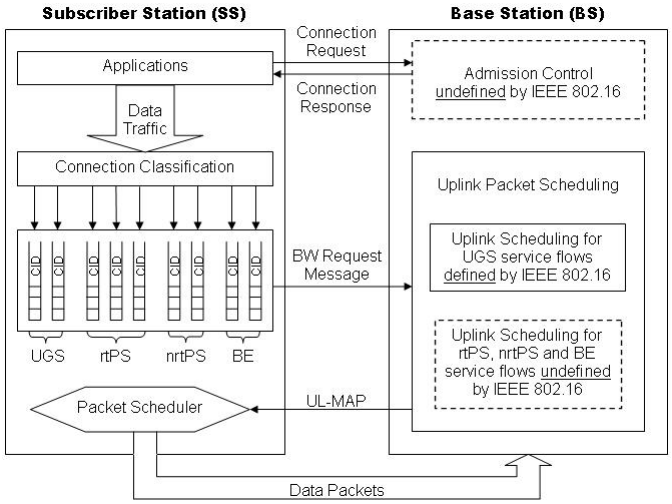


Fig. 1. WiMAX QoS Architecture [4]

The standard uses Time division multiplexing (TDM), in which the MAC frame is divided into an uplink subframe and a downlink subframe. The subframes are further divided into time slots. The BS controls the number of time slots assigned to a SS. The slots assigned to a SS in the uplink subframe allows the SS to transmit data in the uplink direction. The information about allotment of slots to SS in uplink subframe is communicated by the BS through UL-MAP at the start of each frame. The UL-MAP is a MAC management message that defines the uplink access for all SSs for entire uplink subframe.

For SS-initiated flow, a SS first requests a connection. The connection admission control (CAC) located in the BS determines whether the demanded QoS requirements may be fulfilled or not. An SFID is assigned to a flow only if sufficient resources are available to provide the required QoS. In a BS-initiated flow, in addition to above processes, the BS waits for the response from SS indicating whether it can support the requested communication. The classification of frames into SS transmission queues is done according to CIDs. The scheduler of a SS traverses the queues to select the most suitable packets for transmission, which are then transmitted in the time slots allotted to them by the BS uplink scheduler. The BS scheduler only grants bandwidth to SSs and not to individual CIDs. A SSs itself makes the scheduling decision regarding the service flows associated with it. This simplifies the overall scheme and increases effectiveness, as SSs have more updated views of their queues than the delayed estimates available at the BS.

III. Related Work

The scheduling algorithms proposed for WiMAX can be classified into two main categories: well-known algorithms and algorithms especially designed for WiMAX. These algorithms aim to optimize different performance parameters, such as: total maximum data rate, fairness, and latency. In [5], the authors apply following algorithms on rtPS traffic and present the results: round robin (RR), weighted round robin (WRR), maximum Signal-to-Interference ratio (mSIR), and temporary removal scheduler (TRS). The simulation results show that RR and WRR become inefficient in medium and high load conditions and transmit the minimum number of packets. TRS+mSIR and mSIR deliver the maximum number of packets. However they require a large average delay, which make them unsuitable for real-time applications such as VoIP and IPTv. The authors then present a modified version of mSIR, called mmSIR. But still the required average delay is greater than that of RR and WRR.

Some authors ([6], [7], [8]) suggest the use of Earliest Deadline First (EDF) for rtPS traffic. In [8], the use of EDF for rtPS is proposed in both uplink and downlink direction. Downlink rtPS traffic is given priority over uplink rtPS traffic. In [7], it is proposed to use the concept of arrival-service curve provided in [9] to predict the arrival time of incoming rtPS packets. We provide some comments on the use of arrival-service curve in section IV. In this scheme, If enough bandwidth is not available then the bandwidth is distributed among all rtPS connections according to their average data rates. However, this distribution can actually result in some unused portions of bandwidth as shown by simulations in section V.

In [10], the authors propose to use a single scheduler for all classes of services. They argue that scheduling disciplines like Fair Queuing (FQ) and EDF complicates scheduling and therefore they are not suitable for high speed networks. They further argue that the difficulty of accurately determining the deadlines of individual packets stored in SS buffers and potentially unfair behavior of EDF makes it unsuitable for WiMAX networks. However, their scheme is less efficient than EDF for rtPS flows.

IV. Proposed Algorithm

A. Terminology

Firstly, we present the important terminology to understand the rest of the article.

1) r_i^{min} : minimum reserved traffic rate (MRTR) for connection i
2) r_i^{max} : maximum sustained traffic rate (MSTR) for connection i
3) d_i : delay limit for connection i (number of uplink frames)
4) br_i^k: bandwidth requested by connection i in frame k
5) ba_i^k: bandwidth allocated for connection i in frame k
6) n : number of rtPS connections admitted
7) d_{max} : $max(d_i)$, where $i = 1, 2, ...n$
8) $bTbl$: an $n \times d_{max}$ table to store bandwidth allocations
9) r_k : unused bandwidth in frame k
10) f : current uplink frame
11) SR_i : service ratio for connection i
12) SR : total service ratio
13) r^a : current value of available uplink bandwidth

B. Call Admission Control

A new rtPS connection i is admitted by the BS if and only if enough bandwidth is available to guarantee MRTR for the connection, mathematically, $r_i^{min} \leq r^a$. After admitting i the value of r^a is updated, $r^a \leftarrow (r^a - r_i^{min})$. To ensure that connection i never surpass its contract, it is assumed that a traffic limiting module is present at the SS that always keeps the bandwidth demands of i below r_i^{max}. Thus the traffic generated by connection i always remain between r_i^{min} and r_i^{max}.

C. Fairness Parameters

For each uplink frame the BS allocates bandwidth to connections in increasing order of d_i, i.e. priority is given to the connections with tighter delay constraints. In order to guarantee fairness among rtPS flows we introduce a parameter, called *Service Ratio*, which is calculated for each flow as shown in Eq. 1. Another parameter, called *Total Service Ratio*, is also calculated as shown in Eq. 2. SR is simply the ratio of total service provided to total service requested by all connections. A connection i is only allowed to transmit data if $SR_i \leq SR$. If $SR_i > SR$, this implies that there are some connections for which the *Service Ratio* is less than SR and therefore they should be given priority over connection i. The idea is to guarantee MRTR for each session, while fairly distributing the available bandwidth among active rtPS flows. It should be noted that in the ideal scenario all connections would have equal value of *Service Ratio*. Mathematically, $SR_i = SR_j$, where $i, j = 1, 2, 3, ..., n$

$$SR_i = \frac{\sum_{t=1}^{f-1} ba_i^t}{\sum_{t=1}^{f-1} br_i^t} \tag{1}$$

where, $i = 1, 2, ..., n$

$$SR = \frac{\sum_{t=1}^{f-1} \sum_{i=1}^{n} ba_i^t}{\sum_{t=1}^{f-1} \sum_{i=1}^{n} br_i^t} \tag{2}$$

D. Scheduling

In each round bandwidth is allocated to a connection i only if $SR_i \leq SR$. When a new bandwidth request br_i^f arrives, the BS allocates maximum possible bandwidth to br_i^f in f. However if enough space is not available in f then it allocates the remaining bandwidth in frame $f + d_i$. The bandwidth allocations are done by using an $n \times d_{max}$ table, called $bTbl$. Each entry $bTbl[i][j]$ in $bTbl$ is an ordered pair (c, u), where c is the bandwidth that is allocated for connection i in frame j, while u is the bandwidth allocation to i which could be scheduled between frames f and $f + j$. However, there is no guarantee that the algorithm will actually schedule u. We call c as $confirm\ allocation$, while u as $unconfirm\ allocation$. The table is used by the scheduler to generate UL-MAP. At the end of each round of scheduling the first column of $bTbl$ is translated into UL-MAP. The generation of UL-MAP is illustrated at the end of this section. The step by step explanation of the proposed algorithm is presented in subsequent paragraphs.

The procedure *schedule*, lines 1-13, is invoked at the start of each frame to schedule rtPS traffic. The *for* loop, lines 2-11, runs the scheduling algorithm for each connection's bandwidth request. Thus, lines 3-10 are executed for each bandwidth request.

Line 4 tries to allocate the maximum possible bandwidth to br_i^f in the current frame. The function $allocateBw$ returns the amount of bandwidth successfully allocated in the current frame. So, this value is subtracted from the original requested bandwidth to obtain the amount of bandwidth still to be allocated. This bandwidth can be scheduled between $f + 1$ and $f + d_i$. Instead of finding the exact column in $bTbl$ to allocate this bandwidth, the algorithm tries to do the maximum possible allocation in frame $f + d_i$ (line 5). In fact, if some space would become available before $f + d_i$, this request could be scheduled earlier. If there is still some unallocated bandwidth, the algorithm assigns it as $unconfirm\ allocation$ at $bTbl[i][f + d_i]$. This step is actually done at line 8. If some space becomes available between frames f and $f + d_i$, this entry could be converted to $confirm\ allocation$. The condition at line 7 can be true either because $SR_i > SR$ and therefore no bandwidth allocation can be done for connection i or there is some bandwidth which cannot be allocated in statements 4 and 5. Regardless of the case, an $unconfirm\ allocation$ is performed in line 8. Then, the algorithm updates SR_i and SR according to equations 1 and 2.

The function $allocateBw$ is invoked when the procedure $schedule$ wants to allocate some bandwidth for a connection. The definition of $allocateBW$, lines 19-31, is self-explanatory

Algorithm 1 The proposed algorithm

1: **procedure** $schedule()$
2: **for** $i = 1$ to n **do**
3: **if** $SR_i \leq SR$ **then**
4: **set** br_i^f -= $allocateBw(br_i^f, i, f)$
5: **set** br_i^f -= $allocateBw(br_i^f, i, f + d_i)$
6: **end if**
7: **if** $br_i^f > 0$ **then**
8: **set** $bTbl[i][f + d_i].u$ += br_i^f
9: **end if**
10: update SR_i
11: **end for**
12: update SR
13: generate UL-MAP
14: **end procedure**
15:
16:
17: {Function allocateBw attempts to reserve an amount bw of bandwidth for the connection $conn$ in frame $frame$. It returns the amount of bandwidth successfully allocated}
18:
19: **function** $allocateBw(bw, conn, frame)$
20: **if** $bw \leq 0$ or $r_{frame} \leq 0$ **then**
21: **return** 0
22: **end if**
23: **if** $r_{frame} \geq bw$ **then**
24: **set** $allocate = bw$
25: **else**
26: **set** $allocate = r_{frame}$
27: **end if**
28: **set** r_{frame} -= $allocate$
29: **set** $bTbl[conn][frame].c$ += $allocate$
30: **return** $allocate$
31: **end function**

and it is provided for the sake of completeness. The function returns the amount of bandwidth it is able to allocate for the connection in the specified frame.

The generation of UL-MAP is straightforward. At the end of procedure *schedule*, the first column of $bTbl$ corresponds to the UL-MAP to be generated for the current frame. If there is unused bandwidth then $confirm\ allocations$ with the earliest deadline from subsequent frames are scheduled in the current frame. If no more $confirm\ allocations$ are available in subsequent frames and there is still some unused bandwidth then $unconfirm\ allocations$ can be scheduled in order of their deadlines. If there are some packets that cross their deadlines, it is proposed that these packets should be dropped by SS schedulers. The proposed algorithm is illustrated with the help of an example given at the end of this section.

The run-time complexity of the proposed algorithm is easy to calculate. Lines 3 to 10 are executed for each bandwidth request. Lines 4 and 5 call the function $allocateBw$. It is obvious that all steps in the function are done in constant time. Therefore, the complexity of $allocateBw$ is $O(1)$. Similarly, statements 7 to 10 are executed in constant time. Hence, for each bandwidth request, the run-time complexity of the proposed algorithm is $O(1)$.

$$br_k^f = backlog(f) + service(f-1) - backlog(f-1) \quad (3)$$

$$br_k^f = backlog(f) + service(f-1, f) - backlog(f-1) + drop(f-1, f) \quad (4)$$

It should be observed that as a bandwidth request is received, the algorithm decides how much $confirm\ \ allocation$ for this request could be done. Therefore there is no need for SS to send the backlog amount as a bandwidth request, but it can actually send the amount of traffic generated between $f-1$ and f. This have two distinct advantages: firstly less bandwidth is required to make requests, and secondly the BS need not to determine the deadline of individual packets through some complex procedure. However, even if the SS send the backlog as bandwidth request, we can use arrival-service curve [9] used by [7] to determine the actual new bandwidth demand generated during $f-1$ and f. Mathematically, it can be represented by equation 3, where $backlog(f)$ is the current bandwidth request, $backlog(f-1)$ is the previous bandwidth request, and $service(f-1)$ is the bandwidth allocated to the connection in the previous frame. However in the case, expired packets are dropped by SS, we must add the packets dropped during period $[f-1, f]$ to equation 3. The corrected version is given in equation 4.

Illustrating Example

We explain the working of the algorithm with the help of the example shown in figure 2. In this example, there are three connections to schedule: A,B,C with delay limits of 30ms, 40ms, and 60ms respectively. We assume total uplink bandwidth per frame to be 10 units and a frame duration of 20ms. This implies that a packet generated by A,B, and C between $f-1$ and f must be scheduled within next 1,2 and 3 frames respectively. The bandwidth requests generated by the three connections are shown in column 2. For example, the first entry in the first row of column 2, is the amount of traffic that arrived in the input queue of connection A for uplink transmission between frame 0 and frame 1. The bandwidth request for this traffic will be treated at the start of frame 1 by the BS. The third column shows the values of SR and SR_i at the start of scheduling frame f. An entry in the fourth column is the $bTbl$ that is obtained at the end of scheduling for frame f. The shaded entries in $bTbl$ are $unconfirm\ allocations$, while other are $confirm\ allocations$. The underline entries in a $bTbl$ are the allocations done during current frame. The UL-MAP corresponding to frame f is shown in the fifth column.

The scheduling in the example is done as follows. The algorithm is able to schedule the requested bandwidths in [0,1].

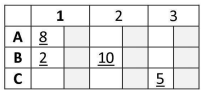
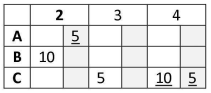
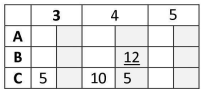
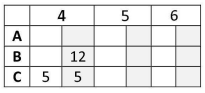
Duration	Bandwidth Requests			Service Ratios				Bandwidth Allocation Table	UL-MAP		
[f-1,f]	A	B	C	SR_A	SR_B	SR_C	SR		A	B	C
[0,1]	8	12	5	1.00	1.00	1.00	1.00		8	2	0
[1,2]	5	0	15	1.00	1.00	1.00	1.00		0	10	0
[2,3]	0	12	0	0.62	1.00	0.75	0.78		0	0	10
[3,4]	0	0	0	0.62	0.50	0.75	0.61		0	5	5
				0.62	0.71	0.75	0.70				

Fig. 2. An example for illustrating the proposed scheduling algorithm. The shaded entries in a Bandwidth Allocation Table are $unconfirm\ allocations$

Note specially the allocations done for connections B and C. Since only 10 units can be allocated to a frame, therefore we cannot do $confirm\ allocation$ of more than 10 units for a frame. For scheduling the requests in [1,2], $SR_A \leq SR$ but there is no bandwidth in current frame. Furthermore, due to delay limits this request cannot be fulfilled in subsequent frames. Therefore, it is entered as an $unconfirm\ allocation$ in the current frame. As there is no provision in the current frame, therefore this request is not scheduled in the UL-MAP of frame 2. In the duration [2,3], B requests 12 units of bandwidth. Since $SR_B > SR$, therefore the algorithm allocates it as an $unconfirm\ allocation$ in the frame $f + d_B$ i.e. in frame 4. The unused 5 units of bandwidth in frame 3 are used to schedule 5 units from the next frame. For [3,4], there is no bandwidth request. There are 5 units of $confirm\ allocation$ and 17 units of $unconfirm\ allocation$ for frame 4. Therefore, 5 units can be allocated to first $unconfirm\ allocation$ for B. The remaining entries cannot be scheduled. The final values of SR_i and SR are shown in the last row.

It is important to understand that all unexpired packets belonging to the same connection are always scheduled in the order of their deadlines by SS scheduler. The important thing is the amount of bandwidth allocated to the connection and not the actual packets against which the allocations are done. This is due to the fact that SS scheduler transmits packets in FIFO order. Consider the example given in Fig 3. We assume two connections A and B, with $d_A = 2, d_B = 2$. Note that the BS grant 5 units to A against demand of 10 units. However, the SS scheduler schedules the packet at the head of A's queue. Note, however, the 5 units were granted against the second packet in queue and not the packet at the head of the queue.

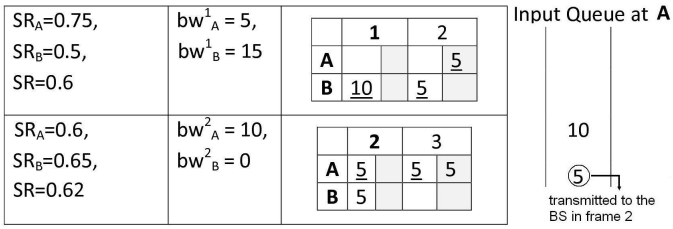


Fig. 3. FIFO Scheduling at SS

Parameter	Value
Frequency bandwidth	20 MHz
Sampling factor	8/7
Cyclic prefix	8
Frame duration	20 ms
Simulation duration	100 s
Packet reception model	PHY802.16 reception
Antenna model	omnidirectional
Antenna height	1.5 m
Antenna gain	1
Antenna loss	0
Transmit power	20 dBm
Link adaptation	Enabled

TABLE I
PARAMETERS OF THE SIMULATION MODEL

V. SIMULATION RESULTS

The performance of the proposed algorithm is evaluated by simulations in Qualnet 5 [11]. It is assumed that: (1) Packets arrive at start of a frame. (2) There is only rtPS traffic (3) All connections are already admitted. (4) Total uplink bandwidth = 10 Mbps (5) Four rtPS connections with parameter as shown in Table II. These parameters imply a very heavy load on system as average input traffic is more than twice the total available bandwidth.

Connection	r^{min} (kbps)	r^{max} (kbps)	Max Delay (frames)
A	2500	4000	3
B	2000	9000	4
C	3000	17000	5
D	2500	10000	7
Total	10000	40000	

TABLE II
INPUT TRAFFIC PARAMETERS

Figure 4 shows the service ratios for each rtPS connection as well as total service ratio SR after scheduling by the proposed algorithm. It is obvious that service ratios of all rtPS connections tend to SR. Even though the available bandwidth could only provide minimum guaranteed service to each rtPS, the proposed algorithm performed very well and dynamically allocate bandwidth to ensure fairness. In fact, SR is the best a connection can get and all the connections seem to follow SR rather well. Hence, it shows that the algorithm is able to fairly allocate maximum possible bandwidth to each admitted rtPS connection.

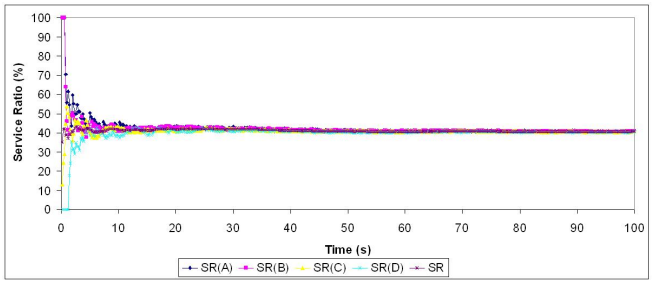


Fig. 4. Service ratio for rtPS connections by applying the proposed algorithm

We also performed simulations for the EDF algorithm proposed in [7]. Figure 5 shows the service ratios obtained by scheduling using EDF on the same set of data. There is not much difference between SR provided by EDF and SR provided by our proposed algorithm. However, obviously there is greater difference among the SR_i for individual connections. In this case, EDF allocates the maximum bandwidth to A, which least bandwidth is allocated to B. This dispersion in *service ratios* is due to the fact that EDF tries to minimize the average delay but does not take fairness into account.

Figure 6 shows the throughput as a function of load. Clearly both algorithm are able to schedule all traffic until the load surpasses the available bandwidth of 10 Mbps. After this point, no matter how much load is applied the algorithms cannot give more throughput. However, EDF tends to drop some packet and throughput is slightly less than 10 Mbps. In case of insufficient bandwidth, EDF distributes bandwidth among rtPS connections according to their average data rates. Sometimes, these allocations may result in over allocation to certain connections and thus some bandwidth remain unused.

Figure 7 depicts average delay packets have to wait in the input MAC queues as a function of load. Under light and medium load conditions the packets are scheduled almost

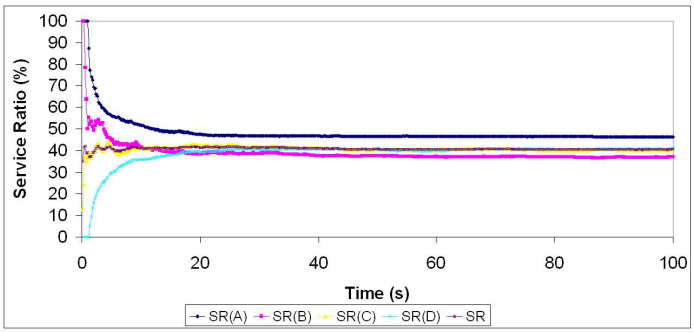


Fig. 5. Service ratio for rtPS connections by applying EDF

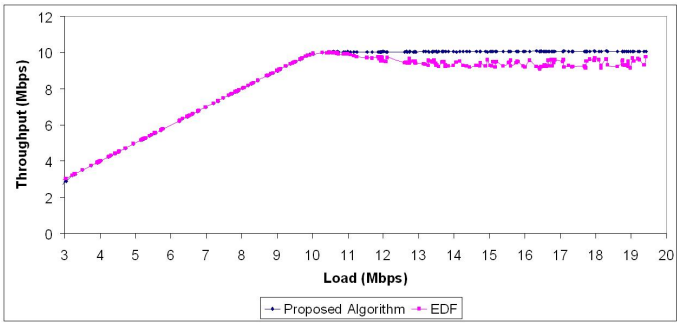


Fig. 6. Throughput vs Load

immediately by both algorithms. However, under very heavy load conditions the packets have to wait four times the normal average waiting time. Note that expired packets are automatically dropped by SS. Under all loads, the proposed algorithm results in slightly lower average delays.

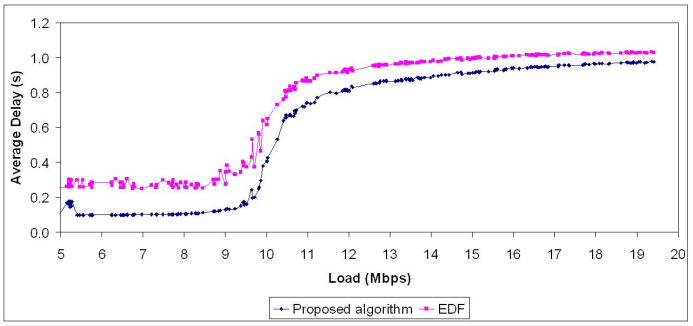


Fig. 7. Average Delay vs Load

Figure 8 shows the ratio of loss packets as a function of load. When the load is under 10 Mbps, both algorithms are able to schedule almost all the input packets and therefore the *loss packet ratio* is almost 0. Any traffic above 10 Mbps threshold cannot be scheduled and therefore the *loss packet ratio* increases sharply after this point. It can be seen that at a load of 20 Mbps, half of the traffic is dropped and so the *loss packet ratio* is around 0.5. It is obvious that the EDF results

in more lost packets as compared to the proposed algorithm.

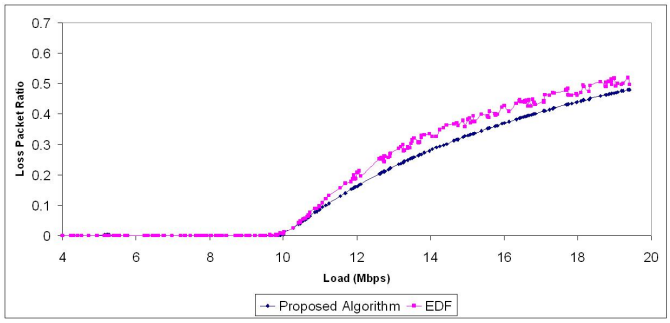


Fig. 8. Loss Packet Ratio vs Load

VI. Conclusion

In this paper, we propose an algorithm for IEEE 802.16 networks that provides fair uplink scheduling of rtPS traffic. The service ratio (SR) for each rtPS connection as well as total SR are computed by the proposed algorithm in order to implement fairness. Simulation results show that the proposed algorithm is able to fairly allocate maximum possible bandwidth to each admitted rtPS connection. Furthermore, we did comparative studies with EDF algorithm. The studied performance indicators are throughput, delay and loss packet ratio. For each parameter, our algorithm provides better results. It is also important to note that the run-time complexity is of the order of $O(1)$. The future work would include other classes of services (i.e UGS, nrtPS, BE). This will allow our algorithm to provide differentiated QoS needs for all service classes.

References

[1] WiMAX Forum, "Deployment of Mobile WiMAX networks by operators with existing 2G and 3G networks," March 2008. [Online]. Available: http://www.wimaxforum.org/sites/wimaxforum.org/files/document_library/deployment_of_mobile_wimax.pdf

[2] IEEE, "IEEE 802.16-2005, IEEE standard for local and metropolitan area networks - Part 16: Air interface for fixed and mobile broadband wireless access systems amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands," February 2006.

[3] R. A. Talwalkar and M. Ilyas, "Analysis of quality of service (QoS) in WiMAX networks," *16th IEEE International Conference, ICON08*, pp. 1–8, December 2008.

[4] F. Ohrtman, *WiMAX handbook*. McGraw-Hill Communications, 2005.

[5] A. Belghith and L. Nuaymi, "Comparison of wimax scheduling algorithms and proposals for the rtps qos class," *Wireless Conference, 2008. EW 2008. 14th European*, pp. 1–6, June 2008.

[6] T. Tsai and C. Wang, "Routing and admission control in IEEE 802.16 distributed mesh networks," *IFIP Internation Conference on Wireless and Optical Communications Networks*, pp. 1–5, 2007.

[7] K. Wongthavarawat and A. Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems," *Internation Journal of Communication Systems*, no. 16, pp. 81–96, 2003.

[8] J. Chen, W. Jiao, and Q. Guo, "An integrated QoS control architecture for IEEE 802.16 broadband wireless access systems," in *IEEE Global Telecommunication Conference*, IEEE Communications Society, Ed., 2005.

[9] R. Cruz, "A calculus for network delay, part i: Network elements in isolation," *IEEE Transaction of Information Theory*, vol. 37, no. 1, pp. 114–131, 1991.

[10] A. Sayenko, O. Alanen, and T. Hmlinen, "Scheduling solution for the ieee 802.16 base station," *Computer Networks*, vol. 52, no. 1, pp. 96 – 115, 2008, (1) Performance of Wireless Networks; (2) Synergy of Telecommunication and Broadcasting Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912860700271X

[11] Qualnet. [Online]. Available: http://www.scalable-networks.com/products/qualnet/