Admission Control and Flow Scheduling for IEEE 802.16 WiMAX Networks with QoS Requirements

Kalyana Chakravarthy Chilukuri
Department of Computer Science and Engineering
M.V.G.R. College of Engineering,
Vizianagaram, A.P., India

Prasad Reddy P.V.G.D
Department of Computer Science and Systems Engineering
Andhra University College of Engineering(A), Visakhapatnam, A.P., India

ABSTRACT

Multihop WiMAX networks show a lot of promise as the last mile in broadband connections for streaming audio and video to users. Such WiMAX networks are based on IEEE 802.16, which specifies the service classes for Quality of Service (QoS), but leaves the admission and scheduling mechanisms open. This paper presents a flow admission control and scheduling scheme for multihop WiMAX networks based on IEEE 802.16. Our admission control and scheduling scheme ensures that the various QoS parameters for the 802.16 service classes are met. We present our flow admission control and scheduling scheme and simulation results for this scheme. We then compare it in terms of QoS provided, with a simple admission and scheduling scheme and an admission scheme proposed by Ghosh et al. in [1]. Simulation results show that our scheme indeed guarantees the QoS parameters (minimum assured bandwidth, maximum allowable packet delay, maximum allowable packet jitter) needed by the different service classes of 802.16.

Keywords

Scheduling;Admission control;Quality of Service

1. INTRODUCTION

With an increasing number of service providers using WiMAX to provide wireless broadband services to their customers, WiMAX is becoming increasingly popular. While single hop WiMAX networks do exist, multihop WiMAX networks are more prevalent. A multihop WiMAX network typically has a Base Station(BS) which can communicate with several subscriber stations (SS) in one or more hops. The subscriber stations communicate with each other and with the customers in their range. Since most of the applications are based on streaming audio or video, gaming, etc., QoS is of great importance in these networks.

IEEE 802.16 classifies service into five categories based on the various QoS parameters to be satisfied. The five classes are - UGS, rtPS, ertPS, nrtPS and BE. The QoS requirements of each class are given in Table I. Flow admission control and scheduling while ensuring these QoS constraints and avoiding interference is not specified by the IEEE 802.16 standard and is open to research. In this paper, we present a flow admission control and scheduling mechanism that ensures interference-free scheduling that satisfies all the QoS constraints as well. Our scheme tries to maximise the number of flows admitted while satisfying the QoS and interference constraints. In addition to the percentage of accepted flows, we use the Schedule Efficiency (SE) metric introduced by [1] compare our scheme with the others.

2. RELATED WORK

Several admission control and scheduling algorithms for IEEE 802.16 have been proposed in the recent past. [4] discusses the performance of scheduling in IEEE 802.16 based wireless mesh networks. [5] provides an insight into the scheduling framework presented in the IEEE 802.16 standard. In [6], the authors propose a priority-based fair scheduling algorithm for subscribing stations to serve a mixture of uplink traffic from different scheduling services and provide an analytical model for evaluating user-perceived delay performance under this scheduling scheme. [7] proposes a dynamic allocation algorithm along with a measurement based CAC algorithm for providing delay guarantees to real-time media flows in IEEE 802.16e flows. In [8], the authors present several schemes for admission control for connections with QoS requirements over multihop wireless backhaul.

In [9], the authors evaluate the weighted round robin based scheduler to deal with packet transmission. [10] considers various practical issues while scheduling and resource are done for the downlink of a cellular OFDM system. [1] proposes a scheme where some of the QoS parameters are met by scheduling a flow close to its deadline. The QoS parameters met by this algorithm are the minimum bandwidth and delay. However, whether the delay is updated as data jumps from hop to hop is not clear. Also, jitter is not considered by the authors.[2] discusses a Linear program model for scheduling priority-based flows using start from Frame(SFB) beginning heuristic. In [3] authors propose an OFDMA Relay Scheduler (ORS) algorithm for scheduling traffic for every MS/RS in each scheduling period.

Table 1. QoS requirements of IEEE 802.16 traffic classes

Class	Application	QoS Parameters
Unsolicited Grant Service (UGS)	VoIP,E1; fixed size periodic packets	max rate, latency and jitter
Real-Time Polling Service(rtPS)	Streaming video/audio	min rate max rate and latency
Enhanced real-time Polling Service (ertPS)	VoIP with activity detection	min rate, max rate latency and jitter
Non real-time Polling Service	FTP	min rate and max rate
Best Effort (BE)	Data transfer, Web	maxrate

2.1 Basic Algorithm

The scheduling algorithm is used to schedule flows in each scheduling period, which consists of an integral number of frames. The same schedule is followed by each frame in the scheduling period. We consider only the DL part of the subframe, but scheduling in the UL part of the frame can be done similarly. The maximum and minimum number of time slots and subchannels required by each flow is calculated as follows. Let β be the number of subchannels per time slot and b be the bandwidth of each subchannel. Let α be the number of time slots in the DL part of the frame. Hence, the bandwidth available in each time slot is βb and the bandwidth available in each frame is αβb. The most basic flow scheduling algorithm considers flows in the order of their priority class and schedules them in the first available slot and subchannel, till the minimum bandwidth requirement of the flow is satisfied. Hence, first all UGS flows are scheduled, then all rtPS flows are scheduled and so on, with the BE flows being scheduled in the end. This method satisfies the minimum bandwidth requirement of the flows.

All flows are accepted, since the bandwidth available at any SS or the BS is the sum of the minimum bandwidths of all the flows, but all the accepted flows do not satisfy the delay requirement of the class to which they belong. Jitter is negligible because each flow is allotted subchannels in consecutive time slots.

Algorithm 1. Basic Algorithm

```
1:  for all links l in F do
2:      slotsalloc = 0
3:      finalslot = starttimeslot of link l
4:      Ts = startOfFrame
5:      subchannelCtr = 0
6:      while (Ts ≤ finalslot) and (slotsalloc < minslots) do
7:          if subchannel[subchannelCtr] is free then
8:              slotsalloc++
9:              update tempschedule
10:         end if
11:         subchannelCtr++
12:         if subchannelCtr = MAXSUBCHANNELS then
13:             subchannelCtr = 0
14:             Ts=Ts+2
15:         end if
16:     end while
17:     if slotsalloc < minslots then
18:         Scheduled = FALSE
19:         return Scheduled
20:     end if
21: end for
22: Scheduled = TRUE
23: return Scheduled
```

2.2 Schedule Flow Subchannel Algorithm

The SFS algorithm proposed by [1] is an improvement over the basic algorithm as it schedules flows such that their delay requirements are met. [1] defines a virtual link to be a physical link associated with a particular flow. Thus, a virtual link is identified by the link id and the flow id and has a number of parameters like the start time slot. Given a number of virtual links to be scheduled and their routes, SFS finds the start time slots of the virtual link as follows. The start time slot is the first time slot in the subframe by which the link must be scheduled, so that the flow reaches the destination by its deadline. The start time of each link is calculated as below:

$$t_s = \left(\frac{d - f_s - kt_1}{t_2}\right) \quad (1)$$

where t_s is the start time slot of a link, d is the deadline of flow, f_s is start time of the next frame, k is the number of hops to the destination, t_1 is the propagation delay and t_2 is the period of each time slot. The start time slot of a link also depends on whether the link is odd or even. The above expression is used to calculate the start time slot for those classes that have a delay requirement, i.e., UGS,ertPS and rtPS. For those classes that do not have any latency requirement, the last slot of the DL frame is considered as the start time slot. The flow is then scheduled at t_s if there are empty subchannels at t_s, then at t_{s-1} (again if t_{s-1} has empty subchannels) and so on till the first time slot, till the bandwidth requirement of the flow is met.

Scheduling in SFS is done in two phases. First, SFS attempts to schedule free time slots and subchannels to the flow and find the maximum bandwidth that can be allotted

all the links of the flow. If any link has been allotted more than the required minimum bandwidth of the flow, the extra timeslot-subchannels are tagged so that they can

be pre-empted by other flows if required. If the bandwidth allotted to a link is less than its minimum bandwidth requirement, the second phase (ScheduleFlowExtra) is executed, which involves discovering previously tagged slot-subchannels and assigning them to the flow being considered. The flows are scheduled in an interference-free manner such that both primary and secondary interference are accounted for by the interference model. Thus, two interfering links are never scheduled in the same timeslot-subchannel. Also, even links transmit in even time slots only and odd links transmit in odd time slots only.

Algorithm 2. Scheduling as per SFS

```
1: for each flow f  in F do
2:    bwAlloc = SFS(maximum bandwidth required by f)
3:    if bwAlloc < min bandwidth requirement of f then
4:       Scheduled = ScheduleFlowExtra()
5:    end if
6:    if bwAlloc > min bw or Scheduled is TRUE then
7:       Tag Extra slots
8:       Make Temporary Schedule Permanent
9:       Update bandwidth allocated to flow
10:      Accept flow f
11:   else
12:      Reject flow f
13:   end if
14: end for
```

Algorithm 3. Schedule Flow SubChannel(bottleneckBw)

```
1: for each link l in f do
2:    Ts = start time slot for link l
3:    slotsalloc = 0
4:    while Ts > startOfFrame do
5:       numfreeslots = FindFreeSubchannels(Ts,l)
6:       if (slotsalloc + numfreeslots) < maxslots then
7:       Add numfreeslots to temporary schedule
8:       slotsalloc = slotsalloc + numfreeslots
9:       Ts = Ts - 2
10:      else
11:      Add (maxslots - slotsalloc) to temporary schedule
12:      slotsalloc = maxslots
13:      break
14:      end if
15:   end while
16:   update bottleneckBw
17: end for
18: return bottleneckBw
```

3. PROPOSED ALGORITHM Admission Control and Flow Scheduling Algorithm with QoS Guarantee (ACFS)

3.1 Network Model

In WiMAX, the wireless media is shared in the point-to-multipoint (PMP) or mesh operational modes. In the PMP mode which is the mode considered in this paper, each base station services several service stations and each service station in turn services several devices and can communicate with other service stations as well. Hence, flows are first centrally scheduled at the BS to satisfy the requests by the devices (coming to the BS via the SS). Once the flow reaches a downlink SS at the first level, it has to be scheduled by the SS to reach the device directly if the device is under this SS or to reach another downlink SS if the device is not directly under this first level SS. Hence, the network assumes a logical tree-like structure as shown in Figure 1.

3.2 Assumptions

We assume that the physical layer is Orthogonal Frequency Division Multiple Access(OFDMA), since this is supported by both IEEE 802.16 and IEEE 802.16e. OFDMA is a combination of time and frequency domain multiple access and hence, can be viewed as a time-frequency grid. A slot is the basic unit of time division in the OFDMA time-frequency grid. Frequency division is achieved by dividing the channel into subcarriers. One or more subcarriers can be grouped to form a subchannel. The basic unit of bandwidth allocation in the frequency domain is the subcarrier. Hence, allocation of bandwidth can be done with a finer resolution in OFDMA, resulting in greater utilisation of bandwidth. However, the scheduling overhead may be high, as the number of basic units that can be scheduled are more in OFDMA as compared to a scheme that uses pure time-division or frequency division and hence, the scheduling load may be higher. The other assumptions that we make are:

- The topology of the network is a tree with the BS at the root and the devices as the leaves.
- The set of interfering links is given as the input by the user, making our scheme independent of any specific interference method.
- Routing from any source to destination is fixed and non-adaptive.
- The flows are non-splittable i.e., they follow only one path all the way from the source to the destination.
- All the nodes are half-duplex, hence they can both

transmit and receive, but not at the same time. This is because the physical model has each time frame divided into a downlink (DL) part and an up link(UL) part. For interference-free scheduling, we adopt the odd-even framework as in [6]. A link in the network is either even or odd. An even link transmits in even slots of the downlink subframe and an odd link transmits in odd slots of the downlink subframe.

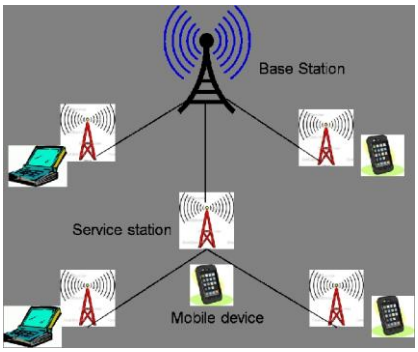

Figure 1. Topology of the Network considered

3.3 Algorithm

Assuming the above, we propose our admission control and flow scheduling algorithm with QoS guarantee which we call ACFS henceforth. Our algorithm aims to maximize the bandwidth utilization for a given number of flows with QoS requirements like minimum rate, delay and jitter requirements. The interference and routing patterns are assumed to be given as input. we first describe the two algorithms (the basic algorithm and the SFS algorithm) with which we compare ACFS. We consider a set of flow requests that have Poisson arrival. The set of flow requests is a random mixture of all classes of requests (UGS, rtPS, ertPS, nrtPS and BE). While ACFS can be used for static or dynamic scheduling, we consider static scheduling to highlight the merits of our algorithm. The number of flows we consider is such that the minimum bandwidth required by all flows put together is equal to the available bandwidth of the network.

Scheduling is done at each BS and SS. Each flow is scheduled at the BS and then at one or more SS based on its route. Hence, the BS has to schedule all the flows, while the number of flows to be scheduled at each SS reduces as we move down the routing tree shown in Figure 1. Since we scheduling is static, we check if a flow cannot be scheduled at any level due to delay, bandwidth or jitter requirements not being met. If so, it is rejected and no bandwidth is allotted to it at the BS or at any SS.

1) Dealing with the Minimum Bandwidth and Delay Con-straints: The algorithm proposed by is the admission control and flow scheduling algorithm with QoS guarantee. ACFS scores over the SFS algorithm because SFS ensures that the bandwidth and delay requirements are met, but does not consider the jitter requirement of a flow, which is crucial for audio/video applications. Also, how the delay is calculated for scheduling a flow at different levels of the routing tree is not mentioned clearly in SFS, as it just considers the "start time of the next frame" in the expression for t_s. In reality, the maximum delay experienced by a flow may vary anywhere from f, which is the frame time, to l-f, where l is the number of levels that the flow has to be scheduled at. For example, consider a flow f that has to be scheduled at the BS and at two levels of SSs below the BS.

If the BS is at level 1, and the flow has to be scheduled at levels 1, 2 and 3. Let $tmin_l$ be the minimum time slot allotted to the flow at level l and $tmax_l$ be the maximum time slot allotted to the flow at level l. The flow will take ≤ a frame time to reach the device if

$$tmax_p < tmin_{p+1} \text{ for all } p \in [1, l] \quad (2)$$

The maximum delay for the flow from the BS to the device will occur when

$$tmax_p \geq tmin_{p+1} \text{ for all } p \in [1, l] \quad (3)$$

If the deadline for a flow is d, the start slot is calculated at the BS as

$$t_s = \left(\frac{d - f_s - kt_1}{t_2}\right) \quad (4)$$

We define d_l to be the deadline that should be used to calculate the start slot at level l, using Equation 4. At the SS at the next level 2, the deadline d_2 is t_2 ($d_1 - tmax_1$). At the next level, the deadline has to be updated depending on how much time has elapsed since the start of the scheduling. If $tmax_p$ is < $tmin_{p+1}$, the time elapsed between the schedule of the flow at the l^{th} level and at the $(l + 1)^{th}$ level is just t_2 ($tmin_{p+1} - tmax_p$), since the flow is scheduled at both the level in the same frame. If $tmax_p \geq tmin_{p+1}$, the flow cannot be scheduled till the next frame at level $(l + 1)^{th}$ level after being scheduled at the l^{th} level. In this case, the time elapsed between the schedule of the flow at the l^{th} level and at the $(l + 1)^{th}$ level is ($f - t_2$ ($tmax_p - tmin_{p+1}$)). Hence, when a flow is being scheduled at an SS which is at level more than 2, the time elapsed till then has to be deducted from the initial deadline d, to get the deadline to be considered at that level, d_l. Hence if t_e denotes the time the time elapsed between the schedule of the flow at the l^{th} level and at the $(l + 1)^{th}$ level, for l > 2,

$$t_e = t_2 * (tmin_{p+1} - tmax_p) \quad \text{if } tmax_p < tmin_{p+1}$$
$$= f - t_2 * (tmax_p - tmin_{p+1}) \text{ if } tmax_p \geq tmin_{p+1} \quad (5)$$

The deadline for the lth level is dl, where

$$dl = d \quad \text{if l=1}$$
$$= d - t_2 - (dl - tmax_1) \quad \text{if l=2}$$
$$= dl - l - t_e \quad \text{if l>2}$$

After the deadline for that level is determined, the start slot for that level is calculated as below:

$$t_s = \left(\frac{dl - f_s - kt_1}{t_2}\right)$$

As in SFS, we scan backward from ts till slot 1, and if there is a timeslot-subchannel that is not already allotted, it is

allotted to a flow. This is process of scanning and allotting is repeated till the minimum bandwidth requirement of the flow is met.

2) Dealing with the Jitter Constraint

After all the flows have been allotted their minimum bandwidth (satisfying their delay constraint), this process is repeated to allot more bandwidth (up to the maximum bandwidth requirement) to each flow, while checking that the jitter does not exceed the maximum allowable jitter. For calculating the jitter, we consider the minimum allottable chunk of bandwidth of a flow, which is equal to the bandwidth of a timeslot-subchannel, b. Let us call this a BFU (Basic Flow Unit). Hence, the minimum number of BFUs of a flow is [bwmin / b]. Each ith BFU is scheduled at a timeslot-subchannel at each level. Note that packet jitter never occurs between two packets that belong to the same BFU, as these packets get scheduled at the same timeslot at all levels and hence experience the same delay. Jitter is experienced because of consecutive packets that belong to different BFUs, as these can be scheduled at different time slots at any level and thus can experience different delays. Let the time slot where the ith BFU is allotted at the ith level be denoted by τ_i . If the ith and (i+1)th BFUs are scheduled in the same time slot (in different subchannels) or are scheduled in consecutive timeslots at all levels, the difference in delay (jitter) between these two BFUs is zero. However, if the ith and (i+1)th BFUs of a flow get scheduled in timeslots such that they experience different delays, packet jitter occurs. The basic algorithm discussed in Section II-D each flow is allotted the first possible timeslot-subchannel starting from the beginning of the DL subframe. This effectively fills up the DL subframe sequentially from the beginning to the end of the DL subframe and consecutive BFUs always get scheduled in the same or in consecutive timeslots, resulting in zero jitter.

The basic algorithm gives no delay guarantee. The SFS algorithm on the other hand, provides delay guarantee, but can result in significant packet jitter. This is because in SFS, the start time slot is calculated and the DL subframe is scanned for empty timeslot-subchannels backwards from the start timeslot, which may result in consecutive BFUs not being allotted consecutive timeslots. These consecutive BFUs then experience different delays. Also, if a flow does get enough bandwidth in the first phase, the second phase (ScheduleFlowExtra) is executed, where the tagged timeslotsubchannels are discovered and allotted to the flow. This may again result in consecutive BFUs being allotted nonconsecutive timeslot-subchannels, resulting in different delays for consecutive BFUs. As consecutive BFUs may experience different delays at different levels, their cumulative delays by the time they reach the device may be very different from one another, resulting in significant packet jitter. ACFS solves this problem by attaching with each (ith) BFU the delay up to the lth level to be τ_i^l . Then, when the (i +1)th BFU is scheduled, we first calculate τ_{i+1}^l and then the difference between the two delays. Only if this does not exceed the jitter constraint, the flow is accepted. Otherwise, the flow is rejected. In this way, the flow is accepted only if it satisfies both the delay and jitter constraints.

Algorithm 4. Admission Control and Flow Scheduling (f, bw)

```
1:  /* n is the number of BFUs of f, bw_min is the minimum bandwidth requirement of flow f*/
2:  n = bw_min / bw
3:  p= number of levels at which f has to be scheduled
4:  for i=1 to n do
5:    for j=1 to p do
6:      τ^i = 0
7:    end for
8:  end for
9:  for each flow f in F do
10:   bwAlloc = SFS(bw)
11:   calculate overall delay for all BFUs allotted
12:   maxJitter=0
13:   for i= 1 to (n-1) do
14:     if τ^i ≥ maxJitter then
15:       maxJitter = τ^i
16:     end if
17:   end for
18:   if maxJitter < maximum allowable jitter then
19:     Update bandwidth allocated to flow
20:     Accept flow f
21:   else
22:     Reject flow f
23:   end if
24: end for
```

Algorithm 5. Scheduling as per Admission Control and Flow Scheduling

```
1:  /* N is the number of flows to be scheduled*/
2:  for i=1 to N do
3:    Admission Control and Flow Scheduling(i, minimum band- width required by i)
4:  end for
5:  for i=1 to N do
6:    if i is accepted then
7:      bwtobeallotted = (maximum bw required by i)-(minimum bw required by i)
8:      Admission Control and Flow Scheduling(i, bwtobeallotted
9:    end if
10: end for
```

4. EXPERIMENTAL EVALUATION OF THE PROPOSED SCHEME

We simulated the three algorithms described above using our custom simulator written in Java. Our simulator can perform both static and dynamic flow scheduling as it runs in two separate threads - one that generates flows and the scheduler that schedules these flows at after a scheduling period. The scheduling period id is configurable. The flows are generated by Poisson arrival process.We limit the number of flows so that the sum total of their minimum bandwidth requirement is less than or equal to the maximum bandwidth of the network. A fixed topology, as shown in Figure 1 and the interference model are provided as input by the user. The number of each class of flows is random and each type of flow is associated with a fixed

bandwidth, delay and jitter requirement, depending on the class to which it belongs.

We maintain five global queues at the BS ordered by 802.16 class priority, with UGS having the highest priority and BE having the least priority. Within each class queue, the flows are ordered in the order of arrival. Each flow is associated with a route generated by the random flow generator, depending on the topology considered. The schedule for the flows is calculated using each of the three algorithms - the basic algorithm, SFS and ACFS - and is stored as a two dimensional array of timeslots and subchannels. We considered a frame length of 5ms and 16 subchannels per time slot. We generated different traffic scenarios by generating different sets of flows. The flow mix in each set is varied from 0% of the network capacity for a particular class to 100% of that class, while the rest of the network capacity is divided equally among all the other four classes. We compared the algorithms based on several parameters, as described below.

A. Percentage of Accepted Flows

The percentage of accepted flows is a measure of the percentage of the total number of flows that are accepted. However, this measure does not consider the bandwidth utilization into consideration. Hence, since BE class has the least minimum bandwidth requirement (which is one time slot subchannel in our algorithm), an algorithm may schedule many BE flows and may result in a higher % of accepted flows, while an algorithm may allot fewer flows of a class that needs higher minimum bandwidth. As seen from Figures 2 to 6, the percentage of accepted flows in ACFS is more than that of the percentage of accepted flows in SFS. However, the percentage of accepted flows in the basic algorithm is more than that of ACFS, as the number of accepted flows of the basic algorithm includes those that do not satisfy the jitter and delay constraints, whereas the number of accepted flows of ACFS is strictly of those that satisfy all the delay and the jitter constraints. The difference in the percentage of flows accepted is not distinct when the number of nrtPS or BE flows increases (Figure 5 and Figure 6), as these classes have no jitter or delay constraints.

Table 2. Simulation Parameters for the Admission Control and Flow Scheduling algorithm

Simulation Interval	200 s
Frame Length	5 ms
Number of Subchannels / time slot	16
Number of Flows	5
Output link Bandwidth	10 MHz
Inter arrival time between packets	Poisson
Maximum number of slots/frame	26
DL/UL slots	26/21
Deadline	50 ms(for max. 3 Hops)
rtPS Jitter	<30ms
Simulation Software	Java

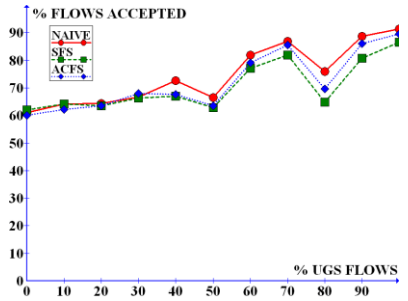

Figure 2. Effect of % of UGS flows on the % of flows

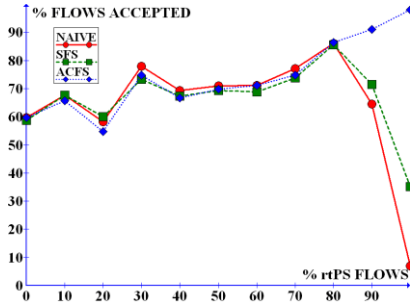

Figure 3. Effect of % of rtPS flows on the % of flows accepted

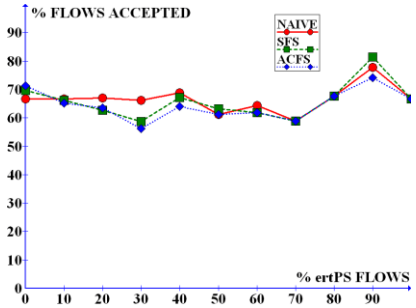

Figure 4. Effect of % of ertPS flows on the % of flows accepted

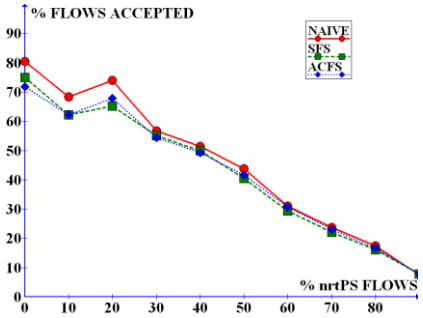

Figure 5. Effect of % of nrtPS flows on the % of flows accepted

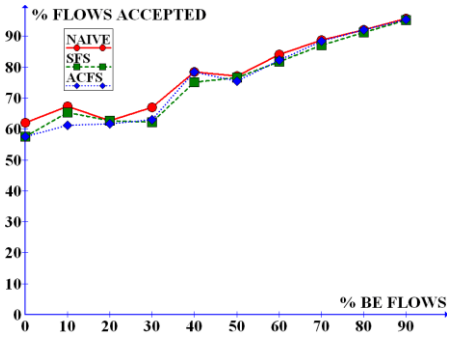

Figure 6. Effect of % of BE flows on the % of flows accepted

B. Schedule Efficiency

We compare the three algorithms also on the basis of the Schedule Efficiency(SE) metric defined by [1], as follows.

SE = [Ct / Ca] Here, C is defined as C = WUGS * bwUGS +WrtPS * bwrtPS +WertPS * bwertPS+WnrtPS *bwnrtPS +WBE *bwBE. The subscripts a and t refer to the accepted and total number of flows respectively, Wsub is the weight of class sub and bwsub is the minimum bandwidth of class sub. The subscript sub refers to the subclass type (UGS, rtPS, ertPS, nrtPS or BE).

An algorithm that schedules more flows of a higher priority class has more Schedule Efficiency than an algorithm that schedules more flows, but of a lower priority class. Figures 7 to 11 depict the performance of the three algorithms in terms of the Schedule Efficiency (SE). It can again be seen that when the flows contain more of higher priority traffic, the SE of ACFS is more than that of the SE or SFS. However, the SE of the basic algorithm is more than that of ACFS, as the number of accepted flows of the basic algorithm include those that do not satisfy the jitter and delay constraints, whereas the number of accepted flows of ACFS is strictly of those that satisfy all delay and jitter constraints.

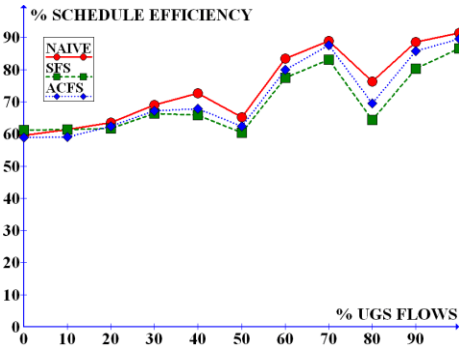

Figure 7. Effect of % of UGS flows on the Schedule Efficiency

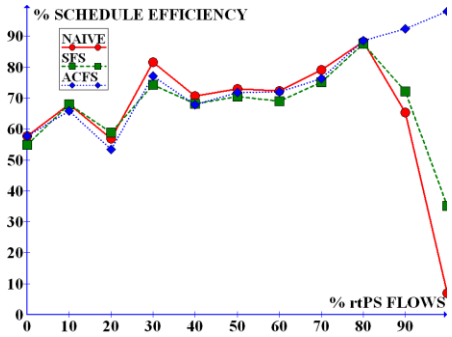

Figure 8. Effect of % of rtPS flows on the Schedule Efficiency

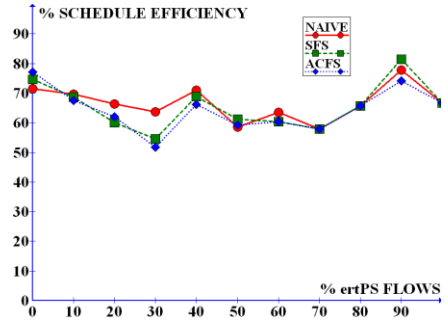

Figure 9. Effect of % of ertPS flows on the Schedule Efficiency

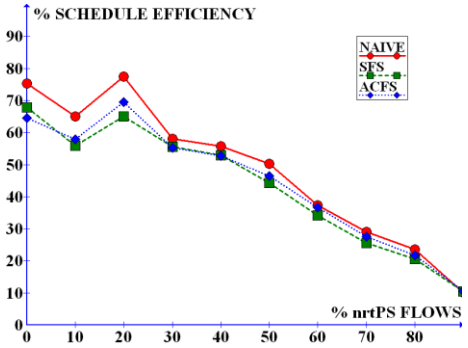

Figure 10. Effect of % of nrtPS flows on the Schedule Efficiency

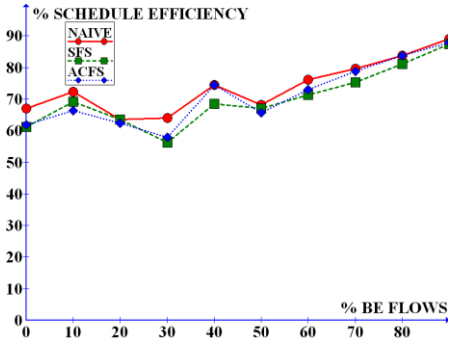

Figure 11. Effect of % of BE flows on the Schedule Efficiency

C. Maximum Packet Jitter

Figures 12 to 16 depict the maximum packet jitter(end-toend) experienced by rtPS flows with various mixes of flows. As can be seen from the graphs, the basic algorithm and SFS result in a varying packet jitter, where the jitter is quite large for some mixes of flows. ACFS is seen to always result in the minimum jitter, below the maximum allowable jitter for rtPS flows. similar results were obtained for packet jitter of UGS flows also. These results show how ACFS is superior to the basic and SFS algorithms in restricting the packet jitter. Since the delay is constrained and minimum bandwidth is also assured by ACFS, it provides all the aspects of good QoS required by the different flows of IEEE 802.16.

We have checked that the total bandwidth used using ACFS is almost equal to that of SFS. This is because both SFS and ACFS try to allot the maximum bandwidth requirement of the flow. While SFS first allots the maximum bandwidth and then tries to collect the tagged slots in favor of new flows, ACFS first allots the minimum required bandwidth and then tries to allot up to the maximum bandwidth, provided the jitter and delay requirements are met.

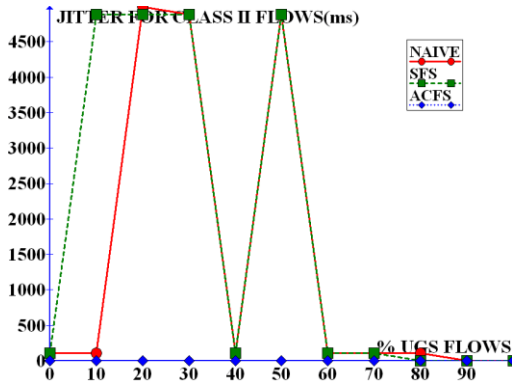

Figure 12. Effect of % of UGS flows on the max jitter of rtPS flows

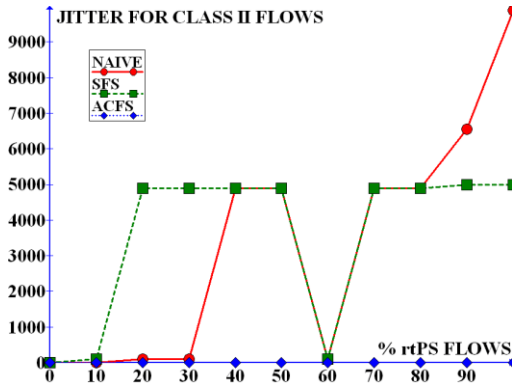

Figure 13. Effect of % of rtPS flows on the max jitter of rtPS flows

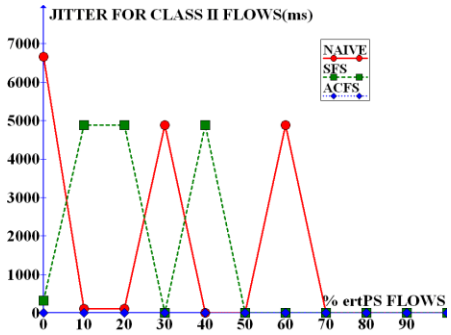

Figure 14. Effect of % of ertPS flows on the max jitter of rtPS flows

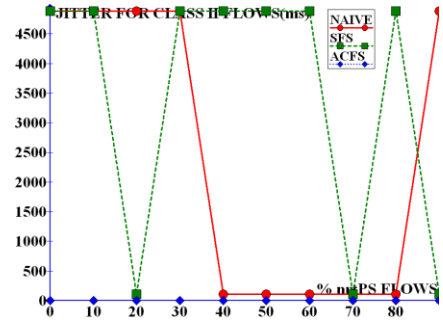

Figure 15. Effect of % of nrtPS flows on the max jitter of rtPS flows

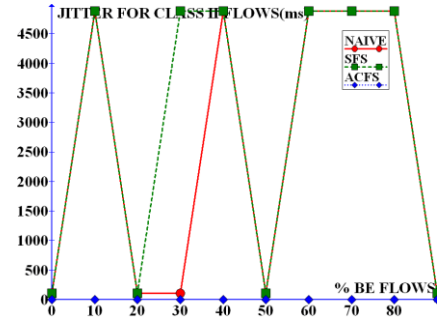

Figure 16. Effect of % of BE flows on the max jitter of rtPS flows

5. CONCLUSIONS

In this chapter, a new Admission Control and Flow Scheduling algorithm is proposed, that admits and schedules any mix of flows, such that all the IEEE 802.16 QoS class requirements are met. The algorithm proved better than the previously proposed SFS and a basic algorithm in that it provides delay, jitter and bandwidth guarantee. It also maximises the bandwidth utilisation, as it first allots the minimum required bandwidth by all flows and then tries to allot up to the maximum bandwidth for each flow, provided the delay and jitter requirements are met. ACFS is also seen to be superior to the other two algorithms in terms of the percentage of flows accepted.

The admission control mechanisms can be extended to Mobile multihop networks and to future 4G networks where scheduling is a far more challenging issue.

6. REFERENCES

[1] D. Ghosh, A. Gupta, and P. Mohapatra, "Admission control and interference-aware scheduling in multi-hop wimax networks," IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, vol. 0, pp. 1–9, 2007.

[2] A. Gupta, D. Ghosh, and P. Mohapatra, "Scheduling prioritized services in multihop OFDMA networks," IEEE/ACM Transactions on Networking(TON),vol.18 issue 6, pp. 1780–1792, 2010.

[3] D.Ghosh, A. Gupta and P.Mohapatra "Adaptive Scheduling of Prioritized Traffic in IEEE 802.16j wireless Networks",

WIMOB 2009, IEEE International Conference on, pp.307-313, 2009.

[4] B. Han, F. Tso, L. Ling, and W. Jia, "Performance evaluation of schedul- ing in ieee 802.16 based wireless mesh networks," IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, vol. 0, pp. 789–794, 2006.

[5] D. Ghosh, A. Gupta, and P. Mohapatra, "Scheduling in multihop wimax networks," SIGMOBILE Mob. Comput. Commun. Rev., vol. 12, no. 2, pp.1–11, 2008.

[6] Y. W. Chan, S. Zukerman, M. Harris, and R. Sch., "Priority-based fair scheduling for multimedia wimax uplink traffic," Communications, 2008. ICC '08. IEEE International Conference on, pp. 301–305.

[7] G. Boggia, P. Camarda, L. A. Grieco, and S. Mascolo, "Feedback based bandwidth allocation with call admission control for providing delay guarantees in ieee 802.11e networks," in in IEEE 802.11e networks, Computer Communications, Elsevier, 2005, pp. 325–337.

[8] S. L. Narlikar, G. Pal, M. Wilfong, and G. Zhang, "Admission control for multihop wireless backhaul networks with qos support," Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE, pp. 92–97, Volume: 1.

[9] C. Cicconetti, A. Erta, L. Lenzini, and E. Mingozzi, "Performance evaluation of the ieee 802.16 mac for qos support," IEEE Transactions on Mobile Computing, vol. 6, no. 1, pp. 26–38, 2007.

[10] J. Huang, V. Subramanian, and R. Agrawal, "Downlink scheduling and resource allocation for ofdm systems," in Proceedings of the Conference on Information Sciences and Systems (CISS), 2006.