Admission Control and Interference-Aware Scheduling in Multi-hop WiMAX Networks

Debalina Ghosh, Ashima Gupta, Prasant Mohapatra
Department of Computer Science
University of California, Davis
Davis, CA 95616
Email: {debghosh, ashgupta, pmohapatra}@ucdavis.edu

***Abstract*—Multi-hop WiMAX networks based on IEEE 802.16 has the potential of easily providing high-speed wireless broadband access to areas with little or no existing wired infrastructure. WiMAX technology can be used as "last mile" broadband connections to deliver streaming audio or video to clients. Thus, Quality of Service (QoS) is very important for WiMAX networks. Providing QoS in multi-hop WiMAX networks such as WiMAX mesh or mobile multi-hop relay networks is challenging as multiple links can interfere with each other if they are scheduled at the same time. We propose efficient heuristic algorithms for scheduling flows in a centrally scheduled multi-hop WiMAX network. The proposed algorithms guarantee bandwidth and delay constraints of flows and allow multiple non-interfering links to be scheduled at the same time. We also define a "schedule efficiency" metric for comparing different flow scheduling algorithms. The simulation results show that the "schedule flow subchannel" algorithm leads to the best schedule efficiency.**

I. Introduction

WiMAX technology is becoming increasingly popular as a number of service providers are deploying WiMAX to provide wireless broadband connectivity to customers. WiMAX networks can be single-hop or multi-hop. In the single-hop mode, a Base Station (BS) communicates directly with several subscriber stations (SSs) [1]. In the multi-hop mesh mode, the SSs can communicate with each other and communication between a BS and a SS may involve multiple hops. The newly formed IEEE 802.16j working group is focusing on mobile multi-hop relay (MMR) networks that will enable multi-hop communication in mobile WiMAX (IEEE 802.16e) networks. In a MMR network, mobile stations or subscriber stations may communicate with a relay station (RS) instead of communicating directly with the BS [2].

QoS is extremely important for WiMAX networks as one of the potential applications is streaming audio and video. The IEEE 802.16 standard specifies different service classes which have QoS parameters like minimum rate, maximum rate, latency and jitter. The different service classes are shown in Table I. An important aspect of QoS is scheduling links for flows in a non-interfering manner to avoid packet loss. Scheduling needs to take the latency, bandwidth, jitter requirements and service classes of the different flows into account. However, the scheduling mechanism is not specified in the standard and hence it is an active research area.

In this paper, we explore the problem of admission control and scheduling of flows in multi-hop WiMAX networks. In particular, we make the following novel contributions:

1) We propose an efficient admission control and flow scheduling algorithm that schedules flows in such a manner that the bandwidth and delay requirements of the flows are satisfied. Our algorithm schedules non-interfering links concurrently leading to efficient spatial reuse. It schedules flows close to the deadline, so that new flows with earlier deadline can be admitted if there is enough capacity. We calculate the latest start time slot by which a link needs to be scheduled so that the flow meets its latency requirements for all links in the route of a flow. The algorithm consists of two phases: in the first phase, we allocate interference-free slots (timeslot-subchannel) to a flow based on the maximum bandwidth requirement of the flow. We execute the second phase only if the first phase fails to allocate enough slots to satisfy the minimum bandwidth requirement of the flow. In the second phase, we take away extra slots (slots allocated beyond the minimum bandwidth requirement to an existing flow) and allocate these slots to the flow under consideration. We admit a flow only if the minimum bandwidth requirement and latency requirement is satisfied.
2) We define a "schedule efficiency" (SE) metric to compare different flow scheduling algorithms. The schedule efficiency takes into account the service class priority of flows, the minimum bandwidth requested by a flow and the number of flows.
3) Finally, we provide extensive simulation results

TABLE I
SERVICE CLASSES IN WIMAX

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

which compares our algorithm with several variants. We compare the different algorithms using the SE metric, the total bandwidth allocated, fraction of accepted flows and computation time.

The results clearly indicate the need for admission control without which the actual number of flows that achieve the guaranteed rate is quite low. Among the five scheduling algorithms, the schedule flow subchannel (SFS) algorithm performs best in terms of schedule efficiency and computation time. All the algorithms compared schedule flows according to service class priority and are all interference aware. But the SFS algorithm performs better than the naive algorithm when there are flows with latency requirements as the SFS algorithm attempts to schedule flows as close to the deadline as possible.

The rest of the paper is organized as follows. We define the network model and state our assumptions in Section II. In Section III we discuss our algorithms for flow scheduling and admission control. We present the simulation results in Section IV. In Section V we describe related work. Section VI concludes the paper.

II. NETWORK MODEL

A typical network model is shown in Figure 1. It consists of one base station and several subscriber stations or relay stations that are connected to the base station using one or more hops. Usually the number of hops is small. All flows are centrally scheduled by the base station (BS).

A. Assumptions

Our assumptions are listed below:

- The topology of the network is a tree with the BS at the root of the tree. This is a standard topology used for most WiMAX networks [3].

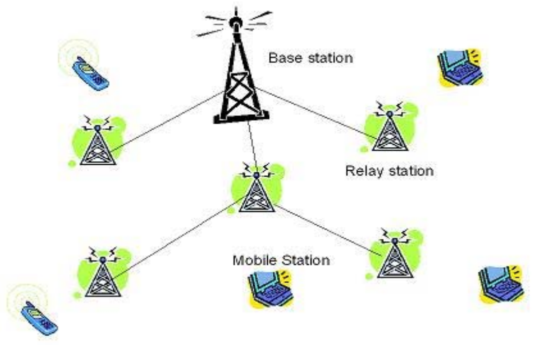

Fig. 1. Network Model

- Routing from a source to a destination is fixed and non-adaptive.
- The flows are non-splittable, that is, they follow only one path.
- All nodes are half-duplex.
- All links use the same coding scheme across all frames. However, our algorithm can be easily extended to different coding schemes.
- The set of interfering links is provided as input by the user. Hence this algorithm is independent of any specific interference estimation method.

B. Physical Layer

We assume the physical layer is Orthogonal frequency division multiple access (OFDMA). Both IEEE 802.16 and IEEE 802.16e Physical layer supports OFDMA. OFDMA divides a channel into a number of subcarriers. The subcarriers can be grouped together into "subchannels". Multiple access is two-dimensional in OFDMA. Thus a particular subscriber can be assigned one or more subchannels. A slot is the basic unit of assignment in the time-frequency grid. Thus OFDMA is a combination of both frequency domain and time domain multiple access. Therefore it results in improved capacity, better scheduling and QoS support and reduced interference. However, as the number of slots in the two dimensional time-frequency grid is more compared to a single dimensional time or frequency domain, the scheduling overhead will be higher.

C. Even-odd labeling

We assume all nodes in our wireless network are half-duplex and they are not capable of transmitting and receiving at the same time. Thus, we adopt the even-odd framework as described in [3]. A link in the network is assigned either an even or odd label and alternate links are assigned different labels. An even link transmits in an even timeslot and an odd link transmits in an odd timeslot.

III. Problem Formulation and Algorithm

A. Statement of the Problem

The problem that we are trying to address can be stated as:

Given a number of flows with minimum rate, maximum rate and latency requirements, how can we admit and schedule the flows so that the minimum rate and latency requirements of the flow can be guaranteed while ensuring maximum bandwidth utilization under a specified interference model?.

This also ensures that *In case of resource availability, the flows are able to transmit at the maximum requested rate.*

We define 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 other parameters like the start time slot. Given a number of flows that need to be scheduled and their routes, we compute the start time slots of the virtual link in the following manner. The start time slot of a link is the timeslot within a frame by which the link must be scheduled so that the flow reaches the destination by its deadline. Assuming each link has the same rate, the start time slot of a link can be defined as:

$$ts = \frac{d - fs - kt_1}{t_2}$$

where ts is the start time slot of a link, d is the deadline, fs is the start 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 corresponding link is even or odd.

The above expression can be used to compute the start time slots of links corresponding to UGS, ertPS and rtPS flows as those flows have latency requirements. For BE and nrtPS flows that do not have latency requirements, the end time of the UL (or DL) frame is considered as the start time slot of those flows.

The problem can be formulated as a list multi-coloring problem where each virtual link is a vertex in a graph and is connected to all interfering virtual links. We know that the list multi-coloring problem is NP-complete even for binary trees [4] and thus we propose a heuristic algorithm to solve the problem.

B. Heuristic Algorithm

The algorithm is used to schedule flows in each scheduling period. A scheduling period consists of an integral number of frames. The same schedule is followed in all the frames of a scheduling period. We calculate the minimum number and maximum number of slots required by a flow in a frame in the manner illustrated in [5]. The flows are scheduled according to the service class priority.

The scheduling algorithm is shown in Algorithm 1. The algorithm consists of two phases. First, we attempt to schedule free timeslots and subchannels to the flow and find out the maximum bandwidth that can be allocated to all the links of the flow (bottleneckBw). If any link has been allocated more bandwidth than the minimum bandwidth requirement of a flow, we tag these extra timeslot-subchannels so that these can be preempted by other flows if required. If the bandwidth assigned to a link is less than the minimum bandwidth requirement, we execute the second phase (ScheduleFlowExtra) which requires discovering these tagged slots and allocating them to the flow being considered. The flows are scheduled in an interference-aware manner. Primary interference at a node v occurs when two nodes transmit to it in the same slot. Secondary interference at v occurs because of two simultaneous transmissions to different receivers. Our assumed interference model accounts for both. Thus, two interfering links are not scheduled in the same timeslot-subchannel. Also, even/odd links are scheduled in even/odd timeslots respectively.

```
for each flow f in F do
    bwAlloc = SFS() or Timeslot-1() or Timeslot-2();
    if bwAlloc $<$ min bandwidth requirement of f then
        Scheduled = ScheduleFlowExtra();
    end
    if bwAlloc $>$ min bw or Scheduled is TRUE then
        Tag Extra slots;
        Make Temporary Schedule Permanent;
        Update bandwidth allocated to flow;
        accept flow f;
    else
        reject flow f;
    end
end
```

Algorithm 1: Scheduling algorithm

Timeslots-subchannels can be allocated in the first phase in three different ways. In the first method (Algorithm 2), we determine the interference-free subchannels in a timeslot that have not been allocated (FindFreeSubchannels). If the number of free subchannels does not fulfill the maximum bandwidth requirements of the flow, then the previous timeslot is considered. If the maximum bandwidth requirements of the flow is met or we reach the start of the frame, then we schedule the next link in the route of the flow. Once all the links of a flow have been considered, the second phase of Algorithm 1 is executed if any link in the route does not meet the minimum bandwidth requirement. Algorithms 3 and 4 show the two other methods by which slots are allocated to a flow. In both the algorithms, only one subchannel in a timeslot is allocated to a flow and then the previous timeslot is considered. In the timeslot-1

```
bottleneckBw = maximum bw required by f;
for each link l in f do
    Ts = start time slot for link l;
    slotsalloc = 0;
    while Ts > startOfFrame do
        numfreeslots = FindFreeSubchannels(Ts,l);
        if (slotsalloc + numfreeslots) < maxslots then
            Add numfreeslots to temporary schedule;
            slotsalloc = slotsalloc + numfreeslots;
            Ts = Ts - 2;
        else
            Add (maxslots - slotsalloc) to temporary schedule;
            slotsalloc = maxslots;
            break;
        end
    end
    update bottleneckBw;
end
return bottleneckBw;
```

Algorithm 2: ScheduleFlowSubchannel (SFS)

algorithm, we try to allocate the first free subchannel in a timeslot whereas in the timeslot-2 algorithm we consider a particular subchannel k in all timeslots and allocate it if it is interference free.

```
bottleneckBw = maximum bw required by f;
for each link l in f do
    initialize slotsalloc and subchannelCtr to 0;
    BWSatisfied = FALSE;
    find subchannels that are free from interference;
    while (subchannelCtr< MAXSUBCHANNELS)
    and (BWSatified is FALSE) do
        Ts = start time slot of link l;
        while (Ts > startOfFrame)
        and (slotsalloc < maxslots) do
            Add the first free subchannel in timeslot Ts to
            tempschedule;
            update slotsalloc;
            Ts = Ts - 2;
        end
        if slotsalloc < maxslots then
            subchannelCtr++;
        else
            BWSatisfied = TRUE;
            break;
        end
    end
    update bottleneckBw;
end
return bottleneckBw;
```

Algorithm 3: Timeslot-1

Given the space requirements of the paper we do not present here the details of the ScheduleFlowExtra algorithm that implements the preemption of the extra tagged slots.

C. Naive algorithm

We compare our heuristic with a naive algorithm (Algorithm 5). The naive algorithm is also interference-aware, i.e., interfering links are scheduled in different slots. But the naive algorithm schedules flows from the start of the frame and only allocates the minimum number of slots required. The naive algorithm consists of only one phase and involves no tagging of extra slots. The naive algorithm attempts to allocate a free subchannel in a particular timeslot. If none of the subchannels are available, it proceeds to the next timeslot.

```
bottleneckBw = maximum bw required by f;
for each link l in f do
    initialize slotsalloc and subchannelCtr to 0;
    BWSatisfied = FALSE;
    while (subchannelCtr< MAXSUBCHANNELS)
    and (BWSatified is FALSE) do
        Ts = start time slot of link l;
        while (Ts > startOfFrame) and (slotsalloc < maxslots) do
            if subchannel[subchannelCtr] is free in timeslot Ts then
                Add subchannel to tempschedule;
                update slotsalloc;
            end
            Ts = Ts - 2;
        end
        subchannelCtr++;
        if slotsalloc >= maxslots then
            BWSatisfied = TRUE;
        end
    end
    update bottleneckBw;
end
return bottleneckBw;
```

Algorithm 4: Timeslot-2

```
for all links l in f do
    slotsalloc = 0;
    finalslot = starttimeslot of link l;
    Ts = startOfFrame;
    subchannelCtr = 0;
    while (Ts <= finalslot) and (slotsalloc < minslots) do
        if subchannel[subchannelCtr] is free then
            slotsalloc++;
            update tempschedule;
        end
        subchannelCtr++;
        if subchannelCtr = MAXSUBCHANNELS then
            subchannelCtr = 0;
            Ts = Ts+2;
        end
    end
    if slotsalloc < minslots then
        Scheduled = FALSE;
        return Scheduled
    end
end
Scheduled = TRUE;
return Scheduled
```

Algorithm 5: Naive Algorithm

D. Schedule Efficiency

We propose a schedule efficiency (SE) metric to evaluate different scheduling algorithms.

$$SE = \frac{C_a}{C_t}$$

$$C = W_{UGS} * bW_{UGS} + W_{rtPS} * bW_{rtPS} + W_{ertPS} * bW_{ertPS}$$

$$+ W_{nrtPS} * bW_{nrtPS} + W_{BE} * bW_{BE}$$

where the subscript a and t in C_a and C_t refer to admitted and total flows respectively, W_{sub} is the weight of flows of class *sub*, bW_{sub} is the total bandwidth (minrate) allocated to *sub* flows and the subscript *sub* refers to the service class type (UGS, ertPS, rtPS, nrtPS, BE) of the flows.

The SE metric measures how efficiently an algorithm schedules flows. An algorithm that schedules higher

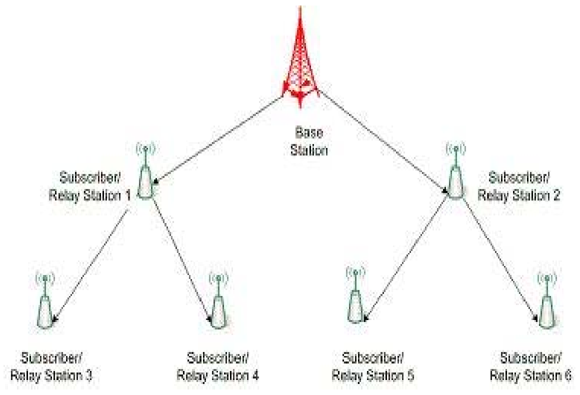
Base Station
Subscriber/ Relay Station 1
Subscriber/ Relay Station 2
Subscriber/ Relay Station 3
Subscriber/ Relay Station 4
Subscriber/ Relay Station 5
Subscriber/ Relay Station 6

Fig. 2. Topology

priority flows is better than one that schedules lower priority flows even if the number of scheduled flows is same. Thus the weight of an accepted flow and its minimum rate in the SE metric takes both the priority and the bandwidth of a flow into account. Hence a scheduling algorithm that has a higher SE value is more efficient than one that generates a lower SE value even if the number of flows accepted by the second algorithm is higher.

IV. Simulation Results

We use a custom simulator written in C. The simulation runs in two threads - the flow generator that generates flows and the scheduler that checks at every configurable scheduling period and schedules these flows. We can perform both dynamic and static flow scheduling depending on whether we run the threads concurrently or not. For purposes of comparison with the other scheduling algorithms we used static scheduling so that the input to all the algorithms is identical. The flows are generated according to Poisson arrival process. For our results we limited the number of flows so that the sum total of their minimum bandwidth requirements matches the maximum capacity of the network. The number of flows generated could be different each time as the type of flows are also generated randomly and each class is associated with a different latency and bandwidth range. We use a uniform random generator to generate the source and destination of each flow. We use a fixed topology that is provided by user input as is the interference model. For the following simulation scenarios we used the topology shown in Figure 2. The frame size and maximum number of subchannels are also configurable and we used 5ms and 256 respectively.

We maintain five global queues at the BS ordered by priority (UGS highest and BE lowest). Within each queue flows are ordered on arrival times. Each flow has an associated route that is computed from the source and destination generated by the flow generator and the topology input by the user. We perform link scheduling for spatial efficiency. The schedule for each link is computed by the algorithm and stored as a two dimensional structure of subchannels indexed by timeslots.

We use five algorithms for scheduling. The SFS, Timeslot-1, Timeslot-2 and Naive algorithms are described in detail in Section III. We also present results from the MinSubchannel algorithm which is similar to the SFS algorithm except that it allocates only for the minimum bandwidth requirements of the flows. In all the algorithms flows are picked for scheduling in service class priority and within each service class they are picked on first arrival time.

We generated different sets of flows for which all of the above algorithms were executed. The flow mix in each set was varied from 10% of the network capacity for one service class to 100% of that class and the rest divided equally among all the other service classes. We compare the algorithms on a number of parameters as described further. The first parameter *fraction of accepted flows* also illustrates the effect of admission control.

- *Fraction of Accepted flows*: This is the measure of total number of accepted flows per the total number of flows. However, it does not take bandwidth into account. Hence an algorithm could accept many BE flows but still be bandwidth inefficient compared to another that would accept fewer BE flows but better number of nrtPS flows. In our simulations we assume that BE flows have a minimum requirement of one slot bandwidth. We illustrate the effect of admission control on the fraction of accepted flows in the Naive algorithm.The NaiveWithoutAc is the naive algorithm with no admission control. This algorithm schedules all flows in order of priority without checking whether the deadline will be met. Although a number of flows get accepted when there is no admission control however the actual number of flows that meet the minimum requirements are low. Figures 3- 6 show that admission control increases the fraction of accepted flows significantly for ugs, rtPs and ertPs flows as these flows have latency requirements. However, as the number of nrtPs flows increase (Figure 6), the difference in the fraction of accepted flows is negligible as nrtPs flows do not have latency requirements.
- *Schedule Efficiency*: “Schedule Efficiency” is a weighted measure of the total minimum required bandwidth of flows accepted per the total minimum required bandwidth of all flows that were to be scheduled. The results obtained for the various flow sets are depicted in the Figures 7- 10.

  As can be seen, Schedule Efficiency is maximum for the SFS algorithm. This indicates that it can

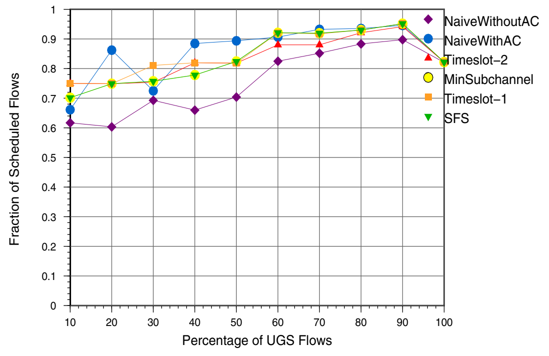

Fig. 3. Effect of Admission Control and Scheduling, varying percentage of UGS flows

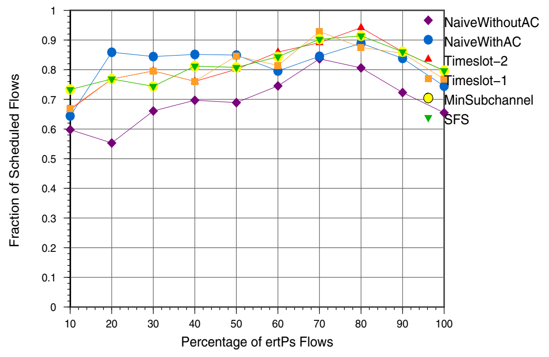

Fig. 4. Effect of Admission Control and Scheduling, varying percentage of ertPs flows

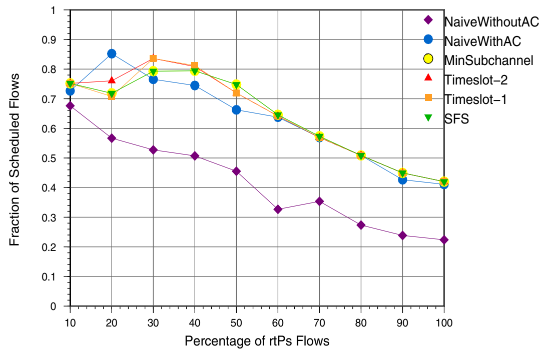

Fig. 5. Effect of Admission Control and Scheduling, varying percentage of rtPs flows

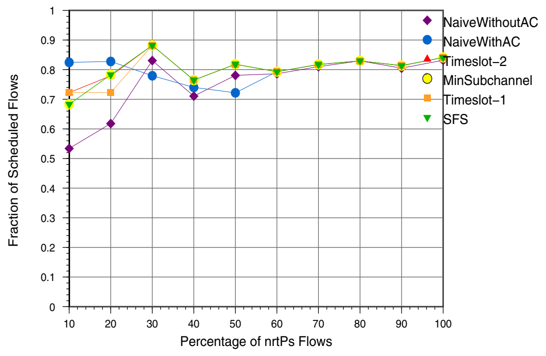

Fig. 6. Effect of Admission Control and Scheduling, varying percentage of nrtPs flows

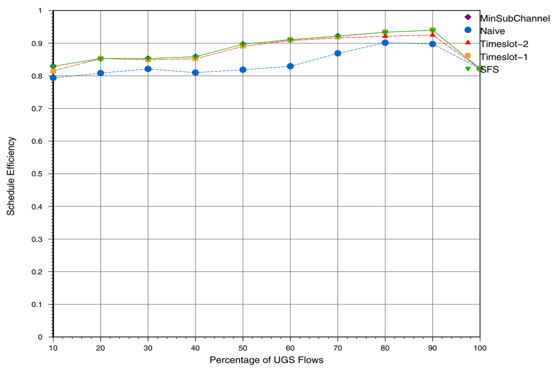

Fig. 7. Schedule efficiency with varying percentage of UGS flows

allocate bandwidth efficiently and with priority to the higher service classes. Schedule Efficiency(SE) for the MinSubchannel algorithm is also the same since it is computed over the minimum bandwidth requirements of the different kinds of flows admitted instead of the actual bandwidth allocated. The SE curve of the Naive algorithm is the least while

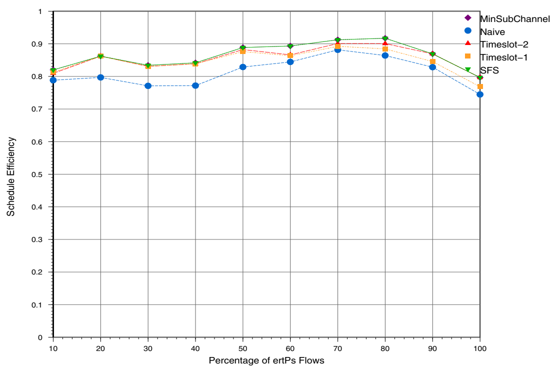

Fig. 8. Schedule efficiency with varying percentage of ertPs flows

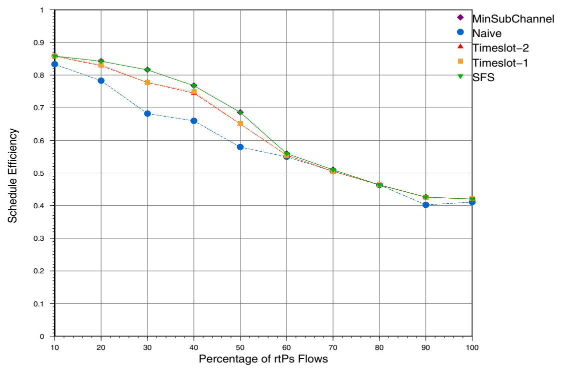

Fig. 9. Schedule efficiency with varying percentage of rtPs flows

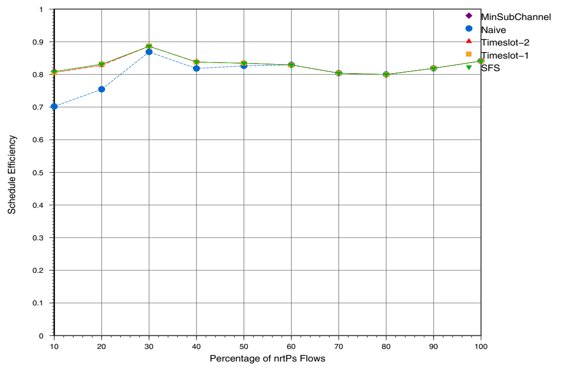

Fig. 10. Schedule efficiency with varying percentage of nrtPs flows

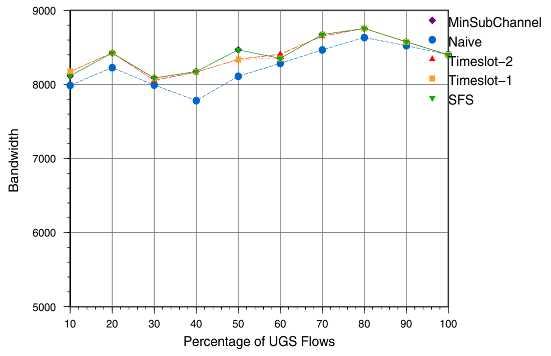

Fig. 11. Bandwidth utilization with varying percentage of UGS flows

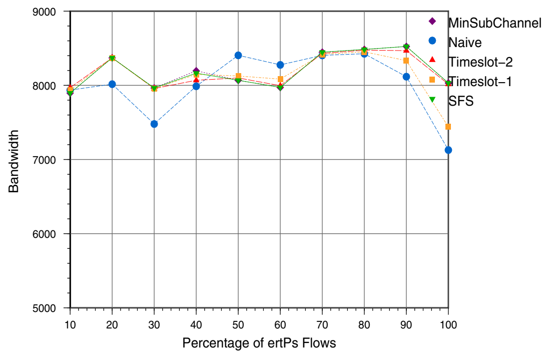

Fig. 12. Bandwidth utilization with varying percentage of ertPs flows

that of the timeslot algorithms are close to that of the SFS algorithm. The difference in performance decreases as the flow priority decreases and the latency requirements are relaxed. Hence the SE curve for larger percentage of nrtPs flows in the total flow mix is approximately the same for all the algorithms.

- *Bandwidth allocated*: This compares the total bandwidth allocated to all the accepted flows. Note that this may not be the same as the total minimum required bandwidth of all the flows since some algorithms, viz., subchannel first, timeslot-1 and timeslot-2 allocate more than the minimum required bandwidth. The results obtained for the various flow sets are depicted in the Figures 11- 14.
  The bandwidth efficiency is highest for the SFS algorithm as we vary the percentage of UGS flows in the flow mix from 10 to 100%. The Naive algorithm shows the lower performance primarily because it starts allocating slots from the beginning of the subframe. Hence it is unable to meet the latency requirements of delay sensitive flows and has to reject them. This is also the primary reason for its poor performance in Figure 13. The secondary reason is that it allocates only the minimum required bandwidth defined by the flows as does the MinSubchannel algorithm which also shows a lower performance. The performance of all the algorithms converges to approximately the same as the flow mix gets saturated with one kind of flow. This is because with just one service class in the flow set, flow priority and hence characteristics of the flow

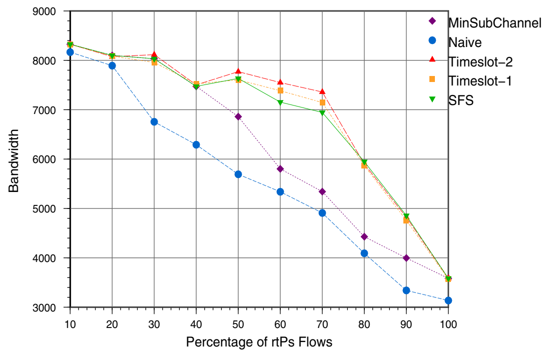

Fig. 13. Bandwidth utilization with varying percentage of rtPs flows

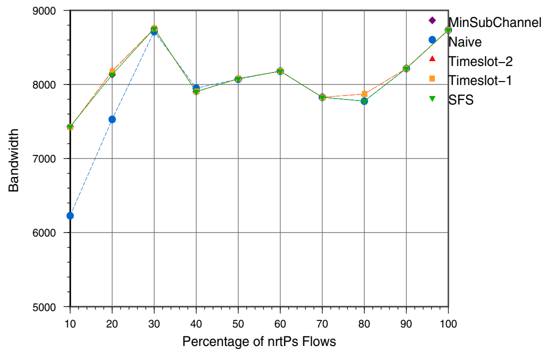

Fig. 14. Bandwidth utilization with varying percentage of nrtPs flows

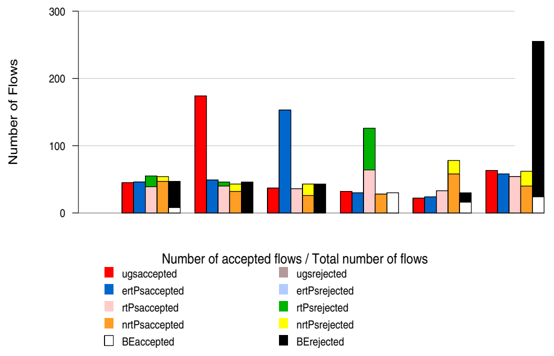

Fig. 16. SFS algorithm

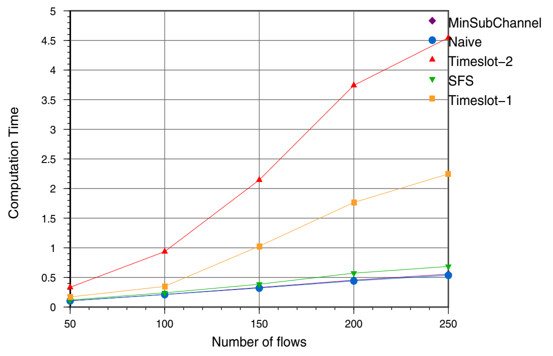

Fig. 15. Number of flows vs Computation Time

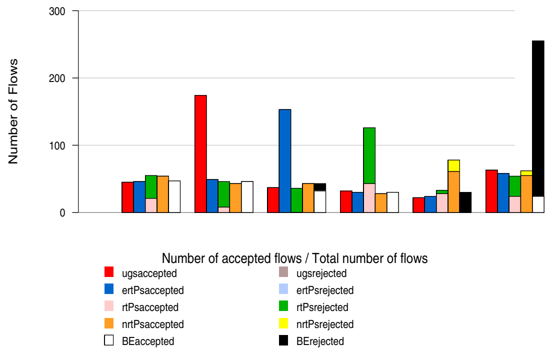

Fig. 17. Naive Algorithm

type have little effect on the algorithm performance. The SFS and timeslot algorithms do best in almost all scenarios except the two cases as shown in Figure 12. This is because the flow mix had higher BE traffic that has a lower bandwidth requirement and the Naive algorithm schedules a higher number of those.

- *Computation Time*: Computation time is the time taken by the scheduling algorithm to execute. It reflects the order of the algorithm. We generated flow sets with varying number of total flows while keeping the flow mix uniformly distributed. We then executed each of the scheduling algorithms on every flow set. The results are depicted in Figure 15. As can be seen the computation times for Naive and MinSubchannel algorithm are the lowest. These do not perform any tagging and preemption. However the computation time of the SFS algorithm also increases linearly with the number of flows and is comparable to the other two in-spite of providing the additional advantages of higher bandwidth utilization. The computation times of both the timeslot algorithms increases rapidly with the number of flows. The order of the increase is a polynomial of degree 3. The inefficiency of the timeslot algorithms is attributed to the fact that each timeslot could be revisited multiple times during the allocation since only one subchannel per timeslot is assigned at a time. The timeslot-2 algorithm is even more inefficient since it searches for the possibility that the same subchannel is free during all the timeslots instead of getting the first free one.

Figures 16 and 17 shows the total number of flows and accepted flows for each service class in a given flow for the SFS and Naive algorithms. We used five different flow sets with varying flow mixes. As can be seen the SFS is bandwidth efficient and can schedule a larger number of higher priority flows than the Naive algorithm. For example, in the first flow mix, the SFS algorithm schedules approximately 80% of the rtPS flows while the naive algorithm schedules approximately 40% of the rtPS flows. However, the naive algorihm is able to schedule all the BE flows while the SFS algorithm can schedule very few BE flows. This is also due to the fact that the minimum bandwidth requirement of a BE flow is only one slot.

V. Related Work

Some researchers have addressed the problem of centralized link scheduling in WiMAX networks. Some of the proposed scheduling algorithms ([5], [6]) schedule only one link at a time and hence they do not effectively utilize the capacity of the network. Others like [3] and [7] try to allocate non-interfering links at the same time.

An interference-aware route construction and scheduling algorithm is developed in [7]. The capacity request $D(k)$ of an SS node k is represented in terms of link demands $Y(j)$ for every link j. The scheduling algorithm iteratively determines $ActiveLink(t)$, that is, the set of active links at the time t. An interference-aware scheduling algorithm is described in [8] where a SS is assigned service token based on its traffic demand. In each timeslot a link is selected based on a certain criterion and the service token of the transmitter is decreased and the service token of the receiver is increased by one. A centralized scheduling scheme using multiple channels and single transceivers in a WiMAX Mesh Network is discussed in [9]. The goal is to minimize the length of scheduling defined as the number of timeslots needed to complete all the data transmissions. The authors in [6] compare the performance of schedulers for multiple traffic classes (multi-class Modified Largest Weighted Delay First) vs a joint scheduler that is used for all classes until the waiting time of a QoS packet in the queue exceeds 50% of its maximum allowable delay. However, none of the above schemes use OFDMA as the physical layer. The authors in [10] define a fairness model and then develop an efficient algorithm for calculating the optimal schedule under the fairness model. Perhaps the most relevant to our scheme is Narlikar, Wilfong and Zhangs' even-odd framework which we have adopted in our algorithm [3]. Each node is alternately labeled even or odd; even nodes transmit in even timeslots and odd nodes transmit in odd timeslots. They present techniques for constructing interference-free routes within the even-odd framework. They also show that if any wired scheduling policy with delay bound is implemented locally at a node, the scheduling framework guarantees approximately twice the delay of the wireline scheduler. However, the scheme requires that a route does not contain two interfering even or odd links. Thus, there may not be a feasible route in their scheme even if two nodes are able to communicate with each other.

VI. Conclusion

In this paper we studied the admission control and scheduling problem in multi-hop WiMAX networks based on IEEE 802.16. This is an NP-Hard problem and we described some heuristic approaches. Specifically, we described five different algorithms, schedule flow subchannel (SFS), timeslot-1, timeslot-2, minsubchannel and the naive algorithm. Each of the algorithms uses centralized scheduling and guarantee collision free schedules by scheduling only non-interfering links in the same slots. The SFS and timeslot algorithms also maximize bandwidth consumption by allocating extra bandwidth and performing preemption if required. We introduced the "schedule efficiency" (SE) metric and measured the five algorithms against this metric, as well as computation time, bandwidth efficiency and fraction of accepted flows. We used a custom simulator and flow generator and measured the performance of the algorithms against various flow sets. We find the SFS algorithm to be the best in terms of the SE and the computation time. Its bandwidth efficiency is also comparable to the other algorithms. While in some cases the fraction of flows accepted has been higher in the naive algorithm however this is because the naive algorithm inefficiently schedules delay sensitive rtPS flows and instead schedules a larger number of BE flows that have a smaller bandwidth requirement. The SFS algorithm is more efficient in scheduling delay sensitive flows. Hence we recommend the SFS algorithm. We are currently working on the mathematical formulation of the problem. The mathematical model will serve as a benchmark for our proposed algorithms and other such heuristics.

References

[1] C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "Ieee standard 802.16: A Technical Overview of the WirelessMAN Air Interface for Broadband Wireless Access," *IEEE Communications Magazine.*, 2002.

[2] R. B. Marks, M. Nohara, J. Puthenkulam, and M. Hart, "IEEE 802 tutorial: 802.16 mobile multihop relay," Website, 2006, "http://www.ieee802.org/16/sg/mmr/index.html".

[3] G. Narlikar, G. Wilfong, and L. Zhang, "Designing multihop wireless backhaul networks with delay guarantees," in *Proceedings of Infocom*, 2006.

[4] D. Marx, "Graph coloring problems and their applications in scheduling," *Periodica Polytechnica Ser. El. Eng.*, vol. 48, no. 1-2, pp. 5–10, 2004.

[5] A. Sayenko, O. Alanen, J. Karhula, and T. Hämäläinen, "Ensuring the QoS requirements in 802.16 scheduling," in *MSWiM*, 2006.

[6] H. Shetiya and V. Sharma, "Algorithms for Routing and Centralized Scheduling to Provide QoS in IEEE 802.16 Mesh Networks," in *WMuNeP*, 2005.

[7] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. J. Haas, "Interference-aware IEEE 802.16 Wimax mesh networks," in *Vehicular Technology Conference*, 2005.

[8] B. Han, W. Jia, and L. Lin, "Performance evaluation of scheduling in IEEE 802.16 based wireless mesh networks," *Computer Communications.*, 2007.

[9] P. Du, W. Jia, L. Huang, and W. Lu, "Centralized Scheduling and Channel Assignment in Multi-Channel Single-Transceiver WiMax Mesh Network," in *WCNC*, 2007.

[10] M. Cao, V. Raghunathan, and P. Kumar, "A Tractable Algorithm for Fair and Efficient Uplink Scheduling of Multi-hop WiMAX Mesh Networks," in *WiMesh*, 2006.