A Scheduling and Call Admission Control (CAC) Algorithm for IEEE 802.16 Wireless Mesh Networks

Jeevan Chalke and Anirudha Sahoo
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay, Powai, Mumbai, India, 400076
email: {jeevan, sahoo}@it.iitb.ac.in

Abstract

The IEEE 802.16 or WiMax standard enables deployment of broadband wireless networks in geographically large areas. It supports three modes: Point to Point (P2P), Point to Multi-Point (PMP) and Mesh. WiMax system provides QoS guarantee in terms of bandwidth, delay and jitter. So scheduling and call admission control (CAC) play an important role in a WiMax system. But the IEEE 802.16 standard does not specify any scheduling or admission control mechanism. In this paper, we present an efficient centralized scheduling scheme and a QoS aware CAC for WiMax network with a mesh topology. The scheduling scheme uses parallel transmission to increase the spatial reuse and hence increases the overall system throughput. We outline delay analysis of the system which is used by the QoS CAC to provide QoS guarantee. We also propose a slot prediction scheme for realtime polling services which can significantly reduce the overall system delay. Using simulation we show that our scheduling algorithm performs much better than that proposed in the standard in terms of acceptance ratio and system throughput. We also show that our QoS aware CAC is better suited for connections requiring QoS than conventional bandwidth based CAC which does not guarantee QoS.

1. Introduction

The IEEE 802.16d (or WiMax) for wireless metropolitan area networks (WMAN) is a standard for wireless broadband access [2]. It supports P2P, PMP and mesh topology and provides a scalable solution for last mile access. WiMax can offer long range coverage with line-of-sight transmission of upto 70 Mbps. It is suitable for many areas that are too remote to provide Internet connectivity using wireline network technology. Mesh mode of deployment increases the coverage of a network using a single base station and multi-hop communication and thereby lowers the deployment cost. WiMax system supports QoS guarantee in terms of bandwidth, delay and jitter. Hence scheduling and call admission control play an important role in such a system. These two modules interact with each other to make sure that the QoS requirement of connections are met while efficiency of the system is also high. But the scheduling and call admission control (CAC) becomes more complex in mesh mode than in PMP mode.

In this paper, we present an efficient centralized scheduling scheme which uses parallel transmission to increase the spatial reuse and hence increases the overall system throughput. We also propose a simple bandwidth based CAC that provides bandwidth guarantee to connection, but does not provide QoS guarantee to connections. We then present a QoS aware CAC (called QoS_CAC_MH) which not only ensures that a connection gets its required bandwidth at each node on its path but also meets its other QoS requirements such as delay and jitter. We present a detailed delay analysis in multi-hop mesh network, which is a very important component of the QoS aware CAC. We also propose a slot prediction scheme for realtime polling services which can significantly reduce the overall system delay. We present our simulation results which show that our scheduling algorithm performs much better than that proposed in the standard in terms of acceptance ratio and system throughput. We also show that QoS_CAC_MH should be preferred over bandwidth based CAC in order to provide QoS guarantees to the connections with very little performance hit in terms of acceptance ratio.

2. Related Work

The WiMax technology is based on the IEEE 802.16 standard [2, 1], which supports both PMP and mesh topologies. The standard specifies a centralized scheduling for mesh topology. The routing and scheduling presented in [6, 7] considers the interference graph which is used to compute blocking metric and then the schedule is determined accordingly. Authors in [9] present a routing and scheduling algorithm which reserves the required resources over the fixed path to provide QoS guarantee. In [4], authors have presented an efficient fair scheduling algorithm according to a new fairness model in which bandwidth is allocated as per the actual traffic demands such that the capacity region is not sacrificed for fairness. The CAC presented in [8] admits the best possible subsets of connection while respecting their required QoS guarantees. Then it tries to admit all the connections with certain degradation of their QoS requirements.

3. Routing Tree

In a mesh network, routing tree plays an important role in forwarding packets. Unlike PMP network, in mesh network, communication between source and destination can happen without the participation of the mesh base station (MBS). Hence, before schedule can be assigned to the subscriber stations (SS), routing tree should be built. This ensures that there is a unique path from every SS to the MBS.

3.1. Building Routing Tree

Building of the routing tree starts with the MBS at the root. Then the tree grows as and when new nodes (or SSs) join the network. A new node who wants to join the network use the following steps to determine its parent.

1. New node broadcasts NEW-ENTRY message after listening to the beacon from the MBS. The message contains MAC address of the node.
2. Each neighbor of this new node receives this message and reply with the following information about itself: hop count (from MBS) and number of children.
3. This new node waits for k consecutive frames to gather these replies from all its neighbors, where the system runs with $k-$hop parallelism (explained later).
4. Then it would run parent selection algorithm (described later in this section) to decide its parent.
5. It then sends a unicast acknowledgement to the parent selected by the algorithm.
6. The parent then forwards this information to the MBS to update its routing tree, which is used by MBS for scheduling and call admission control.

3.2. Parent Selection Criteria

A new node has to run parent selection algorithm if it gets multiple responses to NEW ENTRY message. The algorithm is very simple. If a new node gets multiple response to NEW-ENTRY message, then it chooses the node with minimum hop count as the parent. This ensures that two SSs which are at the same distance from the MBS (in terms of hop count) and are on separate branches of routing tree can share the same frame for transmission (but will be assigned different slots). Additionally, it also ensures that on the same branch, SSs which are at a distance (in terms of hop count) more than the interference range of SSs and do not cause *hidden node* problem can be assigned the same frame. But if there are more than one parent node having same hop count, then it chooses the one with lesser number of children. This helps in load balancing across the nodes. We assume that whenever a link or node goes down, routing protocol running in the mesh network will rerun the parent selection algorithm and form a new routing tree.

4. An Efficient Centralized Scheduling

Centralized scheduling scheme for mesh network proposed in the standard does not utilize spatial reuse [2]. Even if the shortest path from source to destination does not go through the MBS, it reserves required slots at all the hops in the path from source SS to MBS and from MBS to destination SS. It also does not allow parallel (or concurrent) transmissions at the nodes which are outside of interference range of each other. In this section we present a parallel transmission mechanism which not only increases the overall system throughput, but also increases the call acceptance ratio.

4.1. Parallel Transmission

While building a routing tree, we have taken minimum hop count (with respect to MBS) as the first criterion for parent selection. This enables us to have nodes which are equidistant (in terms of hop count) and belong to different branches of the routing tree to share the same frame for transmission. Additionally, nodes in the same branch which are outside of interference range of each other can be assigned the same frame (taking hidden node problem into account). If we assume transmission range and interference range to be equal, then nodes which are three hops away from each other can transmit at the same time, since their (one hop) receivers will be completely outside of interference range of the transmitter. We refer to this as $3-$hop parallelism. In general, based on the interference range, there can be $k-$hop parallelism, where $k \geq 3$. With $k-$hop parallelism, we assign the same frame to the nodes which are multiple of k hops away from the MBS, i.e., nodes which are k, $2k$, $3k$, .. hops away from the MBS, they transmit in the same frame. So every node gets an opportunity to transmit their data after every k frames. Note that as per the scheduling proposed in IEEE 802.16 standard a node gets opportunity after every n frames, where n is maximum hop count of any node in the network. This is because of all the communication is routing through MBS. But in our scheme we allow the communication to go through the nearest common parent. When a new connection request is sent to the MBS, the MBS would determine the path of the connection (based on the source and destination of the connection). It would then allocate slots at all the nodes along the path of the connection. The connection may be rejected if any of the nodes does not have required number of slots.

4.2. Example of Parallel Transmission

As explained earlier, while determining the schedule, the MBS would allocate slots to a node based on its aggregate request. By aggregate request we mean the slot requirement of the node due to its local connection plus the slot requirement due to the node carrying traffic as intermediate node of other connections.

Consider routing tree of a network is as shown in Figure 2. and the sequence of requests shown in Figure 1 made by different SSs. Each request is represented as "source-destination:no_of_slots_required". For example, request number 1, 0-4:2, represents two slots required for data transfer from SS0 (or MBS) to SS4 (node 4). As requests are admitted and scheduled by the MBS, it keeps track of aggregate slot requirements of all the nodes in the network. For a given request, slots are allocated along all the nodes in the path from source to destination. For example, for request 6, SS8 needs 3 slots for a connection to SS7. So this would require 3 slots to be allocated at SS8 and SS6

		MBS	SSID									
Sr. No	Requests	0	1	2	3	4	5	6	7	8	9	10
1	0-4 : 2	(2/0)(2/0)	(0/0)(2/0)									
2	0-1 : 1	(3/0)(3/0)										
3	0-9 : 3	(6/0)(6/0)		(0/0)(3/0)				(0/0)(3/0)				
4	0-2 : 2	(8/0)(8/0)										
5	2-0 : 1			(0/1)(3/1)								
6	8-7 : 3			(0/1)(6/1)				(0/0)(3/3)		(0/3)		
7	3-5 : 2		(0/0)(4/0)		(0/2)							
8	6-0 : 1			(0/1)(6/2)				(0/1)(3/4)				
9	5-0: 1		(0/0)(4/1)				(0/1)					
10	0-10 : 3	(11/0)(11/0)										
11	7-4 : 2	(11/0)(13/0)	(0/0)(6/1)	(0/1)(6/4)					(0/2)			
12	0-6 : 2	(13/0)(15/0)		(0/1)(8/4)								

Figure 1: Sequence of requests

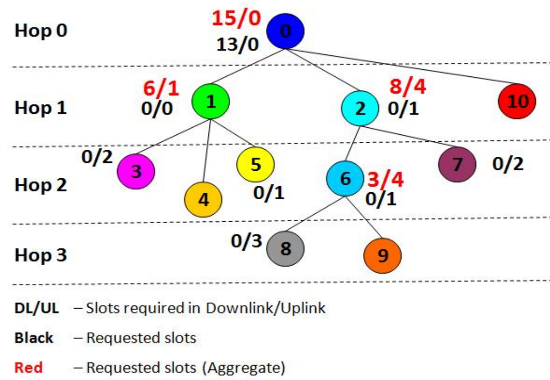

Figure 2: A scheduling tree

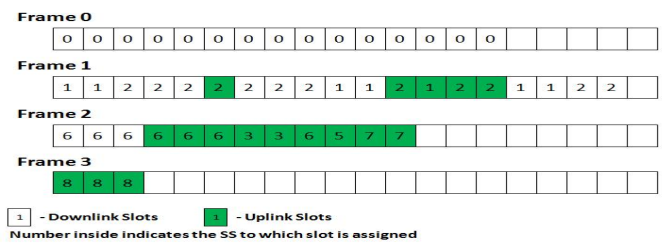

Figure 3: Slot allocation for tree shown in Figure 2

in the uplink (UL) direction and 3 slots to be allocated at SS2 in the downlink (DL) direction. Entries for each of these nodes in the path are made in the format (own)(aggregate), where own is the slot requirement for connections originating from the node itself and $aggregate$ is the total number of slots required due to its own connection and other connections which pass through this node. The slot requirement is presented in (DL/UL)2027 format. When all the requests are admitted and scheduled in the network, the aggregate slot allocation at each node is shown in Figure 2. Figure 3 shows the frames at each hop with slot allocation ordered by arrival of request. Notice that nodes with same hop count are assigned slots in the same frame, whereas nodes belonging to hop 0 and hop 3 use the same frame, i.e., frame 0 and frame 3 are transmitted concurrently.

4.3. Bandwidth based CAC (BW_CAC_MH)

Based on the above design, a very simple bandwidth based call admission control is proposed here. This algorithm makes sure that there are enough slots available at every node along the path of a connection when a connection is admitted.

Algorithm 1 presents simple bandwidth based call admission control algorithm in multi hop network. First, all UGS requests are passed to the CAC and then rtPS and nrtPS requests are processed (different request types are explained in the next section). For rtPS and nrtPS only the minimum required slots are guaranteed. In this algorithm, $Slots[i]$ represents the number of slots available at node i whereas $req.reqBW$ represents the required bandwidth for newly arriving connection (in terms of slots). src, $dest$, $commonParent$ represents the source, destination and the common parent node of the connection.

Algorithm 1 $BW\_CAC\_MH(req)$

$commonParent$ = common parent between src and $dest$ node
if Source $==$ $commonParent$ **then**
  for all nodes i along the path from src to $dest$ /*only downlink*/
  if $Slots[i] \geq req.reqBW$ **then**
    Admit this request; $Slots[i]- = req.reqBW$
  end if
else if Destination $==$ $commonParent$ **then**
  for all nodes i along the path from src to $dest$ /*only uplink*/
  if $Slots[i] \geq req.reqBW$ **then**
    Admit this request; $Slots[i]- = req.reqBW$
  end if
else
  for all nodes i along the path from src to $commonParent$ and for all nodes j along the path from $commonParent$ to $dest$
  if $Slots[i] \geq req.reqBW$ AND $Slots[j] \geq req.reqBW$ **then**
    Admit this request
    $Slots[i]- = req.reqBW$; $Slots[j]- = req.reqBW$
  end if
end if

5. Delay Analysis

In Section 4.3 we presented simple bandwidth based CAC algorithm which do not consider any QoS guarantees such as delay and jitter. But in IEEE 802.16, some class of connection (e.g., UGS, rtPS) have stringent delay and jitter requirement. So in this section we present QoS_CAC_MH algorithm which admits connections by considering its QoS parameters. Before presenting our algorithm, we will first give an overview of different service classes and their respective QoS parameters.

5.1. QoS Service Classes and Parameters

The WiMax standard defines four types of service flow classes [2, 3, 5]. (1) Unsolicited Grant Service (UGS): Mandatory QoS parameters for this service class are *Maximum Sustained Traffic Rate* ($maxRate$), *Maximum Latency* ($maxLat$), *Tolerated Jitter* (TJ), *Unsolicited Grant Interval* (UGI). (2) Real-time Polling Service (rtPS): QoS parameters for this service class are *Minimum Reserved Traffic Rate* ($minRate$), *Maximum Sustained Traffic Rate* ($maxRate$), *Maximum Latency* ($maxLat$), *Unsolicited Polling Interval* (UPI). Tolerated jitter for this service is taken to be equal to UPI. (3) Non Real-time Polling Service (nrtPS): This service class has the following QoS parameters. *Minimum Reserved Traffic Rate* ($minRate$), *Maximum Sustained Traffic Rate* ($maxRate$), *Traffic Priority*. (4) Best Effort (BE): This service class does not need any QoS guarantee, however connections can specify following parameters. *Maximum Sustained Traffic Rate* ($maxRate$), *Traffic Priority*. We assume UGS to have highest priority followed by rtPS, nrtPS and BE in that order.

5.2. Delay Computation

In this section we will compute the delay of a *data unit*, the amount of data that a node transmits in a frame for a given connection, when it has to traverse the WiMax mesh network.

5.2.1. Delay in Downlink

We consider a data unit originating from MBS and destined to a node in the downlink direction. We have designed scheduling of nodes in such a way that the downlink communication has much less delay than uplink communication. As an example, refer to Figure 4. The numbers in the nodes signify their hop count. In the downlink direction Frame 0 is assigned to hop 0 (MBS), Frame 1 is assigned to hop 1 nodes. Assuming a $3-$hop parallelism, Frame 0 can also be used by nodes in hop 3, 6, .. to transmit at the same time as hop 0 nodes. Similarly hop 1, 4, 7, .. nodes can use Frame 1 to transmit. So it is clear that hop 1 node gets the immediate next frame of hop 0 node, hop 2 node gets the immediate next frame of hop 1 node and so on. Thus, once a data unit is transmitted from the MBS it will be sequentially transmitted in the subsequent frames until it reaches its destination. If H is the number of hops from the MBS to a node then the delay (in terms of frames) of a data unit to traverse from MBS to that node in the downlink direction is given by

$$D_d = H \tag{1}$$

5.2.2. Delay in Uplink

We consider a data unit originating from a node and destined to the MBS in the uplink direction. In WiMax mesh network the frames do not have a separate uplink and downlink partition. Thus, whenever a node gets its chance (gets a frame), it would use that frame (but different time slots) to communicate with its uplink as well as downlink neighbor. Going back to our example in Figure 4, then nodes in hop 7 gets Frame 1, hop 6 gets Frame 0 and so on. Thus, after the source node transmits a data unit, the data unit has to wait for $(k-1)$ frames at every intermediate nodes until it reaches the MBS. At the source node, in the worst case, it has to wait for k frames before the node gets a chance to transmit the data unit. Hence, the worst case delay of a data unit originating from a node to reach the MBS is given by

$$D_u = (k-1) \times (H-1) + k \tag{2}$$

Figure 5 shows the frame assignment among the nodes assuming a $3-$hop parallelism. The numbers inside the frame indicate the hop count of nodes to which that particular frame is assigned. The assignment pattern repeats after three frames because of $3-$hop parallelism.

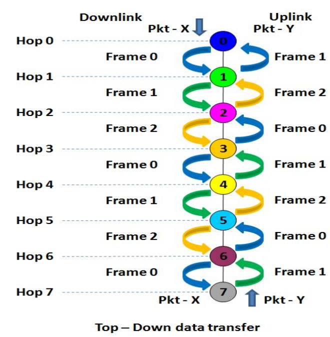

Figure 4: Uplink and Downlink Communication with $3-$hop parallelism

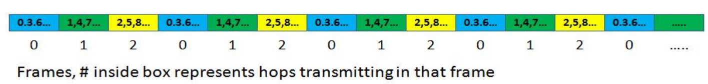

Figure 5: Frame assignment for $3-$hop parallelism

5.2.3. End to End Delay

In the previous section we calculated delay between MBS and a node and vice-versa. But in general, source and destination of a data unit can be any node. So in this section we present the delay computation for any arbitrary source and destination node. Let H_i and H_j be the hop count of source i and destination j nodes from the MBS respectively.

- when both the source and the destination are on the same branch of the routing tree, but the destination is downstream, then the delay between the source i and the destination j is given by

$$D_d^{ij} = (H_j - H_i) \tag{3}$$

  Note that if source is MBS, then (3) reduces to (1).

- when both the source and the destination are on the same branch of the routing tree and the destination is upstream, then the delay between the source i and the destination j is given by

$$D_u^{ij} = (k-1) \times (H_i - H_j - 1) + k \tag{4}$$

  Note that if destination is MBS, then (4) reduces to (2).

- Otherwise, if the source and destination nodes are on different branches of the routing tree, then we need to identify the common parent node p which is H_p hops away from the MBS. So the delay of a data unit from the source i to destination j will be the sum of the delay (in upstream direction) from source i to the common parent node p and the delay (in downstream direction) from p to the destination j. So we can use (3) and (4) with appropriate changes to get the end-to-end delay as

$$D^{ij}_{end\_to\_end} = (k-1)\times(H_i-H_p-1)+k+(H_j-H_p) \quad (5)$$

5.3. Delay Minimization for Real-time Flows

UGS connections have fixed slot requirements throughout its life time. However, in case of rtPS and nrtPS connections the slot requirement varies between $minRate$ and $maxRate$. Thus, this type of connections can change their bandwidth requirements during connection's lifetime. We denote this change by $exBW$ (this value could be positive or negative). But this request for extra bandwidth has to traverse multihop to reach the MBS where a decision will be made and a response will be sent to the SS. This process can delay and result in significant queue build up at the SS. We assume that the request for exBW is generated as soon as the SS recognizes a significant change in arrival rate. We also assume that MBS processes all its requests and generates a schedule map only after a predefined scheduling period denoted as $SchPeriod$ which is an integer multiple of frame duration (T_f). We present some schemes to address the queue build up.

5.4. Grant $minRate$

In this scheme, a connection is only given the $minRate$ throughout its lifetime, no extra bandwidth is given to the connection later on. The advantage of this scheme is that the system can admit more connection, but the average queue length of connection can be high when the bandwidth requirement of the connection is more than $minRate$.

5.5. Grant $maxRate$

This scheme is similar to *Grant* $minRate$ scheme, but connections are granted $maxRate$. Obviously this scheme overallocates bandwidth and hence would admit lesser number of connection compared to *Grant* $minRate$ scheme, but the average queue length of the connection will be much less.

5.6. Grant Normal

In this scheme, connection is admitted with $minRate$. But later on, when it asks for extra bandwidth, it is granted (if available), but the amount of extra bandwidth is exactly equal to what is needed by the SS. Thus, in this case the queue of the connection can potentially increase because of the delay in the response of the BW request. But it may later come down when the traffic arrival slows down.

5.7. Slot Prediction Scheme

In this scheme, when SS asks for extra bandwidth, it asks for more than what is actually needed, based on prediction of delay. As explained in Section 5.2, the number of frames required for the request to reach the MBS is D_u given by (2). The response from the MBS would require D_d frames (given by (1)) to reach the SS. If the request arrives in the middle of the scheduling period, then the request has to wait for the residual scheduling period before a new schedule is prepared by MBS with the new bandwidth requirement. We denote this as the $waitTime$ which is equal to ($SchPeriod - T^{arrival}_{req}$), where $T^{arrival}_{req}$ is the frame number in which the request arrives at the MBS. So the predicted delay in getting the response from the MBS is given by

$$predictedDelay = D_u + D_d + waitTime \quad (6)$$

Now, because of this delay, queue builds up at the SS. Also, connections get data units in every UPI interval. Hence, the predicted queue length (in terms of slots) is given by

$$predictedQLen = exBW \times \lceil (predictedDelay \times T_f/UPI) \rceil \quad (7)$$

SS now predicts the number of additional slots required based on this predicted queue length, so that the queue will be flushed out in one scheduling period. Thus the predicted queue length is thought of as additional slot requirement (over and above the exBW) which should now be added to the exBW. But exBW is the extra bandwidth in every UPI. Thus, the predicted queue length should be distributed along the number of UPI present in a scheduling period. Hence predicted extra bandwidth needed is given by

$$predictedBW = \lceil \frac{predictedQLen}{\lfloor (SchPeriod * T_f)/(UPI) \rfloor} \rceil \quad (8)$$

Now the total bandwidth to be requested ($reqBW$) is given by

$$reqBW = resBW + exBW + predictedBW \quad (9)$$

Where $resBW$ is the bandwidth currently reserved for the connection. Note that we are calculating the predicted BW to account for the queue build-up in the remainder of scheduling period. Once the SS is granted the extra bandwidth, the queue will be flushed out in the next scheduling period and the SS will reduce its bandwidth request to just $exBW$ so that it will not leave any reserved slots unused. It should also be noted that the slot prediction algorithm is run only if the SS asks for more bandwidth, i.e., when $exBW$ is positive.

6. QoS Call Admission Control for Multi Hop Network

So far, we have been discussing bandwidth requirement of connections. But UGS and rtPS connections also have delay and jitter requirement. Hence, the call admission control module at the MBS not only has to consider the bandwidth requirement but also have to make sure that the delay and jitter bounds are met. We refer to this call admission control as QoS CAC for Multi Hop Network (QoS_CAC_MH).

6.1. QoS_CAC_MH Algorithm

As discussed earlier, UGS and rtPS data arrives at periodic intervals of UGI and UPI respectively. We use hyper interval of these parameters to make sure that every connection fulfills its QoS requirement. We define HyperInterval of connections at a node h as follows. for UGS connections

$$HI^{UGS}_h(N^{UGS}) = \forall i\ LCM(UGI_i), 1 \leq i \leq N^{UGS}_h \quad (10)$$

where, UGI_i is the Unsolicited Grant Interval for i^{th} connection. Similarly for rtPS connections

$$HI^{rtPS}_h(N^{rtPS}) = \forall i\ LCM(UPI_i), 1 \leq i \leq N^{rtPS}_h \quad (11)$$

where, UPI_i is the Unsolicited Polling Interval for i^{th} connection. Finally, the HyperInterval for all connections are calculated as

$$HI_h(N_h) = LCM(HI^{UGS}_h, HI^{rtPS}_h) \quad (12)$$

where, $N_h = N^{UGS}_h + N^{rtPS}_h$

Algorithm 2 shows the pseudo code for QoS_CAC_MH Algorithm which checks that the end-to-end delay between source and destination is within the delay requirement of the new request. It then calls $Check\_QoS(req, Start\_Hop, End\_Hop)$ which checks whether the number of slots required at each hop are available. It also ensures that there is no deadline miss and the jitter of the connection is satisfied in every nominal interval in an hyper interval at every hop. In Algorithm 3, $req.nomInterval$ refers to UGI for UGS connections and UPI for rtPS connections. The helper routine $Search(No\_of\_Slots, First\_Slot, Last\_Slot)$ searches for No_of_Slots in the interval between $[First\_Slot, Last\_Slot]$. If the slots are found then the connection is admitted and slots are allocated to the request. The allocation of slots start from the $Last\_Slot$ towards its left. Thus it allocates slots from connection's latest deadline so that a future connection having smaller jitter value can be admitted without altering slot allocation of already admitted connections.

To understand the slot allocation, consider connection requests coming with parameters shown in Table 1. The table shows the parameters in milliseconds and in frames, assuming frame duration of 10ms. We assume that each connection needs one slot only. The slot allocation after all the three requests are admitted is shown in Figure 6. For each connection one slot is allocated starting from its tolerated jitter and to the left where one slot is available. This process is repeated in every nominal interval (of every connection) in the hyper interval.

Algorithm 2 $QoS\_CAC\_MH(req)$

Require: /* This algorithm takes new connection request, req as input and allocates slots if QoS guarantees are satisfied*/

$i = req.src$; $j = req.dest$; $p = FindCommonParent(i, j)$;
if $req.maxLat \geq D^{ij}_{end\_to\_end}$ **then**
  $Check\_QoS(req, H_i, H_p)$
  $Check\_QoS(req, H_p, H_j)$
  accept the request and allocate slots
end if

Algorithm 3 $Check\_QoS(req, Start\_Hop, End\_Hop)$

$No\_of\_Slots = req.reqBW$
for $h = Start\_Hop \quad to \quad End\_Hop$ **do**
  if $No\_of\_Slots > Slots[h]$ **then**
    reject the request and return
  end if
  // Check for any delay deadline miss
  $HI_h(N_h) = LCM(req_i.nomInterval, req.nomInterval)$ $\forall i,\ 1 \leq i \leq N_h$
  $No\_of\_nomInterval = HI_h(N_h)/req.nomInterval$
  $First\_Slot = 0$; $Found = 1$
  if request type == UGS **then**
    $TJ\_in\_Slots = req.jitter \times Slots\_per\_Frame$
  else
    $TJ\_in\_Slots = req.nomInterval \times Slots\_per\_Frame$
  end if
  $Last\_Slot = TJ\_in\_Slots$
  for $j = 1 \quad to \quad No\_of\_nomInterval$ **do**
    $Found = Search(No\_of\_Slots, First\_Slot, Last\_Slot)$
    if $!Found$ **then**
      reject the request and return
    end if
    $First\_Slot = j \times req.nomInterval \times Slots\_per\_Frame$
    $Last\_Slot = First\_Slot + TJ\_in\_Slots$
  end for
end for

Request	*nomInterval*		*Tolerated Jitter (TJ)*	
	ms	*Frames*	*ms*	*Frames*
1	60	6	30	3
2	90	9	60	6
3	30	3	30	3

Table 1: Connection requests

7. Simulation Results

In this section, we present our simulation experiment. We have written our own simulator using Java sdk 1.5 and eclipse IDE. We used a random topology of twenty five SSs and a MBS. We have used the parent selection procedure explained in 3.1 to build the routing tree shown in Figure 7.

7.1. Comparison of Bandwidth Based CAC (BW_CAC_MH)

We presents performance comparison of bandwidth based CAC using our 3−hop parallelism versus IEEE 802.16 standard. In this experiment, we considered only UGS connections.

In bandwidth based CAC, we assumed that there are data units arriving every frame. We have assumed that we have 200 slots per frame. We used Poisson distribution for generating connection requests (arrival rate).

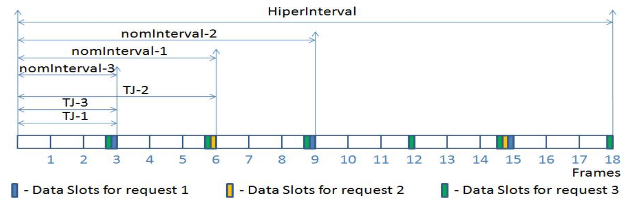

Figure 6: Slot allocation using HyperInterval

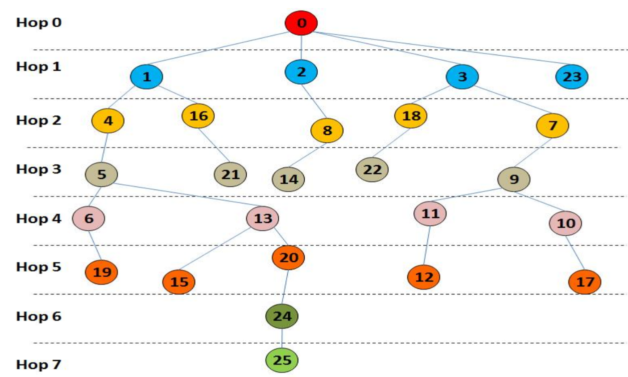

Figure 7: Routing tree generated and used in scheduling and CAC

Figure 8 shows the acceptance ratio versus request arrival rate for our BW_CAC_MH and IEEE 802.16 standard. We define acceptance ratio as the ratio of connection admitted to total number of connections. It is clear that our scheme achieves higher acceptance ratio than the standard based approach.

Figure 9 shows the corresponding throughput of the two schemes. With 3−hop parallelism approach we are able to achieve throughput of about $21Mbps$, whereas without parallelism the throughput is around $8Mbps$. Maximum throughput for the scheme given in IEEE 802.16 standard with our 3−hop parallelism is around $9.5Mbps$. That means our proposed scheme with 3−hop parallelism is more than twice better than the one proposed in standard.

7.2. QoS Aware CAC

In this section we present simulation results of our QoS based CAC. First we consider rtPS flows and compare the performance of different schemes described in Section 5.3. Here we used $minRate = 10\ slots$ and $maxRate = 20\ slots$ for every connection.

Figure 10 shows the system queueing delay in all four schemes as the request arrival rate increases. As expected, *Grant maxRate* has the least delay, since it allocates maximum rate asked by the connection, whereas *Grant minRate* has the maximum delay, since it allocates only minimum rate asked by the connection. *Slot prediction* method has delay in between the above two methods and compared to *Grant Normal* method it incurs less delay.

Figure 11 shows the corresponding performance of different schemes in terms of acceptance ratio. It is clear that there is a tradeoff between acceptance ratio and system delay, i.e., the scheme having highest delay (*Grant minRate*) has the best acceptance ratio and vice versa. But *Slot prediction* scheme has acceptance ratio very close to that of *Grant minRate* scheme, but its system delay is much lower than that of *Grant minRate*. Thus, our proposed *Slot prediction* scheme is a good choice both in terms of system delay and acceptance ratio.

7.3. Comparison of Bandwidth Based CAC and QoS Based CAC

Since bandwidth based CAC admits connections based solely on availability of slots, but does not consider other QoS constraints such as delay and jitter of the connections, it will typically have higher acceptance ratio and slot utilization compared to QoS based CAC. Figure 12 shows the acceptance ratio versus request arrival rate for the two CACs. Here we have used only

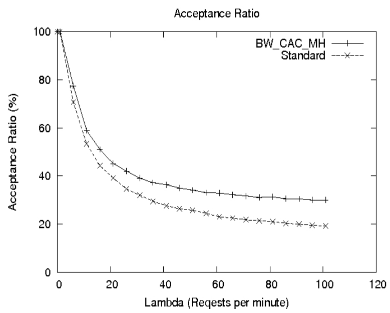

Figure 8: Acceptance ratio

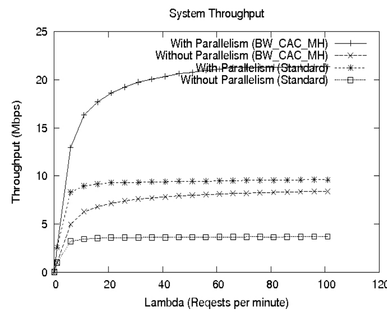

Figure 9: System throughput

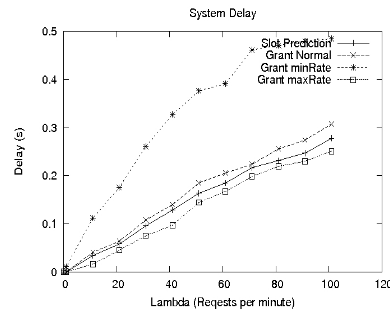

Figure 10: System delay

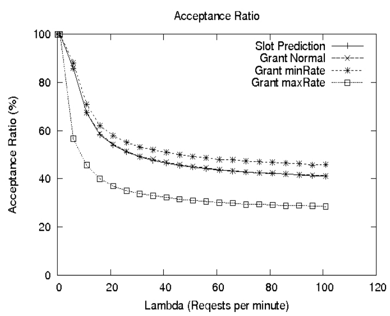

Figure 11: Acceptance ratio for various schemes (QoS based CAC)

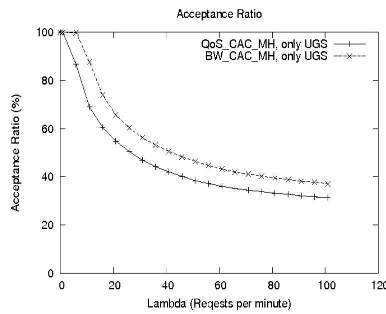

Figure 12: Acceptance ratio with bandwidth based and QoS based CAC

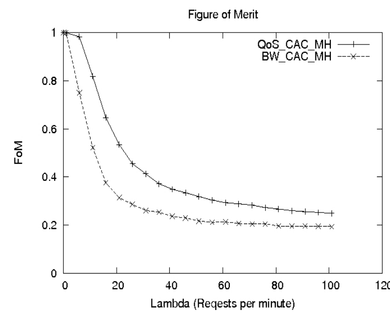

Figure 13: Figure of Merit (FoM) comparison

UGS connections requesting 10 slots each. UGI and TJ are randomly selected from the set $\{30, 60, 90\}$ ms with the constraint $TJ < UGI$. Although our QoS_CAC_MH has lower acceptance ratio it ensures that the every connections meet their delay and jitter constraints. Thus, BW_CAC_MH has a better acceptance ratio but at the cost of some connections missing their deadlines. So there is a tradeoff between acceptance ratio and number of connections missing deadline. Hence, comparing the acceptance ratio of the two CAC algorithms is not fair. So we use a composite performance index called Figure of Merit (FoM) which was also used in [5]. The FoM is defined as

$$FoM = U \times \frac{(conn\_admitted - conn\_miss\_deadlines)}{Total\_No\_of\_conns\_req} \quad (13)$$

where U is the utilization of the network, defined as the ratio of the number of slots assigned to the connections to the total number of slots in the system. $conn\_admitted$ is the number of connections admitted and $conn\_miss\_deadlines$ is the number of connections that miss their respective deadlines. It is clear from Figure 13 that our QoS_CAC_MH always performs better than BW_CAC_MH in terms of FoM.

8. Conclusion and Future Work

In this paper, we have presented an algorithm for building routing tree, an efficient centralized scheduling scheme and a QoS aware Call Admission Control for IEEE 802.16 mesh networks. The proposed centralized scheduling algorithm increases the overall system throughput by more than two times and admits $35 - 45\%$ more number of connections than the conventional scheme presented in the IEEE 802.16 standard. We have presented detailed delay analysis of the system which is used by the QoS aware CAC. This ensures that connections meet their QoS requirements such as bandwidth, delay and jitter. For rtPS connections we presented different algorithms which can be used to request for extra bandwidth. Using simulation we showed that our slot prediction algorithm performs very well in terms of system queueing delay and acceptance ratio for rtPS connections. We showed that our QoS_CAC_MH algorithm performs better than the conventional bandwidth based CAC in terms of composite performance index FoM. Thus, we can conclude that system performance in terms of throughput and acceptance ratio can be improved by using our scheduling algorithm and QoS_CAC_MH which makes sure that connections meet their respective QoS requirements. Additionally, for rtPS connections, slot prediction algorithm should be used to improve system performance in terms of delay and acceptance ratio.

We considered parallelism at the frame level. We are currently working on providing parallelism at the slot level. In this work, we have restricted our routing topology to a tree. We are working on making it a graph topology, in which case the system performance can be further improved by finding shorter paths between two nodes. We are also looking at modification required in the CAC and scheduling algorithm to take variable channel condition into account.

References

[1] Air interface for fixed broadband wireless access systems. *IEEE P802.16-REVd/D5-2004*, 2004.

[2] IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems. *IEEE Std 802.16-2004*, pages 1–857, October 2004.

[3] Wimax forum network architecture (stage 3: Detailed protocols and procedures). *NWG Stage 3 - Release 1.0.0*, March 2007.

[4] M. Cao, V. Raghunathan, and P. R. Kumar. A tractable algorithm for fair and efficient uplink scheduling of multi-hop wimax mesh networks. *In Proceedings of 2nd IEEE Workshop on Wireless Mesh Networks (WiMesh 2006)*, pages 188–193, Sept. 2006.

[5] S. Chandra and A. Sahoo. An efficient call admission control for IEEE 802.16 networks. *Local and Metropolitan Area Networks, 2007. LANMAN 2007. 15th IEEE Workshop on*, pages 188–193, June 2007.

[6] W. Hung-Yu, S. Ganguly, R. Izmailov, and Z. Haas. Interference-aware IEEE 802.16 wimax mesh networks. *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, 5:3102–3106 Vol. 5, May-1 June 2005.

[7] F. Jin, A. Arora, J. Hwang, and H. A. Choi. Routing and packet scheduling for throughput maximization in IEEE 802.16 mesh networks. *IEEE Broadnets 2007*, pages 10–14, Sept. 2007.

[8] S. Lee, G. Narlikar, M. Pal, G. Wilfong, and L. Zhang. Admission control for multihop wireless backhaul networks with qos support. *WCNC 2006. IEEE*, 1:92–97, April 2006.

[9] H. Shetiya and V. Sharma. Algorithms for routing and centralized scheduling to provide qos in IEEE 802.16 mesh networks. In *WMuNeP '05: Proceedings of the 1st ACM workshop on Wireless multimedia networking and performance modeling*, pages 140–149, New York, NY, USA, 2005. ACM.