An Efficient QoS Scheduling Architecture for IEEE 802.16 Wireless MANs

Supriya Maheshwari, Sridhar Iyer and Krishna Paul
K R School of Information Technology
Indian Institue of Technology Bombay
Email: {supriya, sri, krishna}@it.iitb.ac.in

***Abstract*— IEEE 802.16 WirelessMAN standard specifies air interface of a fixed point-to-multipoint Broadband Wireless Access. IEEE 802.16 MAC provides extensive bandwidth allocation and QoS mechanisms for various types of applications. However, details of packet scheduling mechanisms for both downlink and uplink direction are left unspecified in the standard. In this paper, we propose an efficient and scalable QoS scheduling architecture for IEEE 802.16 WirelessMANs. Our main design goals are to provide delay and bandwidth guarantees for various kinds of applications and to maintain fairness among various flows while still achieving high bandwidth utilization. We have implemented IEEE 802.16 MAC integrated with our architecture in Qualnet 3.6. We also present the simulation analysis of our architecture.**

I. Introduction

Given the large demand for high-speed internet access in homes and offices, providing the "last-mile" connectivity is an important problem. Doing so with fiber cable can be very expensive, especially in developing countries. A cheap solution to this problem is Broadband Wireless Access (BWA). BWA is capable of providing high speed network access with rapid flexible deployment at low cost.

The IEEE 802.16 standard was developed to produce high performance BWA systems. The standard specifies a WirelessMAN air interface for Metropolitan Area Networks [1]. A Wireless MAN consists of at least one radio Base Station (BS) and one or more Subscriber Stations (SS) with a fixed point-to-multipoint topology as shown in figure 1. It standardizes a common MAC protocol that works with various physical layer specifications. Section II provides an overview of IEEE 802.16 Wireless MAN standard. With a tremendous increase in data services and real-time applications over wireless networks, IEEE 802.16 has been designed to support QoS for both continuous and bursty traffic. IEEE 802.16 also specifies uplink scheduling services for heterogeneous classes of traffic with different QoS requirements. However, it does not suggest how to efficiently schedule packets from various classes to meet their diverse QoS requirements. Therefore, an efficient QoS scheduling architecture for IEEE 802.16 is required in order to provide QoS guarantees to various applications. Motivation for the problem is discussed in section III.

To the best of our knowledge, there is no literature on the detailed simulation analysis of QoS scheduling architecture integrated with IEEE 802.16 MAC. In [2], G. Chu et al have suggested a QoS architecture based on priority scheduling and dynamic bandwidth allocation. However, they do not provide sufficient details of various components of the architecture, nor do they provide simulation results. In [3], M. Hawa et al have proposed an uplink scheduling architecture to support bandwidth and delay QoS guarantees. However, they mainly address the QoS requirements of DOCSIS [4] and do not provide simulation results to show the efficiency and performance of the proposed architecture. Related work is discussed in section VII.

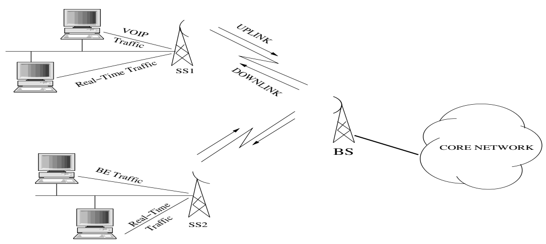

Fig. 1. Wireless Metropolitan Area Network.

In this paper, we propose an efficient QoS scheduling architecture for IEEE 802.16 Wireless MANs which provides delay and bandwidth guarantees to QoS sensitive applications and maintains fairness among various flows while still achieving high bandwidth utilization. Overview of our proposed scheduling architecture is given in section IV followed by details of the architecture in section V. We have implemented and simulated IEEE 802.16 MAC along with our proposed QoS scheduling architecture in Qualnet [5]. Our simulation results verify that our proposed architecture provides required QoS guarantees to various types of applications. Delay guarantees of real-time applications are always met. At peak load, maximum link utilization achieved is around 93%. Section VI provides details of our simulation setup and the corresponding results.

II. Overview of IEEE 802.16 Wireless MAN Standard

IEEE 802.16 WirelessMAN is a standard for fixed Broadband Wireless Access [1]. It consists of at least one BS and one or more SSs with a fixed point-to-multipoint topology. BS is a central entity and provides network connectivity to SSs through wireless link. Each SS, typically a building, consists of a number of users. A schematic wireless metropolitan area network is shown in figure 1.

IEEE 802.16 defines connection-oriented MAC supporting multiple physical layer specifications. The standard specifies one physical layer specification which operates in 10-66 GHz frequency bands. It supports channels as wide as 28 MHz and data rates upto 134 Mbps [6]. Channels are additionally separated in time via a frame. A frame is again subdivided in time via physical slots. Each frame is also divided into two logical channels, downlink channel or downlink subframe and uplink channel or uplink subframe. Downlink channel is used by BS for transmitting downlink data and control information to various SSs. Uplink channel is shared by various SSs for sending bandwidth requests and uplink data to BS. It supports both TDD and FDD for allocating bandwidth on uplink and downlink channel.

- In TDD, uplink and downlink channels may share the same frequency channel but do not transmit simultaneously. Each TDD frame has one downlink subframe followed by an uplink subframe as shown in figure 2.
- In FDD, uplink and downlink channels operate on separate frequencies. Downlink transmissions occur concurrently with uplink transmissions.

The downlink channel is a *broadcast channel* while the uplink channel is *time-shared* among SSs. BS maintains a continuous TDM link to the SSs on the downlink channel, whereas access in the uplink direction is by Demand Assigned Multiple Access TDMA (DAMA-TDMA). One key feature of 802.16 is that BS first allocates bandwidth to each SS to enable them to send requests for bandwidth needed to transmit uplink data. BS then assigns a variable number of physical slots to each SS for uplink data transmissions according to their bandwidth demand. This information is sent to all SSs through uplink control message.

A. Existing QoS Provisions of IEEE 802.16

IEEE 802.16 provides mechanisms to support QoS for both uplink and downlink traffic through SS and BS. The principal mechanism for providing QoS is to associate packets traversing the MAC interface into a *service flow*. A *service flow* is a MAC layer transport service that provides unidirectional transportation of packets in both uplink and downlink direction. A set of QoS parameters such as minimum reserved bandwidth, traffic priority etc. along with the direction are associated with each *service flow*. During the connection set up phase, these service flows are established and activated by BS and SS. A unique Connection Identifier (CID) is assigned to all activated service flows. Many higher layer sessions may operate over the same MAC layer CID if their QoS requirements are same. Apart from this, in order to support QoS in uplink direction, IEEE 802.16 standard provides the following features:

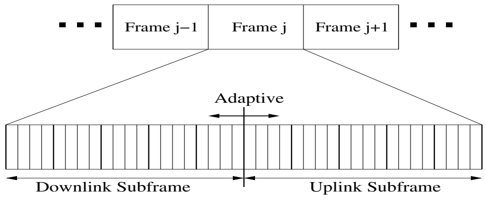

Fig. 2. 802.16 TDD Frame.

1) Uplink Scheduling Services: IEEE 802.16 specifies four scheduling services for uplink flows. Each connection in uplink direction is mapped to one of these services. BS scheduler follows a set of rules for each uplink service while allocating bandwidth to SSs for uplink transmission.

- *Unsolicited Grant service* (UGS): UGS service flow type is designed to support real time services that generate fixed units of data periodically, such as Voice over IP. Here the BS schedules a fixed size data grants periodically without an explicit request from the SS which eliminates the overhead and latency of SS requests to meet the flow's real-time needs. UGS connections are not allowed to use any request slots. The key parameters of an UGS flow are *Unsolicited Grant Size, Nominal Grant Interval*, and *Tolerated Grant Jitter* [1].
- *Real-time Polling Service* (rtPS): rtPS is designed to support real-time service flows that generate variable size data packets periodically, such as MPEG video. The service offers real-time, periodic, unicast request slots, which meet the flow's real-time needs and allow the SS to specify the size of the desired grant. rtPS connections are not allowed to use contention request slots. The key parameters of a rtPS flow are *Nominal Polling Interval, Tolerated Poll Jitter*, and *Minimum Reserved Traffic Rate* [1].
- *Non-real-time Polling Service* (nrtPS): nrtPS is designed to support non real-time service flows that require variable size data grants on a regular basis, such as high bandwidth FTP. It is similar to rtPS but offers unicast request slots less frequently and SS is allowed to use contention request slots. The key parameters of a nrtPS flow are *Nominal Polling*

Interval, Minimum Reserved Traffic Rate, and *Traffic Priority* [1].

- *Best Effort* (BE): This service is designed to support best effort traffic and offers no guarantee. SS is allowed to use both contention and unicast request slots. The key parameters of a BE flow are *Minimum Reserved Traffic Rate* and *Traffic Priority* [1].

2) Bandwidth Request and Grant Mechanisms: In IEEE 802.16, access in the uplink direction is by DAMA-TDMA. SSs use bandwidth request mechanism to specify uplink bandwidth requirement to the BS. BS polls SS by allocating bandwidth to them for the purpose of making bandwidth requests. Bandwidth is always requested on per connection basis. Bandwidth can be requested by sending a bandwidth request packet or piggybacking it with a data packet. Requests can be *aggregate* or *incremental*. When the BS receives an *incremental* bandwidth request, it adds the quantity of bandwidth requested to its current perception of the bandwidth needs of the connection. When the BS receives an *aggregate* bandwidth request, it replaces its perception of the bandwidth needs of the connection with the quantity of bandwidth requested.

IEEE 802.16 specifies two modes for granting bandwidth requested by SS.

- *Grant Per Connection* (GPC): In GPC mode, BS scheduler treats each connection separately and bandwidth is explicitly granted to each connection. SS transmits according to the order specified by the BS.
- *Grant Per Subscriber Station* (GPSS): In GPSS mode, BS scheduler treats all the connections from a single SS as one unit and grants bandwidth to SS. An additional scheduler is employed at SS which determines the service order for its connections in the granted slot.

GPSS mode is more scalable and efficient as compared to GPC. It is also capable of providing lower delay to real-time applications because SS is more intelligent in GPSS mode and can react quickly to the needs of real-time flows.

IEEE 802.16 has been developed keeping in view the stringent QoS requirements of various applications. But IEEE 802.16 standard does not suggest packet scheduling mechanisms for downlink and uplink flows so as to provide required QoS to various applications. Scheduling details are left as the responsibility of implementers. In the following section, we discuss the need for a QoS scheduling architecture for IEEE 802.16.

III. Motivation

For IEEE 802.16, scheduling mechanisms are required for both downlink and uplink flows. In the downlink direction, scheduling is relatively simple because BS knows the exact status of its queues and their QoS requirements, therefore, an existing scheduling mechanism will suffice. However, uplink scheduling is more complex as it needs to be in accordance with uplink QoS provisions provided by IEEE 802.16. A single scheduling mechanism for the entire system will not work since uplink scheduling involves both SS and BS. They need to coordinate with each other for uplink bandwidth allocation through request-grant protocol. While downlink scheduling requires only one scheduler at BS, uplink scheduling requires two components, one at the BS and one at the SS. The component at BS allocates bandwidth to SSs and the one at SS schedules uplink packets in the granted slot. In GPC mode, much of the scheduling complexity lies at BS while the uplink scheduler at SS is quite simple. In contrast, in GPSS mode, uplink scheduling logic is evenly distributed at both BS and SS. In addition to this, scheduling mechanisms at BS and SS must also provide QoS guarantees to various flows. Hence, an efficient and scalable QoS scheduling architecture for the complete system is required to maintain QoS and fairness for different types of downlink and uplink flows.

In this paper, we propose an efficient QoS scheduling architecture for IEEE 802.16 to provide tight QoS guarantees to various real-time applications. Our architecture is based on GPSS mode for granting uplink bandwidth requests due to its inherent advantages over GPC mode. Our main *design goals* are to provide delay and bandwidth guarantees to various applications and to maintain fairness among various flows while still achieving high bandwidth utilization. We have done comprehensive simulation analysis to show the effectiveness of our architecture in providing QoS guarantees to various real-time applications.

IV. Overview of the QoS Scheduling Architecture

Our proposed QoS scheduling architecture is a distributed architecture implementing GPSS mode for granting bandwidth to SSs. The architecture supports all four kinds of uplink services specified in IEEE 802.16 standard.

A. Design Goals

Our main *design goals* are as follows:

- To provide delay and bandwidth guarantees for various kinds of applications.
- To maintain fairness among various flows based on their priority.
- To achieve high bandwidth utilization.

Fairness in our context means that the QoS guarantees of a high priority flow are not affected by a low priority flow, both within an SS and across SSs. For example, consider a scenario shown in figure 1. It consists of two SSs and a single BS. $SS1$ has a UGS flow and a rtPS flow. $SS2$ has a rtPS flow and a BE flow. In this situation, our architecture needs to guarantee that the bursty BE traffic at $SS2$ does not affect the deadlines of rtPS flow at $SS2$ as well as delay-sensitive flows at $SS1$. In addition to this, each SS must be allocated minimum bandwidth reserved by them to ensure that real-time deadlines are met.

B. Design Decisions

In order to meet the above stated design goals we have taken following design decisions:

- We have chosen GPSS mode for granting bandwidth to SS as it is scalable, efficient and allows SS to react faster to the needs of real-time applications.
- BS allocates uplink bandwidth to each SS based on SSs fixed and variable requirements for various flows using max-min fair allocation strategy.
- Each SS distributes the allocated bandwidth among its various flows based on their priority and minimum bandwidth requirement using a combination of strict priority and Weighted Fair Queuing (WFQ) scheduling. UGS flows, reserved rtPS, nrtPS and BE flows, and remaining flows are served in that order using strict priority scheduling. Order of transmission among reserved flows is decided using WFQ scheduling. This design ensures that reserved flows get higher priority while fairness among them is achieved through WFQ scheduling.
- In the downlink direction, flows with minimum bandwidth reservation are served first. WFQ scheduling is used to determine order of packet transmission for reserved flows. Remaining bandwidth is assigned to best-effort flows i.e. flows with no bandwidth reservations.

C. Overview of the Architecture

The block diagram of our proposed QoS scheduling architecture is shown in figure 3. A brief overview of the working of this architecture is as follows:

- The following sequence of events take place at BS MAC layer:
  - The incoming IP packet is classified into one of the connections, shaped and placed in one of the downlink traffic queues.
  - BS also receives uplink data and request packets sent on uplink channel by various SSs. Data packets are handed to higher layer while request packets are classified and placed in the uplink grant queue accordingly. Periodic data and request grants generated by BS are also treated in the same manner as uplink bandwidth requests.
  - Both the downlink traffic queues and uplink grant queues are examined by BS and the total bandwidth is divided into downlink and uplink subframe accordingly.
  - BS then schedules downlink data packets in the bandwidth provided and creates downlink control message.
  - BS also allocates uplink bandwidth to various SSs based on their bandwidth demands and encodes this information in uplink control message.
  - At the start of frame, BS transmits downlink and uplink control message on downlink channel followed by downlink data for various SSs.
- The following sequence of events take place at SS MAC layer:
  - The incoming IP packet is classified into one of the connections, shaped and placed in one of the uplink traffic queues.
  - SS also receives downlink and uplink control messages sent by BS on downlink channel. SS listens to downlink channel for the entire downlink subframe duration to discover if there are any downlink packets intended for it. Downlink data packets addressed to it are sent to higher layer.
  - SS determines its uplink transmission time and duration of transmission by decoding uplink control message.
  - SS also generates bandwidth requests to be sent to BS.
  - SS then schedules uplink data packets and request packets in the allocated slot and transmits them accordingly.
  - After completing uplink transmission, SS again starts listening on downlink channel with the start of next frame.

A detailed description of the architecture is given in the following section.

V. Architecture Details

Various components of our architecture are as shown in figure 3. We now explain the working of each component in detail. Some of the components of our architecture are specified in the standard while others are introduced by us to meet QoS design goals. The following components are as required by the standard:

1) *BS Downlink Data Classifier*: This component classifies each incoming IP packet to be transmitted on downlink channel into one of *Downlink Traffic Queues*. In our design, there are $n + 1$ such queues, where n is the number of reserved downlink flows as shown in figure 3. There is a separate queue corresponding to each flow with minimum bandwidth reservations called as *Type 1* queues. Each queue of *Type 1* is identified by a unique CID. All the remaining flows are mapped to a single queue called as *Type 2* queue. Classification includes mapping the IP packet to a particular CID and then placing it in a

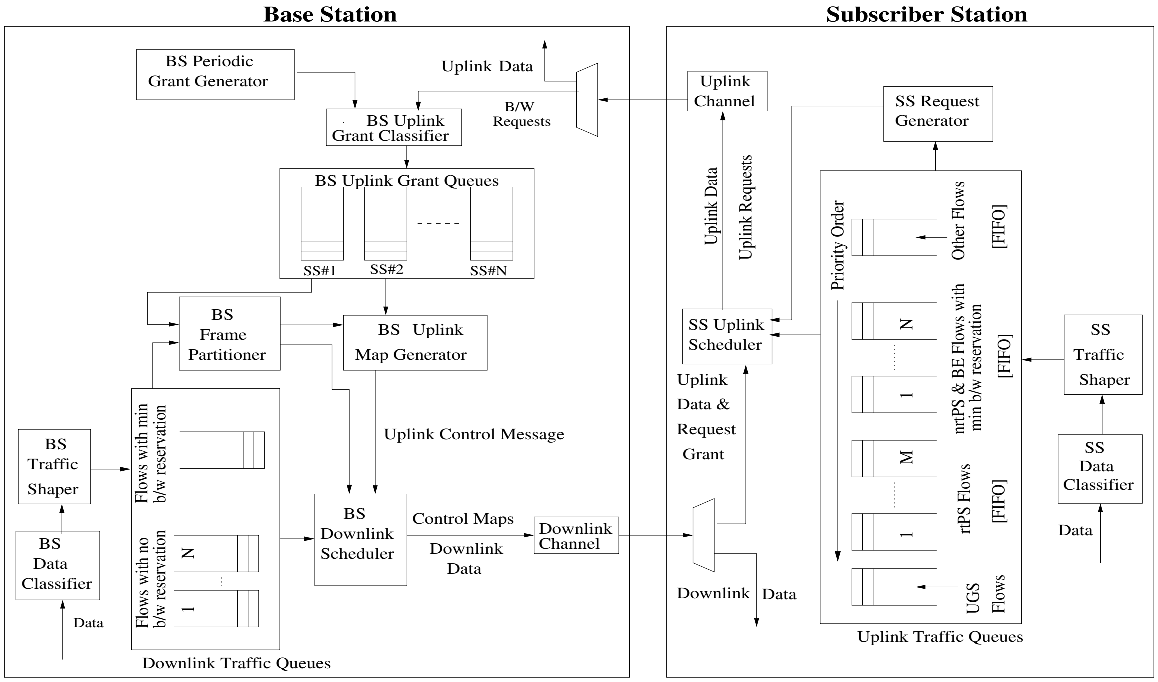

Fig. 3. QoS Scheduling Architecture for IEEE 802.16.

proper queue. A set of classifiers are associated with each CID. Each classifier consists of a set of parameters such as IP Type of Service value, IP Source and Destination address etc. A priority value is also associated with each classifier. These classifiers are applied to each packet in the order of priority. The highest priority classifier which matches the packet determines the CID for the packet. Finally, packet is placed in one of the *Downlink Traffic Queues* depending upon its CID and bandwidth reservations for that connection.

2) *BS Traffic Shaper*: *Traffic Shaper* is employed at BS to examine the incoming traffic according to its parameters and shape the traffic which violates the parameters.
3) *BS Periodic Grant Generator*: This component generates periodic data and request grants for uplink flows. It keeps track of all the admitted UGS, rtPS and nrtPS flows. It generates one data grant per nominal grant interval for each active uplink UGS flow. Each grant is generated at time t_k where $t_k = t_0 + k * nominal\_grant\_interval$ and marked with a deadline equal to $t_k + jitter$. Here t_0 is the initial reference time. Similar method is used for generating request grants for rtPS and nrtPS flows.
4) *BS Uplink Grant Classifier*: This component handles classification of periodic grants and bandwidth requests. Periodic grants generated by *BS Periodic Grant Generator* and uplink bandwidth requests transmitted by various SSs are fed to this component. It classifies each grant and request packets into one of uplink grant queues. In our design, there is an uplink grant queue corresponding to each SS as shown in figure 3. Classification is done by mapping CID to the corresponding SS.
5) *BS Frame Partitioner*: Our architecture supports TDD for allocating bandwidth for downlink and uplink channel. In our design, we use a fixed partition scheme which divides the total frame bandwidth equally between downlink and uplink subframe.
6) *SS Uplink Data Classifier*: This component classifies each incoming IP packet to be transmitted on uplink channel into one of *Uplink Traffic Queues*. In our design, each SS has one queue for UGS flows called as *Type 1* queue, a separate queue for each rtPS flow called as *Type 2* queues, a separate queue for each nrtPS and BE flow with minimum bandwidth reservations called as *Type 3* queues. All the remaining nrtPS and BE flows are mapped to a single queue called as *Type 4* queue. These set of queues are called *Uplink Traffic Queues*. The classification process is same as in *BS Downlink Data Classifier*.
7) *SS Traffic Shaper*: The working of this component is similar to *BS Traffic Shaper*.

The following components have been introduced to satisfy the design goals of our architecture.

1) *SS Request Generator*: Bandwidth request for various connections to be transmitted in an uplink subframe is generated by *SS Request Generator*. For each connection, *aggregate* request is generated. *Aggregate* request for a connection is equal to the current queue length for that connection.
2) *BS Uplink Map Generator*: This component is responsible for allocating bandwidth to each SS for uplink transmission. We use *max-min fair allocation* strategy for this purpose [7]. Bandwidth is allocated on per SS basis rather than on per connection basis. Amount of bandwidth allocated to each SS is based on following:
   - Amount of bandwidth requested by each SS for transmitting uplink data.
   - Periodic bandwidth requirement of SSs UGS flows.
   - Bandwidth required for making additional bandwidth requests.

   Algorithm 1 gives the algorithm for allocating uplink bandwidth to each SS. Input to this algorithm is total bytes requested per flow by each SS. BS calculates total bytes requested for each flow per SS by examining uplink grant queues. BS also calculates total number of bytes requested for each uplink flow across all SS. Uplink bandwidth is distributed among various SSs in two stages using *max-min fair allocation* strategy.
   - In the first stage, uplink bandwidth is distributed among four uplink flows. In the first round, each uplink flow is allocated its percentage of bandwidth, normalized by its weight. This ensures that high priority reserved flows are always satisfied before low priority flows. In the second round, excess bandwidth allocated to any flow is distributed among unsatisfied flows in proportion to their weight. This process continues until either all four uplink flows are satisfied or no bandwidth is available.
   - In the second stage, bandwidth allocated to each flow is distributed among all SSs. In the first round, bandwidth allocated to UGS flows is equally distributed among all SSs. In the second round, excess bandwidth allocated to any SS more than its requirement for UGS flows is evenly distributed among unsatisfied SSs. This process continues until either UGS requirement of all SSs are satisfied or no bandwidth for UGS flows is available. The above process is repeated for rtPS, nrtPS and BE flows too. The order of transmission among SSs is decided according to the deadline of UGS data to be transmitted by each SS.

Some of the important notations used in the algorithm are as follows:

- $totalUplinkBytes$: Total bytes available for uplink transmission.
- $rtPS\_Req[i]$: Bytes requested by i^{th} SS for rtPS flows.
- $nrtPS\_Req[i]$: Bytes requested by i^{th} SS for nrtPS flows.
- $BE\_Req[i]$: Bytes requested by i^{th} SS for BE flows.
- $CBR\_Req[i]$: UGS data and request grants for i^{th} SS.
- $CBRWeight$: 4 i.e. Weight assigned to UGS flows and request grants.
- $rtPSWeight$: 3 i.e. Weight assigned to rtPS flows.
- $nrtPSWeight$: 2 i.e. Weight assigned to nrtPS flows.
- $BEWeight$: 1 i.e.Weight assigned to BE flows.
- $flow\_Req[i]$: Total bytes requested for i^{th} uplink flow across all SS.
- $flow\_Alloc[i]$: Total bytes allocated for i^{th} uplink flow across all SS.
- $flow\_Weight[i]$: Weight assigned to i^{th} uplink flow.
- $flow\_Req\_SS[i][j]$: Total bytes requested for i^{th} uplink flow by j^{th} SS.
- $SSUplinkBytes[i]$: Total bytes granted to i^{th} SS for uplink transmission.

Algorithm 1 BS Uplink Map Generator

flow_Req[1] = flow_Req[2] = 0;
flow_Req[3] = flow_Req[4] = 0;
flow_Alloc[1] = flow_Alloc[2] = 0;
flow_Alloc[3] = flow_Alloc[4] = 0;
flow_Weight[1] = CBRWeight;
flow_Weight[2] = rtPSWeight;
flow_Weight[3] = nrtPSWeight;
flow_Weight[4] = BEWeight;
{Flow number 1 for UGS, 2 for rtPS, 3 for nrtPS and 4 for BE.}
totalSS = N; {Total number of SS is N.}
for all $j = 1; j \leq totalSS; j++$ **do**
    flow_Req_SS[1][j] = CBR_Req[j];
    flow_Req_SS[2][j] = rtPS_Req[j];
    flow_Req_SS[3][j] = nrtPS_Req[j];
    flow_Req_SS[4][j] = BE_Req[j];
    flow_Req[1] += CBR_Req[j];
    flow_Req[2] += rtPS_Req[j];
    flow_Req[3] += nrtPS_Req[j];
    flow_Req[4] += BE_Req[j];
end for
{Uplink bandwidth is distributed among all four

```
flows using max-min allocation.}
while Σ_{i=1}^{4} flow_Weight[i] != 0 && totalUplinkBytes != 0 do
  extraBytes = 0;
  totalWeight = Σ_{i=1}^{4} flow_Weight[i];
  for all i = 1; i ≤ 4; i++ do
    flow_Alloc[i] += ⌊flow_Weight[i] / totalWeight⌋ * totalUplinkBytes;
    if flow_Weight[i] != 0 && flow_Req[i] ≤ flow_Alloc[i] then
      temp = flow_Alloc[i];
      extraBytes += temp - flow_Req[i];
      flow_Alloc[i] = flow_Req[i];
      flow_Weight[i] = 0;
    end if
  end for
  totalUplinkBytes = extraBytes;
end while
{For all uplink flows.}
for all i = 1; i ≤ 4; i++ do
  totalSS = N; {Total number of SS is N.}
  if flow_Req[i] == flow_Alloc[i] then
    for all j = 1; j ≤ totalSS; j++ do
      SSUplinkBytes[j] += flow_Req_SS[i][j]; flow_Req_SS[i][j] = 0;
    end for
  else
    for all j = 1; j ≤ totalSS; j++ do
      satisfied[j] = 0; {Keeps track of SSs with unsatisfied requests.}
    end for
    {Bandwidth allocated to i^th flow is distributed among all SSs using max-min allocation.}
    while flow_Alloc[i] > 0 do
      extraBytes = 0;
      flow_Alloc_SS = ⌊flow_Alloc[i] / totalSS⌋;
      for all j = 1; j ≤ N; j++ do
        if satisfied[j] == 0 then
          if flow_Alloc_SS ≥ flow_Req_SS[i][j] then
            temp = flow_Req_SS[i][j];
            SSUplinkBytes[j] += temp;
            temp1 = flow_Alloc_SS;
            temp2 = flow_Req_SS[i][j];
            extraBytes += temp1 - temp2;
            flow_Req_SS[i][j] = 0;
            satisfied[j] = 1; totalSS -= 1;
          else
            temp = flow_Alloc_SS;
            SSUplinkBytes[j] += temp;
            flow_Req_SS[i][j] -= temp;
          end if
        end if
      end for
      flow_Alloc[i] = extraBytes; {Extra Bytes are distributed among unsatisfied SSs.}
    end while
  end if
end for
```

3) *BS Downlink Scheduler*: This component is responsible for scheduling packets from *Downlink Traffic Queues* at BS for transmission on downlink channel. In our design, bandwidth allocation to both types of queues follows strict priority discipline. Packets from *Type 1* queues are transmitted first. Remaining bandwidth after transmission of all packets from *Type 1* queues, if any, is allocated to packets from *Type 2* queue. In our implementation, we use Weighted Fair Queuing (WFQ) scheduling algorithm to serve the flows fed to *Type 1* queues [8]. WFQ weight for the flows of *Type 1* queues is assigned based on the minimum bandwidth reserved by them. Packets from a single queue are served in the FIFO order.

4) *SS Uplink Scheduler*: This component is responsible for scheduling packets from *Uplink Traffic Queues* at SS for transmission on uplink channel. Each SS determines its slot duration and transmission time through the uplink control message sent by BS. Bandwidth allocated to each SS for uplink transmission must be properly allocated to various uplink flows so as to satisfy their QoS requirement.
In our design, each SS uses a combination of strict priority scheduling and WFQ scheduling to serve various uplink flows. *Type 1* queue for UGS flows is the highest priority queue and is served first. SS then transmits bandwidth request packets for various flows other than UGS flow. Packets from *Type 2* and *Type 3* queues are served next. We use WFQ scheduling algorithm to serve the flows fed to *Type 2* and *Type 3* queues. WFQ weight for the flows of *Type 2* and *Type 3* queues is assigned based on the minimum bandwidth reserved by them. The remaining bandwidth after serving all higher priority flows is allocated to *Type 4* queue. Packets from a single queue are served in the FIFO order.
Suppose an SS has four uplink flows of each type. Both rtPS and nrtPS flows has minimum bandwidth reservation while BE flow has made no bandwidth reservation. As per our design, UGS flow is mapped to *Type 1*, rtPS flow to *Type 2*, nrtPS flow to *Type 3* and BE flow to *Type 4*. At the time of uplink transmission, SS transmits packets from *Type 1* queue first. Packets from *Type 2* and *Type 3* queues are transmitted in the remaining time in order of their increasing finish times. Packets from *Type 4* queue

are transmitted next in the remaining time left after all packets from high priority queues are transmitted.

VI. Simulation Results

A. *802.16 Simulator*

We have used Qualnet 3.6 for implementing and simulating IEEE 802.16 MAC along with our architecture. Qualnet 3.6 simulator does not provide a patch for IEEE 802.16 MAC and PHY. Therefore, we have implemented IEEE 802.16 MAC in Qualnet 3.6. IEEE 802.16 MAC is not tied to any particular physical layer specification. Since IEEE 802.11b PHY was readily available we have used it as the physical layer for the simulation. Since our study does not depend on channel properties, we feel that the use of IEEE 802.11b PHY for quick simulation analysis is justified.

B. *Simulation Setup*

The simulation topology consists of one BS and a number of SSs. All the nodes use 802.16 MAC layer and 802.11b physical layer. Channel bandwidth is 11 Mbps which is divided equally between downlink and uplink subframe. Each frame is of 10 milli-second duration. We have assumed that the channel is error-free so that each packet is successfully received at the destination. Also, number of subscriber stations cannot change dynamically during the simulation. Each SS has a combination of various application flows such as VOIP, FTP, Telnet etc. Each of these application level flows are mapped appropriately to one of the uplink scheduling flows. We have chosen average delay of uplink flows at SSs and effective bandwidth utilization as performance metrics.

C. *Simulation Results*

We have performed the following experiments to illustrate the performance of our architecture.

- The first experiment aims at showing the maximum number of subscriber stations that can be supported. Here we have performed a number of simulation runs each with different number of SSs. In every run, each SS has same number of uplink flows with similar QoS parameters. Total load offered by each SS is approximately $6-7\%$ of the available uplink bandwidth. Each simulation is run for 20 seconds. We have measured average delay of each uplink flows across all SSs for every simulation run. Figure 4 shows the graph obtained for the same. It is observed that initially, average delay of all the flows is nearly equal, around 5.5 milli-second. As the number of SS increases, average delay of BE flows starts increasing rapidly. There is a small increase in average delay of nrtPS too. On increasing the number of SSs in the system beyond 15, there is a rapid increase in average delay of non-real time flows. At the same time average delay of rtPS flows also starts increasing slowly. Therefore, on an average 15 subscriber stations can be supported.

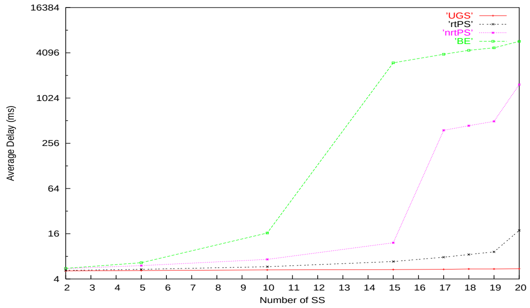

Fig. 4. Average Delay Vs Number of SS.

- The second experiment aims at showing the maximum effective bandwidth utilization achieved with our architecture. Here we have performed a number of simulation runs each with increasing offered load while keeping number of SSs same across all runs. We have setup a topology with one BS and five SSs. Offered load is equally distributed among all SSs. Each simulation is run for 25 seconds. Both offered load and bandwidth utilization are calculated as a percentage of link bandwidth. Control packets are not included in bandwidth utilization calculation. Figure 5 shows the graph for effective bandwidth utilization with increasing offered load. Effective bandwidth utilization increases linearly as the offered load is increased. On increasing the offered load beyond 100%, effective bandwidth utilization remains more or less constant. Therefore, maximum effective bandwidth utilization achieved with our architecture is around 93%. We have also calculated bandwidth utilized for each of the four uplink flows as shown in figure 6. We have placed no limit on the bandwidth provided to UGS flows. As expected, bandwidth utilization of all uplink flows increases linearly on increasing offered load. But the bandwidth utilization of lower priority flows starts decreasing as the total offered load increases beyond 90%. At this point, percentage of load offered by UGS and rtPS flows is more as compared to nrtPS and BE flows. This shows that our architecture provides more bandwidth to high priority flows as compared to lower priority flows in order to meet the deadlines of high priority traffic.
- Third experiment aims at showing that our architecture meets delay guarantees of real-time applications and maintains fairness among flows in accordance to their priority. For this, we have simulated three different scenarios as follows.
  - *Scenario 1*: This scenario is setup to show that the delay guarantees of real-time flows at an SS are not

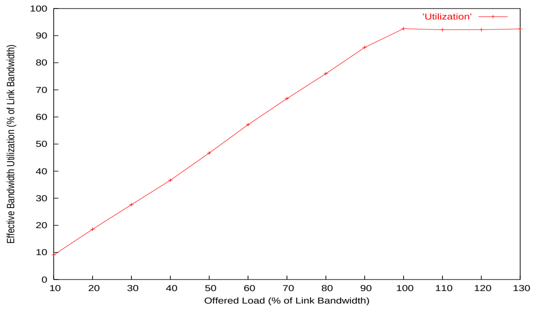

Fig. 5. Effective Bandwidth Utilization Vs Offered Load.

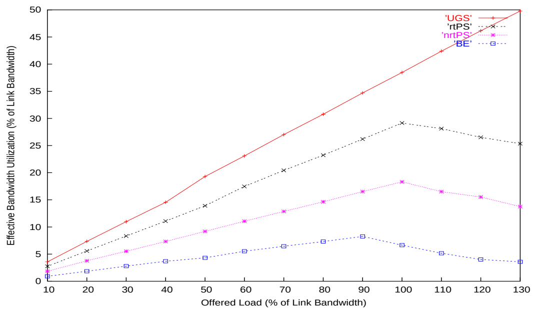

Fig. 6. Effective Bandwidth Utilization Vs Offered Load.

affected by the amount of lower priority load offered by that SS. Here, load offered by higher priority traffic is more than the lower priority one. In this scenario, a simulation topology with one BS and five SS has been setup. Each SS has four uplink flows. Total load offered to the system is 90% of the link bandwidth which is equally distributed among all SSs. Out of this, 40% load is offered by UGS flows, 30% by rtPS flows, 20% by nrtPS flows and 10% by BE flows. Simulation is run for 25 seconds. We have measured average delay experienced by each uplink flow for each SS. Figure 7 shows average delay comparison of various uplink flows for one SS. It is clearly visible from the graph that average delay of UGS and rtPS flows is no more than one frame duration. Also, average delay of UGS flows is almost constant as they are given highest priority at the time of transmission.

– *Scenario 2*: This scenario is also setup to show that the delay guarantees of real-time flows at an SS are not affected by the amount of lower priority load offered by that SS. This scenario is similar to previous one except that the load offered by lower priority traffic is more than the higher priority one. Here maximum load is offered by BE flows. 40% load is offered by BE flows, 30% by nrtPS flows, 20% by rtPS flows and 10% by UGS flows. Simulation is

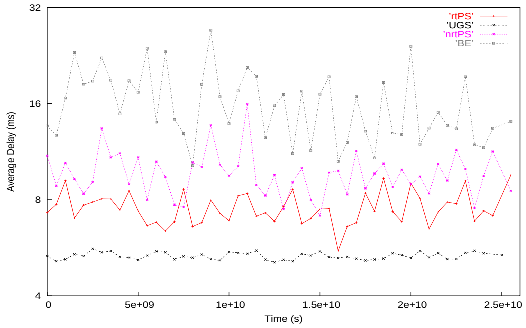

Fig. 7. Average Delay Vs Time[Scenario 1].

run for 25 seconds. We have measured average delay experienced by each uplink flow for each SS. Figure 8 shows average delay comparison of various uplink flows for one SS. Here also same trend is observed for the average delay experienced by various uplink flows. Regardless of the high amount of load offered by BE flows, delay guarantees of real-time traffic are met.

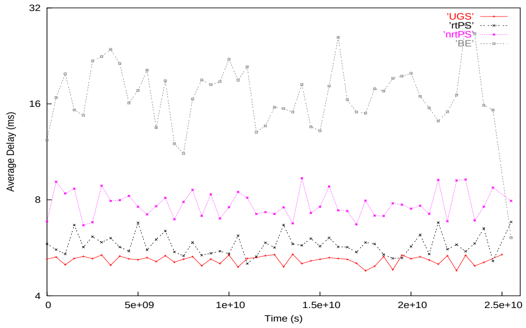

Fig. 8. Average Delay Vs Time[Scenario 2].

– *Scenario 3*: This scenario is setup to show that real-time flows are given priority across SSs too so that their delay guarantees are not affected by the amount of lower priority load offered by another SSs. In this scenario, a simulation topology with one BS and three SS has been setup. Total load offered to the system is 90% of the link bandwidth. Here different flows have been setup at various SS. SS 1 has one UGS flow and one rtPS flow. SS 2 has one UGS flow and one nrtPS flow. SS 3 has one UGS flow and one BE flow. Load offered by all UGS flows is equal. Out of remaining flows, BE flows offer maximum load while rtPS offer minimum load. Figure 9 shows average delay comparison of various uplink flows. Here also UGS flows of all the SSs have a much lower delay as compared to other flows. Also, rtPS flow at SS 2 gets its share of bandwidth irrespective of other flows at SS 2 and 3. As a result delay guarantees of high

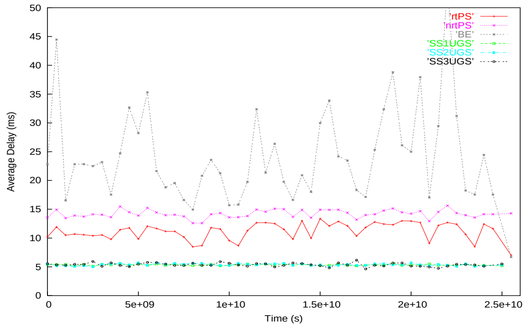

Fig. 9. Average Delay Vs Time[Scenario 3].

priority flows are always satisfied.

Results obtained in the above three scenario's prove that our architecture is capable of providing tight delay guarantees to various real-time applications.

VII. Related Work

IEEE 802.16 MAC has not been widely studied and there are very few proposed architectures in the literature. In [2], G. Chu et al have suggested a QoS architecture based on priority scheduling and dynamic bandwidth allocation employing GPSS mode for granting bandwidth to SS. BS upstream scheduler implements Weighted Round Robin (WRR) [9] for allocating bandwidth to various SS. Uplink scheduler at each SS implements fair queuing scheduler for each uplink service flow with strict priority scheduling among the various flows. In [3], M. Hawa et al have proposed an uplink scheduling architecture to support bandwidth and delay QoS for both DOCSIS [4] and IEEE 802.16 employing GPC mode for granting bandwidth to SS. BS's upstream scheduler implements a mix of priority scheduling and fair queuing scheduling in order to grant bandwidth to various connections at each SS. SS's simply transmits in their allocated slots. At BS, UGS flows are given highest priority for bandwidth allocation. Remaining flows are served using WFQ. In both the proposals, authors have mentioned that the architecture is capable of supporting diverse QoS requirements of various types of applications.

However in [2], it does not specify how to assign weights to each SS for implementing WRR. Another disadvantage is that a simple WRR with weights being assigned in proportion to the amount of bandwidth requested by each SS will not work. This may result in giving higher weight to an SS with large amount of bandwidth required for BE flows. This in turn can effect deadlines of real-time flows at other SSs. Method of assigning weights to various uplink flows at SS is also not mentioned in the paper.

Also in [3], treatment given to unreserved flows may create problem to reserved flows in case there are many unreserved flows and reserved bandwidth by various flows is less as compared to unreserved bandwidth. Another disadvantage is that the proposed scheduler implements GPC mode for uplink bandwidth grant as GPSS is not supported by DOCSIS standard. However, GPSS mode is more efficient as compared to GPC. In addition to this, no simulation results have been presented in both the proposals to show the goodness of the architecture. Downlink scheduling mechanisms at BS are not discussed either.

VIII. Conclusion and Future Work

In this paper we have presented an efficient QoS scheduling architecture for IEEE 802.16. The main purpose of the architecture is to provide tight QoS guarantees to various applications and to maintain fairness among them while still achieving high bandwidth utilization. Our architecture supports diverse QoS requirements of all four kinds of service flows specified in IEEE 802.16 standard. We have also presented simulation analysis of our architecture integrated with IEEE 802.16 MAC. We have shown, through simulation, that our architecture is capable of achieving high bandwidth utilization. Simulation results show that sufficient bandwidth is allocated to high priority flows so that their QoS guarantees are always met. It is evident through the simulation results that lower priority traffic does not affect QoS guarantees of high priority real-time traffic i.e. fairness is maintained among flows at an SS and across SSs too.

References

[1] IEEE 802.16-2001, "IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems," Apr. 8, 2002.

[2] G. Chu, D. Wang, and S. Mei, "A QoS architecture for the MAC protocol of IEEE 802.16 BWA system," *IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions*, vol. 1, pp. 435–439, June 2002.

[3] M. Hawa and D. W. Petr, "Quality of Service Scheduling in Cable and Broadband Wireless Access Systems," *Tenth IEEE International Workshop on Quality of Service*, pp. 247–255, May 2002.

[4] Data Over Cable Service Interface Specifications (DOCSIS), "Radio Frequency Interface Specification," *SP-RFIv1.1-107-010829*.

[5] Scalable Networks Inc., "The qualnet 3.6 Simulator." `http://www.scalable-networks.com`, 2004.

[6] C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "IEEE Standard 802.16: A Technical Overview of the WirelessMAN^{TM} Air Interface for Broadband Wireless Access," *IEEE Communications Magazine*, vol. 40, pp. 98–107, June 2002.

[7] S. Keshav, *An Engineering Approach to Computer Networking*. Pearson Education, Sixth ed., 2003.

[8] A. K. Parekh and R. G. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Trans. Netw.*, vol. 2, no. 2, pp. 137–150, 1994.

[9] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.

[10] Andrew S. Tanenbaum, *Computer Networks*. Prentice-Hall India, Fourth ed., 2003.