# An Efficient QoS Scheduling Architecture for IEEE 802.16 Wireless MANs

**MTech Dissertation**

submitted in partial fulfillment of the requirements

for the degree of

**Master of Technology**

by

**Supriya Maheshwari**

**(Roll No. 03329026)**

under the guidance of

**Prof. Sridhar Iyer**

and

**Prof. Krishna Paul**

**Kanwal Rekhi School of Information Technology**

**Indian Institute of Technology Bombay**

**2005**

*Dedicated to my family*

# 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. We propose an efficient 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.

# Contents

# List of Figures

# Abbreviations

| | | |
|---|---|---|
| BE | : | Best Effort |
| BS | : | Base Station |
| BWA | : | Broadband Wireless Access |
| CID | : | Connection Identifier |
| DAMA | : | Demand Assigned Multiple Access |
| DCD | : | Downlink Channel Descriptor |
| DL-MAP | : | Downlink Map |
| DOCSIS | : | Data Over Cable Service Interface Specification |
| FDD | : | Frequency Division Duplexing |
| GPC | : | Grant Per Connection |
| GPSS | : | Grant Per Subscriber Station |
| MAN | : | Metropolitan Area Network |
| nrtPS | : | Non Real-Time Polling Service |
| rtPS | : | Real-Time Polling Service |
| SS | : | Subscriber Station |
| TDD | : | Time Division Duplexing |
| TDMA | : | Time Division Multiple Access |
| TDM | : | Time Division Multiplexing |
| UCD | : | Uplink Channel Descriptor |
| UGS | : | Unsolicited Grant Service |
| UL-MAP | : | Uplink Map |

# Chapter 1

# Introduction

## 1.1 Broadband Wireless Access

The rapid growth in demand for high-speed Internet access for residential and business customers has created a demand for “last-mile” broadband access. However, providing “last-mile” broadband access with fiber or coax cable can be very expensive, especially in developing countries. A cheaper solution is Broadband Wireless Access (BWA). Broadband is a transmission facility where bandwidth is wide enough to carry multiple voice, video or data channels simultaneously. BWA is capable of providing high speed network access to a broad geographic area with rapid flexible deployment at low cost. Other advantages of BWA are high scalability, lower maintenance and upgrade costs.

The IEEE 802.16 standard was developed to produce high performance BWA systems. The standard specifies a WirelessMAN air interface for fixed point to multi-point BWA systems [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 2.1. It standardizes a common MAC protocol that works with various physical layer specifications. It has been designed to address systems operating in 10-66 GHz frequency range. BS is the central entity and coordinates transmission in both the directions.

## 1.2 Need for a QoS Scheduling Architecture for IEEE 802.16

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. In both downlink and uplink directions, packets traversing the MAC layer interface are associated with a service flow. A unique set of QoS parameters such as delay, bandwidth etc are associated with each MAC level service flow. In order to better support QoS in uplink direction, standard specifies uplink scheduling services for heterogeneous classes of traffic and bandwidth request-grant mechanisms. Each uplink flow is allocated bandwidth for transmission depending on the amount of bandwidth requested and its scheduling service type. However, 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.

Therefore, 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, hence, 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. Hence, an efficient QoS scheduling architecture for the complete system is required to maintain QoS and fairness for different types of downlink and uplink flows.

## 1.3 Problem Statement

IEEE 802.16 has been developed keeping in view the stringent QoS requirements of various applications. 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.

We propose an efficient QoS scheduling architecture for IEEE 802.16 Wireless MANs with a fixed point-to-multipoint topology. Our main design objectives are to provide delay and bandwidth guarantees to QoS sensitive applications and maintain fairness among various flows while still achieving high bandwidth utilization.

## 1.4 Contributions of this Project

The significant contributions of this work are:

- An efficient QoS scheduling architecture for IEEE 802.16 WirelessMANs.
- Implementation of IEEE 802.16 MAC along with our proposed QoS scheduling architecture in Qualnet 3.6.
- Simulation analysis of our proposed architecture to show the effectiveness of our architecture in providing QoS guarantees to various real-time applications.

## 1.5 Dissertation Outline

As a starting point we did extensive study of IEEE 802.16 standard which we briefly describe in chapter 2. We present a brief overview of our proposed QoS scheduling architecture in chapter 3. In that chapter we discuss our main design goals, various design decisions and the rationale behind them. Chapter 4 provides a detailed description of various components of our architecture. In order to perform the simulation analysis, the architecture has to be integrated with IEEE 802.16 MAC implementation. We chose Qualnet 3.6 for this purpose which is a well known network simulator. However, IEEE 802.16 MAC patch is not available for Qualnet 3.6. In chapter 5, we discuss the implementation of IEEE 802.16 MAC and our architecture in Qualnet 3.6. In chapter 6 we present simulation results in which we show that our architecture meets the QoS guarantees of various kinds of applications and achieves high bandwidth utilization. Chapter 7 deals with the survey of some of the work done in 802.16 domain. We finally conclude the report by pointing out the ways in which our work could be extended.

# Chapter 2

# IEEE 802.16 - Wireless MAN Standard

IEEE standard 802.16 defines WirelessMAN air interface for fixed point-to-multipoint BWA systems that are capable of providing multiple services [1]. The standard gives specification for medium access control layer and physical layer of the OSI reference model. The standard currently addresses 10-66 GHz frequency range. This chapter describes IEEE 802.16 standard in brief.

## 2.1 Architecture

A schematic wireless metropolitan area network is shown in figure 2.1. It consists of a central radio BS and a number of SSs. The BS is connected to public networks and can handle multiple sectors simultaneously. The SS includes buildings like small office, home office, multi-tenant customers and small-medium enterprise. The 802.16 WirelessMAN provides network access to buildings through exterior antennas communicating with the BS. Each SS consist of a number of users. Both BS and SS are fixed(stationary) whereas users inside a building may be fixed or mobile. Currently 802.16 WirelessMAN specifies technology for bringing network to a building only, users inside a building can connect to each other using any conventional in-building network.

The 802.16 protocol stack is illustrated in figure 2.2 [2]. The physical layer consists of two sublayers, one of which is physical medium dependent and deals with the actual transmission. The other sublayer above it is Transmission Convergence (TC) sublayer to hide the different transmission technologies from the medium access control layer.

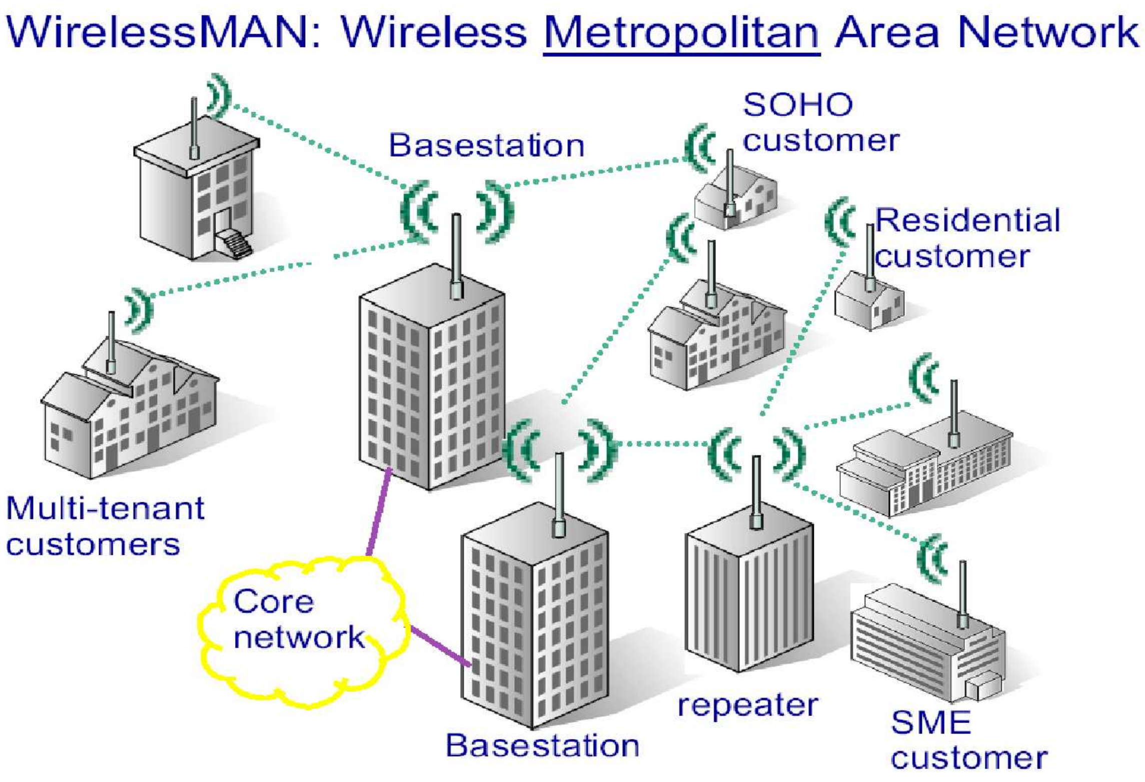

Figure 2.1: Wireless Metropolitan Area Network.

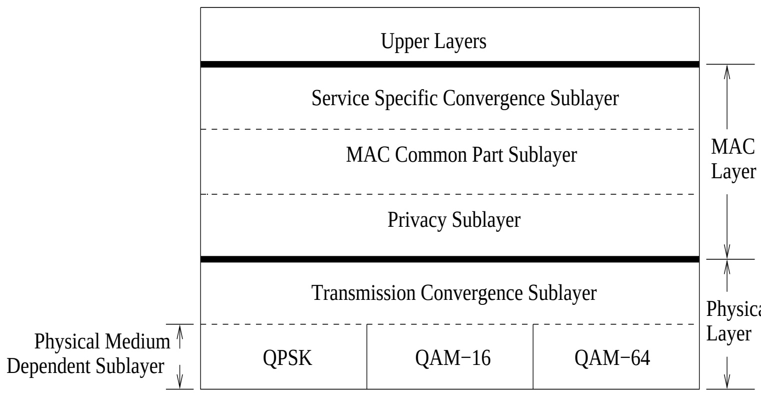
Upper Layers
Service Specific Convergence Sublayer
MAC Common Part Sublayer
Privacy Sublayer
Transmission Convergence Sublayer
QPSK
QAM−16
QAM−64
MAC Layer
Physical Layer
Physical Medium Dependent Sublayer

Figure 2.2: The 802.16 Protocol Stack.

The next layer of the stack is medium access control layer which consists of three sublayers as follows.

1. The bottom one is Privacy Sublayer which deals with privacy and security. Security is a major issue for public outdoor networks. This sublayer provides authentication for network access and connection establishment to avoid theft of service. It also provides encryption, decryption and key exchange for data privacy.

2. Next sublayer is the MAC Common Part Sublayer (CPS). The 802.16 MAC is connection oriented. It is designed to make efficient use of spectrum. It supports hundreds of users per channel and provides high bandwidth to the users. It accommodates both continuous and bursty traffic in order to support variety of services such as constant bit rate, real-time variable bit rate and so on.

3. The service specific Convergence Sublayer (CS) provides interface to the network layer above the MAC layer. Its function is to map transport layer specific traffic to 802.16 MAC which is flexible enough to carry any type of traffic. The 802.16 standard specifies two service specific CSs. The ATM CS is defined for mapping ATM services and the Packet CS is defined for mapping packet services to and from 802.16 MAC connections. It also preserves or enables QoS, and enable bandwidth allocation.

## 2.2 Physical Layer Details

IEEE 802.16 standard specifies one physical layer specification which operates in 10-66 GHz frequency bands. Waves in this spectrum are short in length, due to which, line-of-sight propagation is necessary. Also millimeter waves in this frequency range travel in straight line, as a result of which the BS can have multiple antennas, each pointing at a different sector. This is shown in figure 2.3 [2]. Each sector has its own users and is independent of the adjoining ones. Due to sharp decline in signal strength of millimeter waves with distance from the BS, signal to noise ratio also drops very fast. For this reason, 802.16 uses three different modulation schemes with Forward Error Correction (FEC) to make the channel look better than it really is. The 802.16 PHY supports adaptive burst profiling in which transmission parameters, including the modulation and coding schemes,

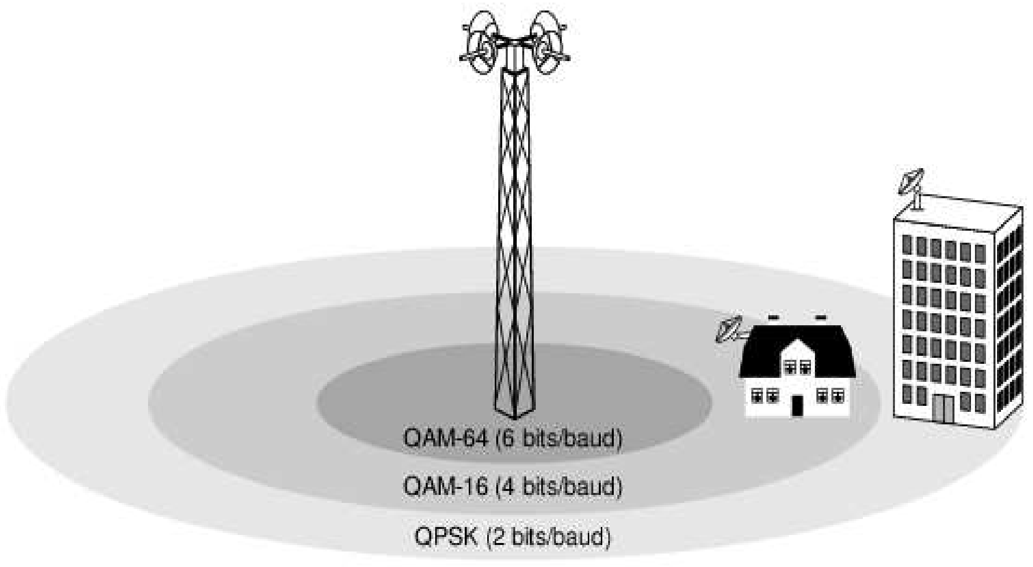

Figure 2.3: The 802.16 Transmission Environment.

may be adjusted individually to each SS on a frame-by-frame basis. It supports channels as wide as 28 MHz with data rates upto 134 Mbps [3]. Channels are additionally separated in time via a frame i.e. each 28 MHz carrier is subdivided into frames that are repeated continuously. Duration of each frame can be 0.5 ms, 1 ms or 2 ms. A frame is again subdivided in time via physical slots. Number of physical slots $n$ in a frame is a function of symbol rate and frame duration i.e. $n = (SymbolRate \times FrameDuration)/4$. Each frame is also divided into two logical channels, downlink channel and uplink channel. There is a downlink subframe corresponding to downlink channel and uplink subframe corresponding to uplink channel.

The downlink channel is a *broadcast channel*. It is used by BS for transmitting downlink data and control information to various SSs. The BS is completely in control for the downlink direction. It maps the downstream traffic onto time slots and transmits a TDM signal, with individual SS allocated time slot serially. Each SS receives all portions of the downlink except for those bursts whose burst profile is either not implemented by the SS or is less robust than the SSs current operational downlink burst profile. Half-duplex SSs do not attempt to listen to portions of the downlink coincident with their allocated uplink transmission.

The uplink channel is *time-shared* among all SSs. The BS is responsible for granting bandwidth to individual SSs in the uplink direction through Demand Assigned Multiple

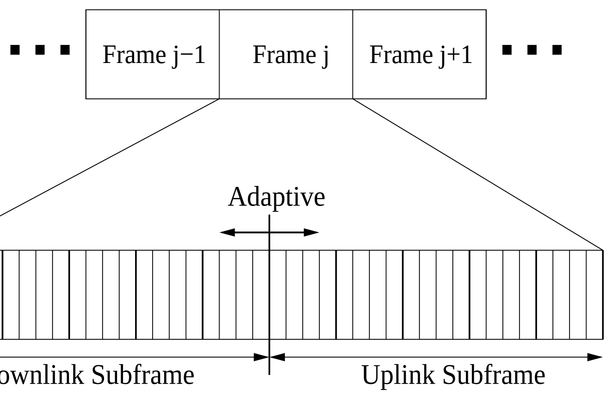

Figure 2.4: The 802.16 TDD Frame Structure.

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.

### 2.2.1 Duplexing Techniques

The IEEE 802.16 supports both Time Division Duplexing (TDD) and Frequency Division Duplexing (FDD) for allocating bandwidth on uplink and downlink channel [1].

- 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 2.4. Physical slots allocated to each subframe may vary dynamically according to bandwidth need in each direction. Between two subframes a slot is used as a guard time to allow stations to switch direction.

- In FDD, uplink and downlink channels operate on separate frequencies. Downlink transmissions occur concurrently with uplink transmissions. Therefore, duration of a subframe (downlink or uplink) is same as the frame duration. On downlink, both full-duplex and half-duplex SSs are supported simultaneously. The FDD frame structure is shown in figure 2.5.

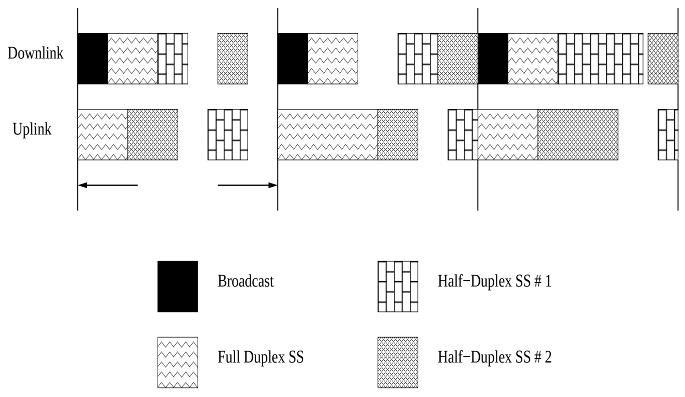

Figure 2.5: The 802.16 FDD Frame Structure.

## 2.3 Downlink Subframe

The downlink subframe is as shown in figure 2.6. A downlink subframe starts with a preamble used by the physical layer for synchronization. This is followed by the frame control section which contains Downlink Map (DL-MAP) for the current downlink frame and one Uplink Map (UL-MAP) for uplink channel for a specified time in the future. It may contain Downlink Channel Descriptor (DCD) and Uplink Channel Descriptor (UCD) messages. DCD and UCD messages define the characteristics of downlink and uplink physical channel respectively. DL-MAP message specifies frame duration, frame number, downlink channel ID and time when physical layer transitions (modulation and FEC changes) occur within the downlink subframe. UL-MAP message specifies uplink channel ID, the start time of uplink subframe relative to the start of the frame and bandwidth grants to specific SSs. Uplink bandwidth is allocated to various SSs in terms of mini-slots where each mini-slot is equal to $2^m$ physical slots ($m$ ranges from 0 through 7). Allocation of mini-slots to various SSs for uplink transmission is stated in UL-MAP.

The control section is followed by a TDM portion which carries data, organized into bursts with different burst profiles. Data is transmitted to each SS using a negotiated burst profile in the order of decreasing robustness to allow SSs to receive their data before being presented with a burst profile that could cause them to lose synchronization with the downlink. Each SS receives and decodes the downlink control information and looks for MAC headers indicating data for that SS in the remainder of the downlink subframe. In FDD systems, the TDM portion may be followed by a TDMA segment to better support half-duplex SSs. Many half-duplex SSs may need to transmit earlier in the frame than

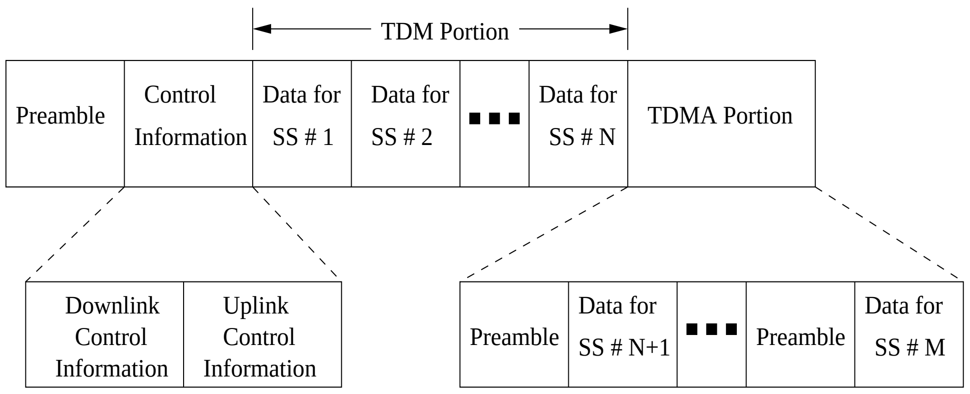

Figure 2.6: The 802.16 Downlink Subframe Structure.

they receive and lose synchronization due to their half-duplex nature. TDMA segment contains an extra preamble at the start of each new burst profile that allow them to regain synchronization. A TDD downlink subframe is same as FDD downlink subframe without a TDMA segment.

## 2.4 Uplink Subframe

The uplink subframe is as shown in figure 2.7. It consists of three classes of bursts which are transmitted by the SS [1]. They are:

- Bursts that are transmitted in contention slots reserved for initial ranging.
- Bursts that are transmitted in contention or unicast slots reserved for requesting bandwidth.
- Bursts that are transmitted in unicast slots specifically allocated to individual SSs for transmitting uplink data.

Any of these burst classes may be present in any uplink subframe. They can occur in any order and any quantity limited by number of time slots allocated for uplink transmission by the BS. The SSs transmit in their specified allocation using the burst profile given in UL-MAP entry. SS Transition Gaps separate the transmissions of the various SSs during the uplink subframe, followed by a preamble allowing the BS to synchronize to the new SS.

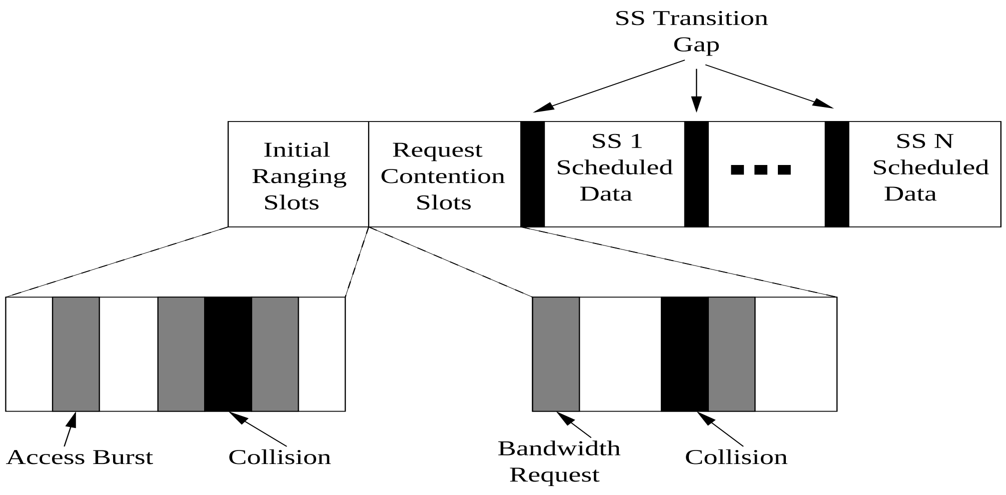

Figure 2.7: The 802.16 Uplink Subframe Structure.

## 2.5 MAC Common Part Sublayer Details

The 802.16 MAC CPS specifies the mechanism to efficiently access the shared medium i.e. the space through which radio waves propagate. On the downlink, the BS is the only central entity transmitting to the SSs. As a result, it does not need to coordinate with other stations. The BS has a sectorized antenna which is capable of transmitting to multiple sectors simultaneously. All SSs in a given frequency channel and sector receive the same transmission. Messages sent by the BS may be unicast to a particular SS, multicast to a group of SSs or broadcast to all SSs. SSs share the uplink channel in DAMA-TDMA fashion.

The IEEE 802.16 MAC is connection-oriented [1]. It maps both connection-oriented and connection-less services to a connection. Connection is a mechanism for requesting bandwidth, associating QoS and traffic parameters, and for various other actions related to data communication. A connection is identified by a 16-bit Connection Identifier (CID). Each SS has a standard 48-bit MAC address which uniquely identifies the SS and is used for registering and authenticating it. For all future operations CID is used as a primary address. On entry to the network, the SS is assigned three management connections in each direction as follows:

1. The first one is *Basic* connection used for the transfer of short time critical MAC messages such as ranging etc.

2. The *Primary* connection is used for the transfer of longer, delay tolerant messages,

such as authentication and connection setup.

3. The *Secondary* connection is used for the transfer of standards-based management messages such as Dynamic Host Configuration Protocol (DHCP), Trivial File Transfer Protocol (TFTP).

SSs are also allocated unidirectional transport connections for their contracted service flows. Service flow defines the QoS parameters for the connection. Connections are also reserved for contention-based initial access, broadcast transmissions in the downlink and broadcast and multicast polling of the SSs bandwidth needs.

### 2.5.1 MAC Protocol Data Unit Formats

The MAC Protocol Data Unit (PDU) is the data unit exchanged between the MAC layers of the BS and its SSs [3]. The CS receives external network Service Data Units (SDUs) through CS Service Access Point (SAP) and associates them to the proper MAC service flow and CID. MAC CPS receives this data from the CS and encloses it in the MAC PDU to send it to its recipient. It consists of a fixed length header, a variable-length payload and an optional Cycle Redundancy Check (CRC). Header can be of two types: the *generic header* and the *bandwidth request header*. The MAC PDU with *generic header* contains MAC management message or CS data in payload field while *bandwidth request header*, used to request bandwidth for the connection, contains no payload.

MAC PDU may contain various types of subheaders as follows.

1. *Grant Management Subheader*: It is used by an SS to request for bandwidth to the BS.

2. *Packing Subheader*: Two or more SDUs may be packed into a single PDU which is indicated by packing subheader.

3. *Fragmentation Subheader*: A SDU may be fragmented into two or more SDUs. The fragmentation subheader is used to indicate the presence and orientation of SDU fragments in payload.

*Fragmentation* is the process in which a MAC SDU is divided into two or more MAC SDUs. *Packing* is the process in which two or more MAC SDUs are packed into a single

MAC PDU. Both can be used simultaneously for efficient use of bandwidth. At the receiving side, inverse of these processes is performed to keep this format modification transparent to the receiving entity. *Concatenation* is the process by which multiple MAC PDUs may be concatenated into a single burst in either the uplink or downlink directions.

### 2.5.2 Network Entry and Initialization of an SS

Various phases of network entry and initialization of an SS are described below [1]:

- *Channel Acquisition*: On initialization, the SS scans the possible channels of downlink frequency band. Once it gets a valid downlink control message on any of the channel, it acquires a downlink channel. After deciding on the downlink channel to use for transmission, SS attempts to synchronize to the downlink transmission by detecting periodic frame preambles. SS then searches for downlink control messages to know downlink control parameters such as downlink channel Id, frame duration etc. SS then waits for UCD message from the BS in order to acquire an uplink channel for transmission. It collects UCD messages with different CID and try each one until it gets a usable channel.

- *Initial Ranging*: Ranging is the process of acquiring the correct timing offset such that the SSs transmissions are aligned to a symbol that marks the beginning of a minislot boundary [3]. After learning downlink and uplink channel parameters, the SS scans UL-MAP message for initial ranging slots. In this slot, it will send an initial ranging request with minimum power setting and will try again with next higher power level until it gets a response from the BS. After receiving ranging request from the SS, the BS commands a timing advance and a power adjustment to the SS in the ranging response. The ranging response also provides *Basic* and *Primary* management CID to the SS. This request/response steps continues until response notifies ranging successful or aborts it.

- *Negotiate Basic Capabilities*: After completion of ranging, the SS reports its PHY capabilities to the BS. These include modulation and coding schemes supported by the SS and whether it supports TDD, half-duplex or full-duplex FDD. The BS responds with a message in which intersection of the SS and BS capabilities are set to ‘ON’.

- *SS Authentication*: The SS then authorizes itself to BS and exchanges key.
- *Registration*: Registration is the process by which the SS receives its secondary management CID. To register with the BS, the SS sends registration request to the BS. The BS then responds with a registration response. The version of IP used on the secondary management connection is also determined during registration.
- *IP Connectivity*: After registration, the SS attains an IP address via DHCP and establishes time of day via Internet Time Protocol. The DHCP server also provides the address of the TFTP server from which the SS downloads a configuration file which provides vendor-specific configuration information.
- *Connection Setup*: Some service flows for an SS are provisioned with BS before initialization of the SS. The BS sends request message to the SS to setup connections for these pre-provisioned service flows belonging to the SS. The SS responds with a successful response message to establish the connection.

Admitted connections are assigned uplink transmission slots for request and data transmission as per their service type. This information is passed to each SS through uplink control messages sent on downlink channel. Each SS determines the start time and duration of slots assigned to its connections. SS then transmits bandwidth requests and uplink data appropriately in the assigned slots. This phenomenon occurs in every frame.

### 2.5.3 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 with 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 average delay, 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 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:

#### 2.5.3.1 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].

#### 2.5.3.2 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.

### 2.5.4 Contention Resolution Algorithm

Collisions may occur during contention slots in which all SS are allowed to transmit ranging or bandwidth request. The process of contention resolution is based on truncated binary exponential backoff, with the initial and maximum backoff window (specified as power of 2) controlled by the BS. When an SS has some bandwidth request to send and wants to enter contention resolution process, it sets its internal backoff window to initial backoff window. It then selects a random number within its backoff window which indicates the number of contention transmission opportunities that the SS defers before transmitting. If data grant is allocated to this SS in a subsequent uplink subframe then the contention resolution is complete. Otherwise the transmission is lost if no grant has been given within a pre-specified time limit. The SS keeps repeating the whole process by increasing its backoff window by a factor of 2 until it is less than the maximum backoff window. This process continues until the maximum number of retries are reached after which appropriate action is taken.

## 2.6 Need for a QoS Scheduling Architecture for IEEE 802.16

As described above, IEEE 802.16 has been developed keeping in view the stringent QoS requirements of various applications. 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. We propose an efficient QoS scheduling architecture for IEEE 802.16 Wireless MANs with a fixed point-to-multipoint topology. The next chapter provides a brief overview of our proposed QoS scheduling architecture.

# Chapter 3

# 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.

## 3.1 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 3.1. 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.

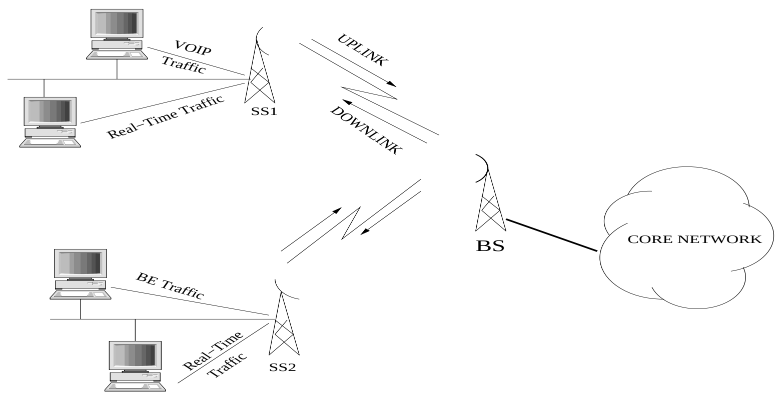

Figure 3.1: Wireless Metropolitan Area Network.

## 3.2 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 weighted max-min fair allocation strategy. High priority flows are assigned more weight to make sure that bandwidth requests for these flows are granted ahead of lower priority flows.

- 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.

## 3.3 Overview of the Architecture

The block diagram of our proposed QoS scheduling architecture is shown in figure 4.1. 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 chapter.

# Chapter 4

# Proposed Architecture Details

Our architecture consists of various components as shown in figure 4.1. It mainly includes downlink and uplink data classifiers, traffic shapers at BS and SS, uplink map generator at BS, downlink and uplink scheduler. Some of the components of our architecture are specified in the standard while others are introduced by us to meet QoS design goals. We now explain the working of each component in detail.

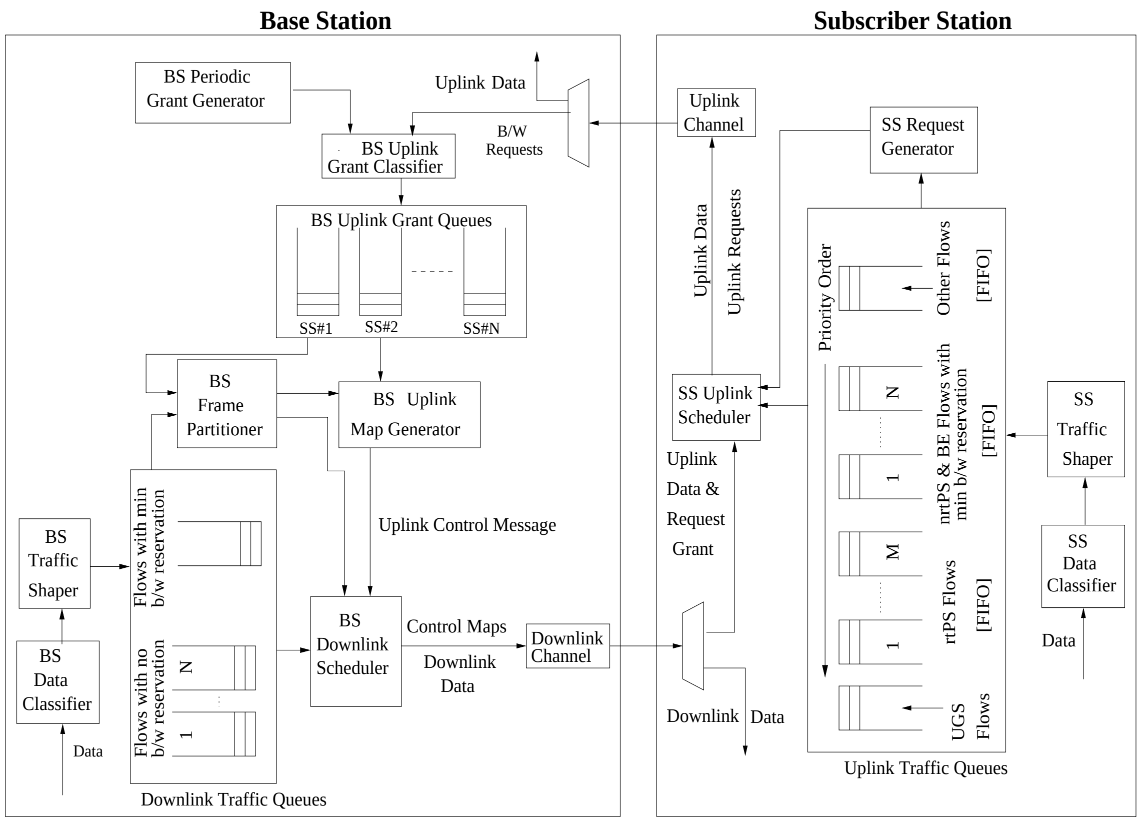

Figure 4.1: QoS Scheduling Architecture for IEEE 802.16.

## 4.1 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 4.1. 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 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.

## 4.2 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.

## 4.3 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.

## 4.4 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 4.1. Classification is done by mapping CID to the corresponding SS.

## 4.5 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.

## 4.6 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*.

## 4.7 SS Traffic Shaper

The working of this component is similar to *BS Traffic Shaper*.

## 4.8 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 i.e.

$$AggregateRequest_i = QueueLength_i$$

## 4.9 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 [4]. 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 for allocating bandwidth to SSs for uplink transmission is described below:

**Algorithm 1** BS Uplink Map Generator

```
1: flow_Req[1] = flow_Req[2] = flow_Req[3] = flow_Req[4] = 0;
2: flow_Alloc[1] = flow_Alloc[2] = flow_Alloc[3] = flow_Alloc[4] = 0;
3: 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.*/
4: totalSS = N; /*Total number of SS is $N$.*/
5: for all $j = 1; j \leq totalSS; j++$ do
6:     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];
7:     flow_Req[1] += CBR_Req[j]; flow_Req[2] += rtPS_Req[j];
       flow_Req[3] += nrtPS_Req[j]; flow_Req[4] += BE_Req[j];
8: end for
```

```
    /*Uplink bandwidth is distributed among all four flows using max-min allocation.*/
 9: while $\sum_{i=1}^{4} flow\_Weight[i]$ != 0 && totalUplinkBytes != 0 do
10:   extraBytes = 0;   totalWeight = $\sum_{i=1}^{4} flow\_Weight[i]$;
11:   for all $i = 1; i \leq 4; i++$ do
12:     flow_Alloc[i] += $\lfloor \frac{flow\_Weight[i]}{totalWeight} \rfloor$ * totalUplinkBytes;
13:     if flow_Weight[i] != 0 && flow_Req[i] $\leq$ flow_Alloc[i] then
14:       extraBytes += flow_Alloc[i] - flow_Req[i];
15:       flow_Alloc[i] = flow_Req[i];
16:       flow_Weight[i] = 0; /*Weight is set to 0 for satisfied flows.*/
17:     end if
18:   end for
19:   $totalUplinkBytes = extraBytes$;
      /*Extra Bytes will be distributed among unsatisfied flows.*/
20: end while
    /*For all uplink flows, 1 = UGS, 2 = rtPS, 3 = nrtPS, 4 = BE.*/
21: for all $i = 1; i \leq 4; i++$ do
22:   totalSS = N; /*Total number of SS is $N$.*/
23:   if flow_Req[i] == flow_Alloc[i] then
24:     for all $j = 1; j \leq totalSS; j++$ do
25:       SSUplinkBytes[j] += flow_Req_SS[i][j]; flow_Req_SS[i][j] = 0;
26:     end for
27:   else
28:     for all $j = 1; j \leq totalSS; j++$ do
29:       satisfied[j] = 0; /*Keeps track of SSs with unsatisfied requests.*/
30:     end for
        /*Bandwidth allocated to $i^{th}$ flow is distributed among all SSs using max-min
        allocation.*/
31:     while flow_Alloc[i] > 0 do
32:       extraBytes = 0; flow_Alloc_SS = $\lfloor \frac{flow\_Alloc[i]}{totalSS} \rfloor$;
33:       for all $j = 1; j \leq N; j++$ do
34:         if satisfied[j] == 0 then
35:           if flow_Alloc_SS $\geq$ flow_Req_SS[i][j] then
```

```
36:                 SSUplinkBytes[j] += flow_Req_SS[i][j];
37:                 extraBytes += flow_Alloc_SS - flow_Req_SS[i][j];
38:                 flow_Req_SS[i][j] = 0; satisfied[j] = 1; totalSS -= 1;
39:               else
40:                 SSUplinkBytes[j] += flow_Alloc_SS;
41:                 flow_Req_SS[i][j] -= flow_Alloc_SS
42:               end if
43:             end if
44:           end for
45:           flow_Alloc[i] = extraBytes; /*Extra Bytes will be distributed equally among
              unsatisfied SSs.*/
46:       end while
47:     end if
48: end for
49: for all $j = 1; j \leq N; j++$ do
50:   SSUplinkBytes[j] += $\frac{totalUplinkBytes}{N}$;
51: end for
```

## 4.10 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 [5]. 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.

## 4.11 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.

# Chapter 5

# Implementation Details

In this chapter we discuss the implementation details of IEEE 802.16 MAC and our architecture. 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.

We have abstracted the relevant MAC details from the standard for the purpose of simulation. IEEE 802.16 MAC features supported by our implementation are as follows.

1. TDD frame structure.
2. All four uplink scheduling services.
3. GPSS mode for bandwidth allocation.
4. *Aggregate* bandwidth requests.
5. *Concatenation* of MAC packets into a single transmission burst.
6. Provides interface for IP layer.
7. Static service flows.

Qualnet 3.6 is a discrete event simulator. Protocols operate as a finite state machine that changes state on the occurrence of an event. Every protocol consists of three main functions. Every protocol begins with an initialization function which reads external input to configure the state of the protocol. The control is then passed to an event dispatcher.

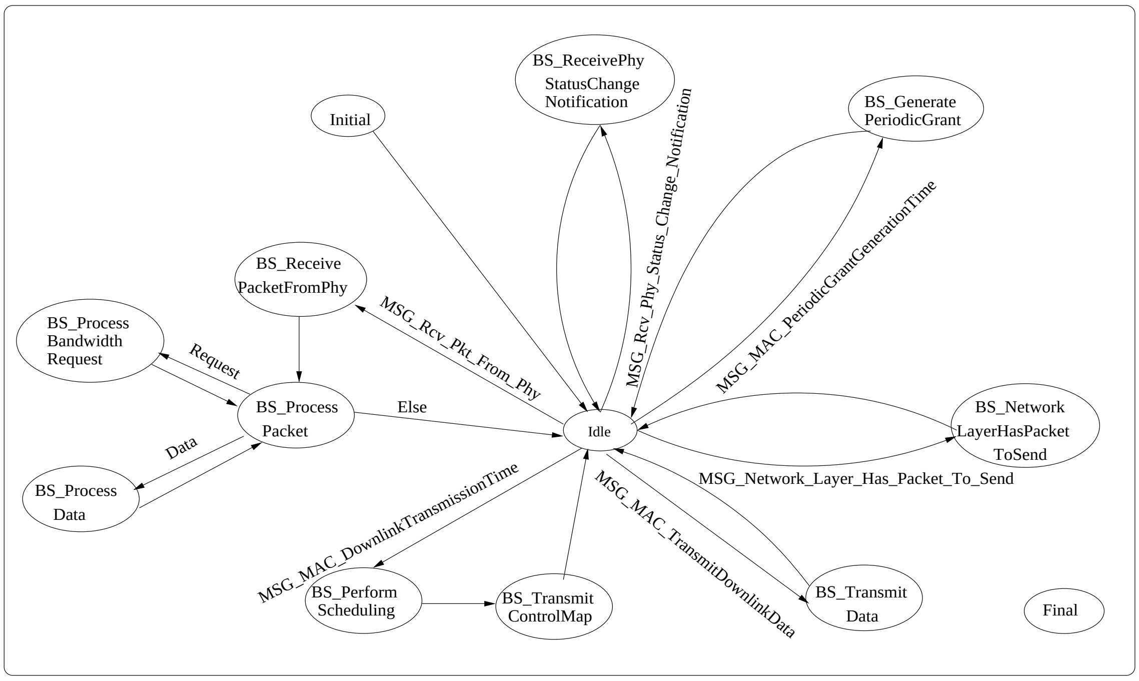
Initial
BS_ReceivePhy StatusChange Notification
BS_Generate PeriodicGrant
BS_Receive PacketFromPhy
BS_Process Bandwidth Request
BS_Process Packet
BS_Process Data
Idle
BS_Network LayerHasPacket ToSend
BS_Perform Scheduling
BS_Transmit ControlMap
BS_Transmit Data
Final
Request
Data
Else
MSG_Rcv_Pkt_From_Phy
MSG_Rcv_Phy_Status_Change_Notification
MSG_MAC_PeriodicGrantGenerationTime
MSG_Network_Layer_Has_Packet_To_Send
MSG_MAC_DownlinkTransmissionTime
MSG_MAC_TransmitDownlinkData

Figure 5.1: BS State Transition Diagram.

On arrival of an event, it determines to which protocol it should be directed and calls the event dispatcher for that protocol. The event handler is called as per the type of the event to process it. Finally, at the end of the simulation, finalization function is called to print the statistics for every protocol at each node. For the implementation purpose, we have drawn state transition diagrams of BS and SS as shown in figure 5.1 and figure 5.2.

BS and all SSs are initialized along with other MAC parameters such as downlink and uplink channel ID etc. Various uplink and downlink connections are setup and are assigned a unique connection identifier. After initialization, BS and SS waits in idle state for an event to take place. In the idle state, MAC layer may receive messages from network layer for sending a packet, from physical layer for receiving a packet or from itself on expiration of MAC timers.

On receiving a *MSG_Network_Layer_Has_Packet_To_Send* message from network layer for sending an IP packet, BS adds it to one of the Downlink Traffic Queues after classification and shaping. MAC header is also added to this packet. If the IP packet is fed to one of the Type 1 queues then WFQ virtual time is updated based on the weight of currently active queues. A queue is active if it has at least one packet. WFQ finish time for the current packet is also calculated based on current WFQ virtual time or WFQ finish time of the last packet in the queue, whichever is greater.

*MSG_Rcv_Pkt_From_Phy* is sent by physical layer to notify the MAC for receiving a packet sent by an SS. Uplink data packets handed to BS MAC layer are sent to higher layer after removing MAC header. Uplink bandwidth request packets are classified and placed in uplink grant queues. Change in status of physical channel is notified to MAC through *MSG_Rcv_Phy_Status_Change_Notification* message.

In order to generate periodic data and request grants, BS schedules a timer message *MSG_MAC_PeriodicGrantGenerationTime*. On expiration of the timer, particular grant is generated and timer for next grant is scheduled.

A timer message *MSG_MAC_DownlinkTransmissionTime* is scheduled for the start of downlink subframe. On expiration of the timer, control messages are created by BS. BS Uplink Map Generator allocates bandwidth to each SS for uplink transmission in the current frame using algorithm 1. Uplink bandwidth allocation is encoded in uplink control message and passed on to BS Downlink Scheduler. BS Downlink Scheduler generates downlink control message based on the current state of Downlink Traffic Queues using BS downlink scheduling strategy. WFQ virtual time is also updated at the time of departure of a packet from Type 1 queues. If a Type 1 queue becomes empty then it is marked inactive. BS transmits downlink and uplink control messages on the downlink channel at their scheduled time. A timer message *MSG_MAC_TransmitDownlinkData* is then scheduled for transmission of downlink data. Downlink data is transmitted upon expiry of the above mentioned timer. BS then schedules a timer for transmitting next downlink subframe.

When SS receives *MSG_Network_Layer_Has_Packet_To_Send* message from network layer for sending an IP packet, SS adds it to one of the Uplink Traffic Queues after classification and shaping. MAC header is also added to this packet. If the IP packet is fed to one of the Type 2 or Type 3 queues then WFQ virtual time is updated based on the weight of currently active queues. WFQ finish time for the current packet is also calculated based on the current WFQ virtual time or WFQ finish time of the last packet in the queue, whichever is greater.

For the downlink subframe duration, SS PHY continuously listens to downlink channel to discover if there are any downlink packets intended for it. *MSG_Rcv_Pkt_From_Phy* is sent by physical layer to notify the MAC for receiving a packet sent by the BS. Downlink

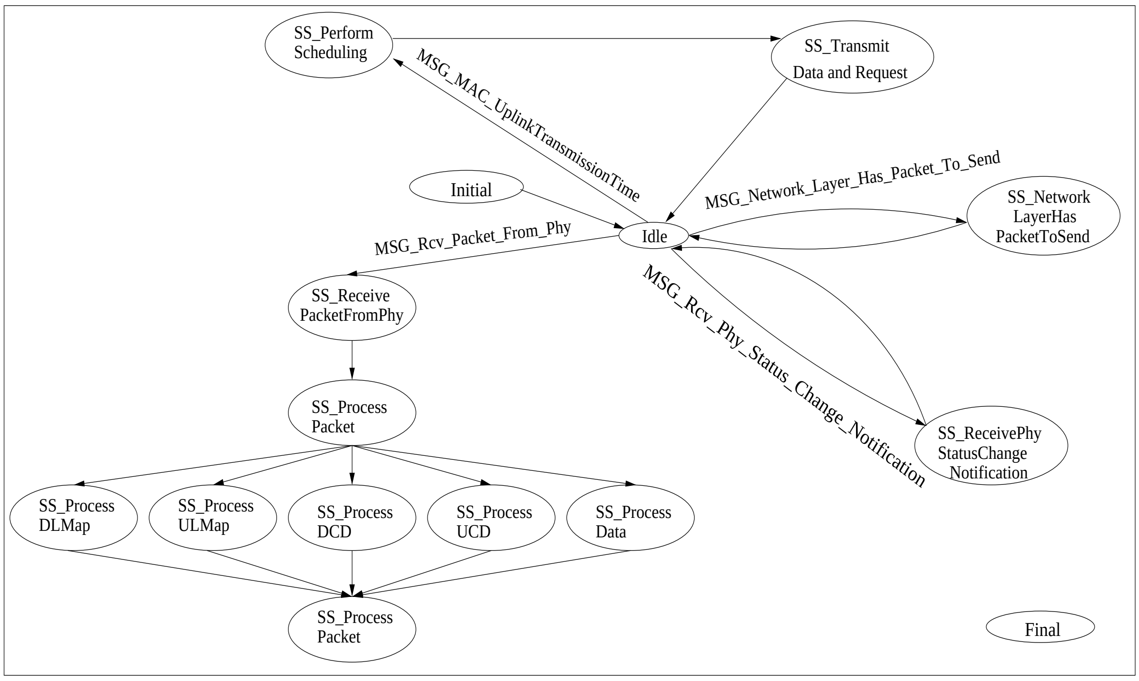
SS_Perform Scheduling
SS_Transmit Data and Request
MSG_MAC_UplinkTransmissionTime
Initial
Idle
MSG_Network_Layer_Has_Packet_To_Send
SS_Network LayerHas PacketToSend
MSG_Rcv_Packet_From_Phy
SS_Receive PacketFromPhy
MSG_Rcv_Phy_Status_Change_Notification
SS_Process Packet
SS_ReceivePhy StatusChange Notification
SS_Process DLMap
SS_Process ULMap
SS_Process DCD
SS_Process UCD
SS_Process Data
SS_Process Packet
Final

Figure 5.2: SS State Transition Diagram.

packets addressed to the SS are received and processed based on their type. Downlink and uplink control messages sent by BS on downlink channel are used for determining various control parameters for both downlink and uplink channels. Downlink control message is decoded to determine start time of the frame, frame duration etc. SS determines its uplink transmission time and duration of transmission in the current frame by decoding uplink control message. A timer message *MSG_MAC_UplinkTransmissionTime* is scheduled which notifies the SS the uplink transmission time on expiry. Downlink data packets are handed over to higher layer after removing MAC header.

At the start of its uplink transmission slot, SS Uplink Scheduler schedules data packets and request packets to be transmitted in the current uplink subframe using SS uplink scheduling strategy. Bandwidth request packets are generated by SS Request Generator. WFQ virtual time is updated at the time of departure of a packet from Type 2 and Type 3 queues. If a Type 2 and Type 3 queue becomes empty then it is marked inactive. Scheduled packets are transmitted by SS in the assigned slot. After completing transmission, SS PHY again listens to downlink channel.

## 5.1 Integration with Qualnet 3.6 Simulator

Qualnet 3.6 is a very fast, scalable and efficient simulator. Its source code is highly modular and very well documented. So, it was not very hard for us to implement and integrate IEEE 802.16 MAC and our architecture with Qualnet 3.6. In this section, we present the major changes made by us for integrating our code with Qualnet 3.6.

We have written 802.16 MAC protocol in files *mac_802_16.h* and *mac_802_16.c*. These two files are then placed in *mac* folder in qualnet directory. To compile these files along with existing files, the file names are added to *Makefile-common* file in *main* folder in qualnet directory. File *mac.h* includes the name of existing MAC protocols. So, we made an entry of 802.16 MAC protocol in this file. Then the initialization function, event dispatcher and finalization function name for 802.16 MAC protocol are added to *mac.c* file. The various timer messages defined above are added to the file *api.h*.

In Qualnet 3.6, a message structure is used to define an event. This message structure carries information about the event such as its type and, the associated data too. We have added some more fields to this structure to carry extra information for implementing *Concatenation* of packets for IEEE 802.16 MAC.

# Chapter 6

# Simulation Analysis

So far we have gone through the details of or architecture and its implementation. In this chapter we present our simulation setup and the results which show that our architecture fulfills the stated design goals.

## 6.1 Simulation Setup

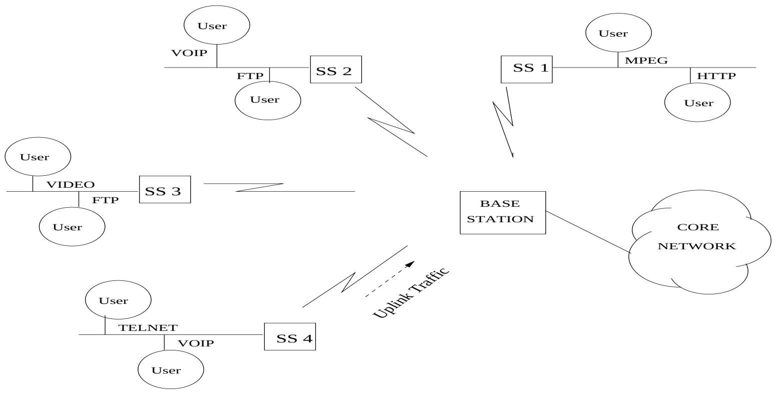

Figure 6.1: Simulation Setup.

The simulation topology consists of one BS and a number of SSs as shown in figure 6.1. 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.

## 6.2 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 6.2 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.

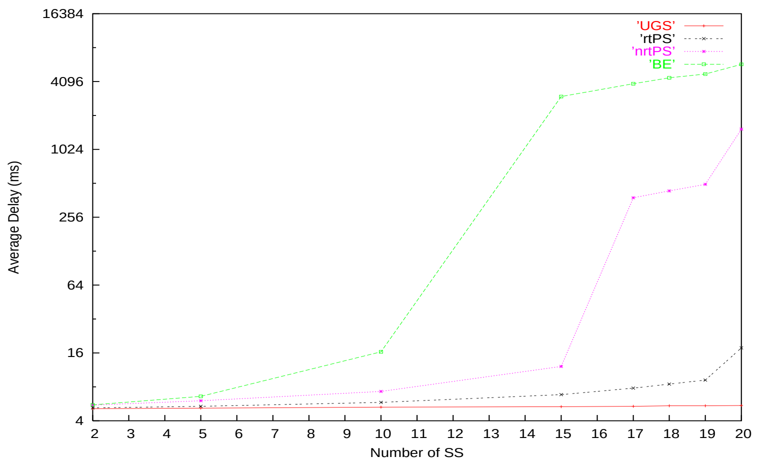

Figure 6.2: Average Delay Vs Number of SS.

- 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

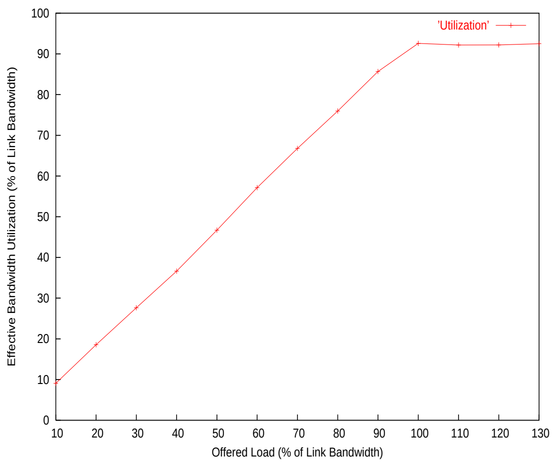

Figure 6.3: Effective Bandwidth Utilization Vs Offered Load[Scenario 1].

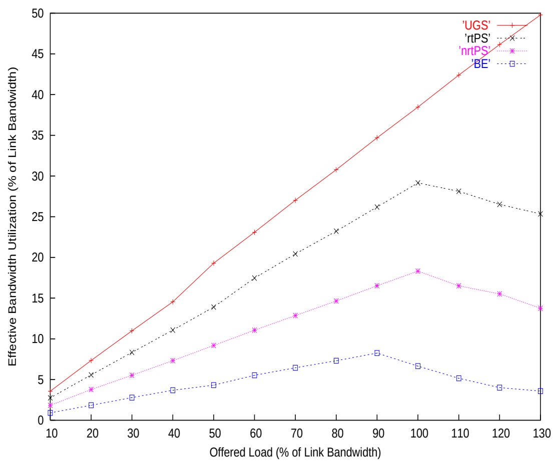

Figure 6.4: Effective Bandwidth Utilization Vs Offered Load[Scenario 1].

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. We have calculated effective bandwidth utilization in two different scenarios as follows:

– *Scenario 1*: In this scenario, the load offered by high priority traffic is more as compared to lower priority traffic. Figure 6.3 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 effective bandwidth utilized for each of the four uplink flows as shown in Figure 6.4. 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%.

– *Scenario 2*: In this scenario, the load offered by low priority traffic is more as compared to higher priority traffic. Figure 6.5 shows the graph for effective bandwidth utilization with increasing offered load for scenario 2. Figure 6.6

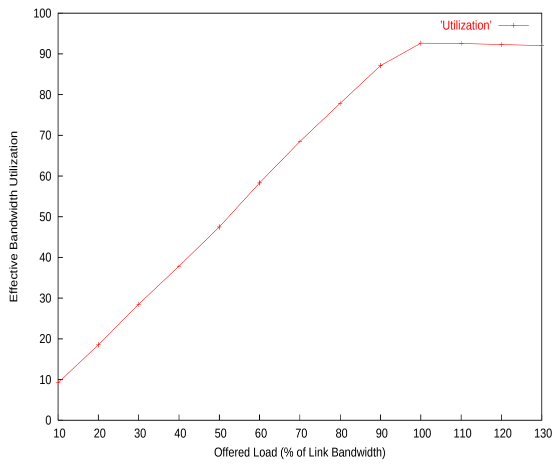

Figure 6.5: Effective Bandwidth Utilization Vs Offered Load[Scenario 2].

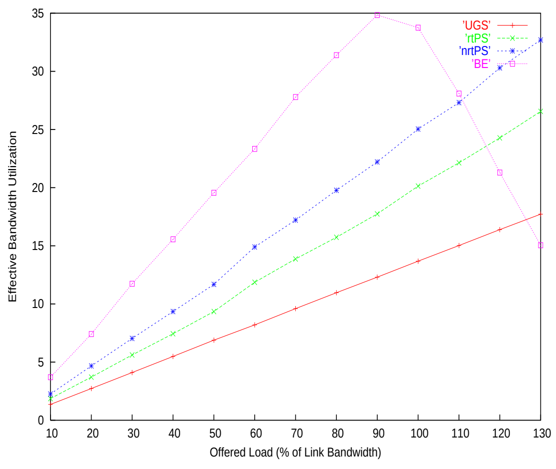

Figure 6.6: Effective Bandwidth Utilization Vs Offered Load[Scenario 2].

shows effective bandwidth utilization for each of the four uplink flows. It is visible from the graphs that same behavior is exhibited in both the scenarios. 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 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 6.7 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

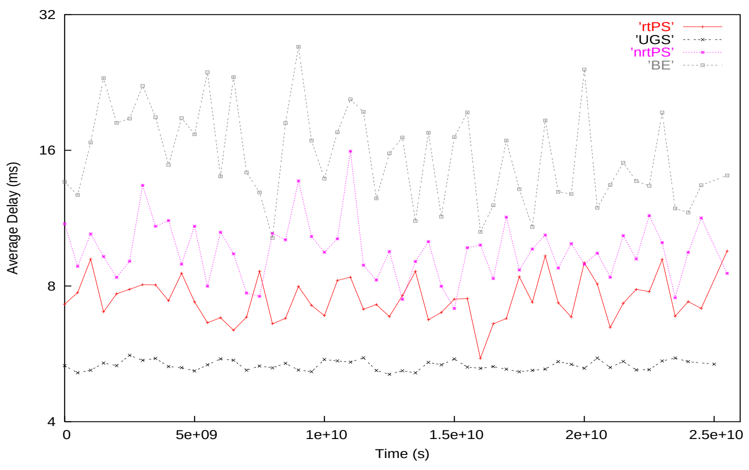

Figure 6.7: Average Delay Vs Time[Scenario 1].

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.

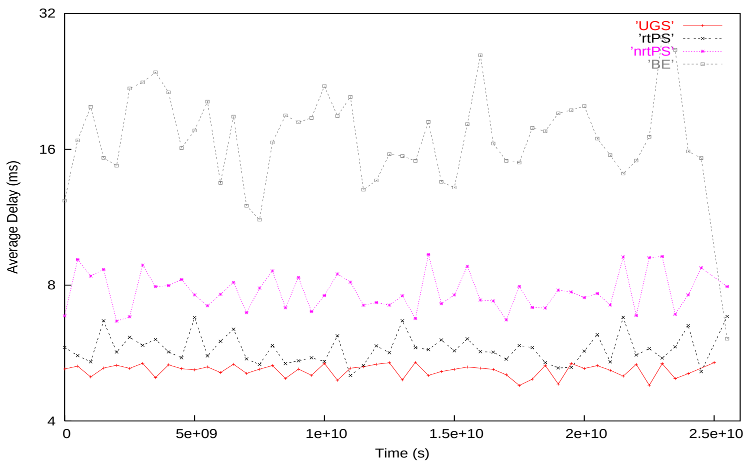

Figure 6.8: Average Delay Vs Time[Scenario 2].

– *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 run for 25 seconds. We have measured average delay experienced by each uplink flow for each SS. Figure 6.8 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.

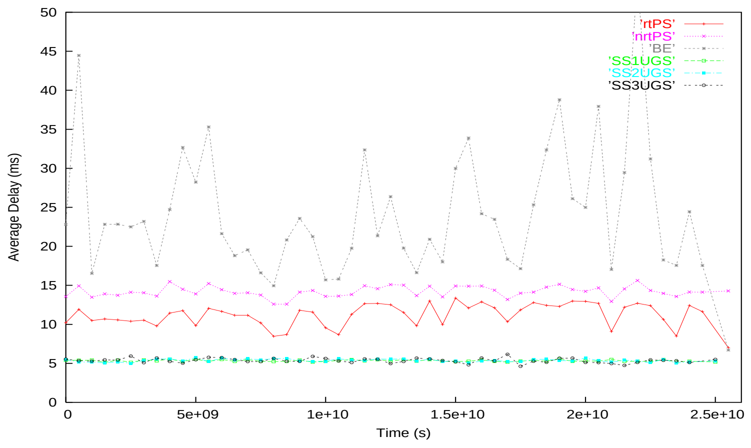

Figure 6.9: Average Delay Vs Time[Scenario 3].

– *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 flows offer minimum load. Figure 6.9 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 priority flows are always satisfied.

Results obtained in the above three scenarios show that our architecture is capable of providing tight delay guarantees to various real-time applications.

## 6.3 Additional Experiments

This section provides the graphs depicting the results of some additional experiments conducted to observe the performance of our architecture.

- We have not implemented fragmentation of MAC packets. In this experiment, we have calculated total bytes granted to each SS and total bytes utilized by them to observe the effect of not allowing packet fragmentation. A simulation topology with one BS and five SS has been setup. We can see in Figure 6.10 that each SS has utilized around 95% of the total bytes granted to it. In the absence of fragmentation, an SS will not be able to transmit a packet if bandwidth required for transmitting it completely is not available.

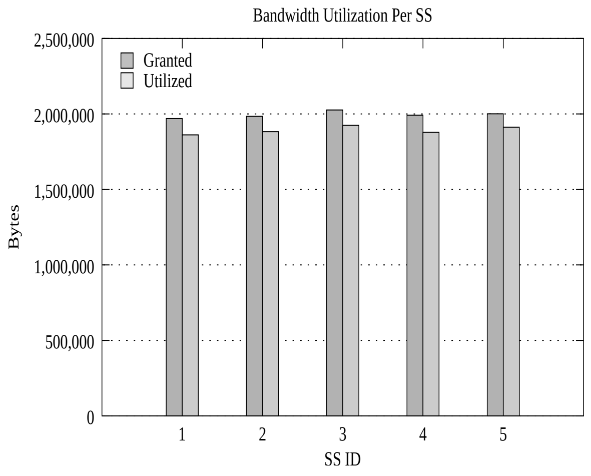

Figure 6.10: Bandwidth Utilization Per SS.

- In this experiment, we have measured effective bandwidth utilization with increasing number of SSs. In every simulation run, each SS has same number of uplink flows with similar QoS parameters. Total load offered by each SS is approximately 8%. Each simulation is run for 25 seconds. We have calculated effective bandwidth utilization with increasing SSs in two different scenarios as follows:
  - *Scenario 1*: In this scenario, the load offered by UGS and rtPS flows is more than the load offered by nrtPS and BE flows. Figure 6.11 shows the graph for effective bandwidth utilization with increasing number of SSs. It is observed that maximum effective bandwidth utilization achieved with increasing

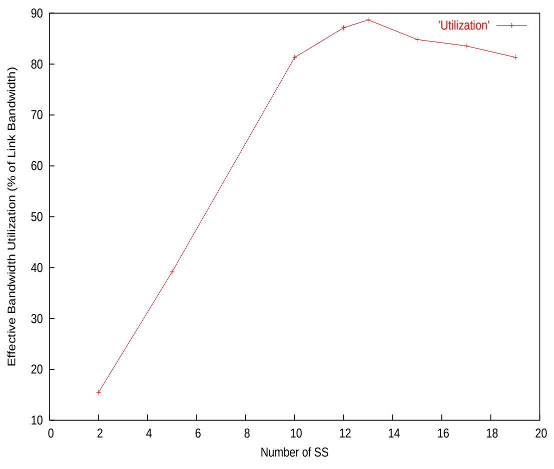

Figure 6.11: Effective Bandwidth Utilization Vs Number of SS[Scenario 1].

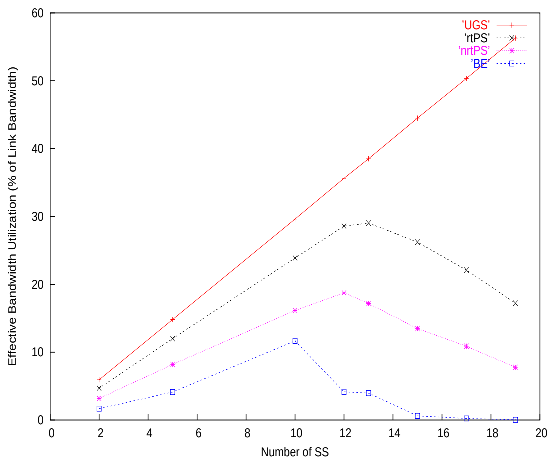

Figure 6.12: Effective Bandwidth Utilization Vs Number of SS[Scenario 1].

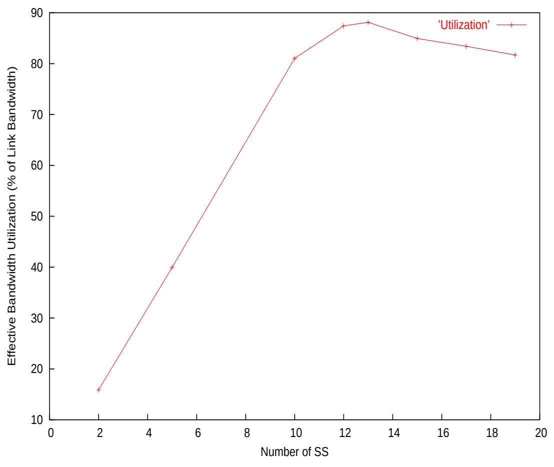

Figure 6.13: Effective Bandwidth Utilization Vs Number of SS[Scenario 2].

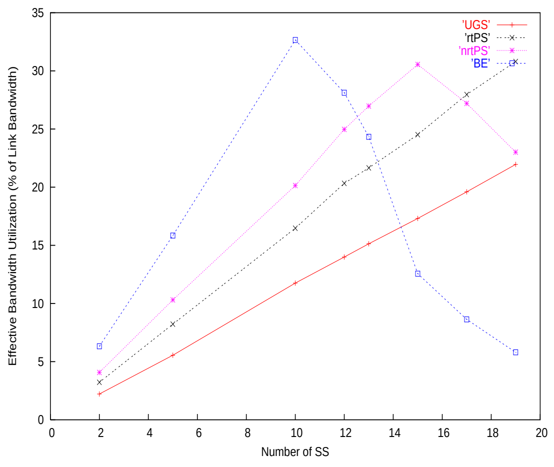

Figure 6.14: Effective Bandwidth Utilization Vs Number of SS[Scenario 2].

number of SSs is around 88%. It is less due to the reason that more number of control packets (Bandwidth Request Packets) are sent as the number of SSs in the system are increased. Also with increasing number of SSs, more bandwidth gets wasted because packet fragmentation is not allowed. We have also calculated effective bandwidth utilization for each of the four uplink flows as shown in Figure 6.12. As the number of SS increases, total load offered to the system also increases. Initially, bandwidth utilization of all uplink flows is in proportion to the total load offered by them. However, the bandwidth utilization of lower priority nrtPS and BE flows starts decreasing when total load offered to the system becomes grater than 90%.

– *Scenario 2*: In this scenario, the load offered by low priority traffic is more as compared to higher priority traffic. Figure 6.13 shows the graph for effective bandwidth utilization with increasing offered load for scenario 2. Figure 6.14 shows effective bandwidth utilization for each of the four uplink flows. It is observed that regardless of the amount of load offered by any uplink flow, higher priority flows are always provided more bandwidth in order to provide promised QoS for them.

# Chapter 7

# Related ork

IEEE 802.16 MAC has not been widely studied and there are very few proposed architectures in the literature. In this chapter we present survey of some of the work done in 802.16 domain.

## 7.1 A QoS Architecture for the MAC Protocol of IEEE 802.16 BWA System

In [6], 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.

At the BS, based on the bandwidth requests from all SS, uplink bandwidth is dynamically distributed between contention and reservation slots. BS upstream grant scheduler computes the bandwidth allocation and time of transmission for each SS based on bandwidth requests using WRR algorithm [7]. A traffic policing function is also employed at the BS to ensure that SSs connection conforms to the negotiated traffic parameters.

At the SS, for each uplink service type, multiple connections are aggregated into their respective service flow and assigned a priority. A different scheduling algorithm is used for each of the priority queues with packet being transmitted from the highest priority class that has a packet available.

However in [6], working of BS's upstream scheduler is not specified in sufficient detail i.e. 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. In addition to this, no simulation results have been presented in order to show the goodness of the architecture. Downlink scheduling mechanisms at BS are not discussed either.

## 7.2 QoS Scheduling in Cable and BWA Systems

In [8], M. Hawa et al have proposed an uplink scheduling architecture to support bandwidth and delay QoS for both DOCSIS [9] and IEEE 802.16. They have chosen GPC mode for bandwidth grants. It is a centralized approach wherein all the scheduling decisions are taken at BS.

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 priority-enhanced WFQ i.e. if two data grants have equal WFQ virtual finish time, then the higher priority grant is served first. WFQ weight to reserved flows is assigned based on their minimum bandwidth reservations while unreserved flows are given weight based on the unreserved bandwidth.

Authors have also provided an algorithm for allocating appropriate number of contention request slots in each frame period so as to reduce the number of possible collisions and to shorten the contention resolution process. Buffer management problem for various types of queues is also dealt in the paper.

However in [8], 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 to show efficiency and performance of the architecture. Downlink scheduling mechanisms at BS are not discussed either.

# Chapter 8

# Conclusion and Future ork

In this report 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.

## 8.1 Future Work

In IEEE 802.16, nrtPS and BE flows also use contention mini-slots for sending bandwidth requests to the BS. An appropriate contention mini-slot allocation algorithm can be incorporated in our current design. The ratio of unicast mini-slots to contention mini-slots in an uplink subframe should be such that sufficient amount of uplink data can be transmitted as well as collision of bandwidth requests is as low as possible.

Current design uses a fixed partition scheme which divides total frame bandwidth equally between downlink and uplink subframe. This scheme can be modified to dynamically allocate bandwidth for downlink and uplink transmissions based on the status of Downlink Traffic Queues and Uplink Grant Queues.

Fragmentation, Packing, request piggybacking and other MAC features can be added to the current implementation of IEEE 802.16 MAC in Qualnet 3.6.

Admission control mechanisms are also not specified in IEEE 802.16 standard. Therefore, an efficient admission control mechanism for IEEE 802.16 MAC can be devised and integrated with our QoS scheduling architecture. A combined performance study of the whole system can then be carried out.

Some research is currently in progress for the design of a wireless communication system using unlicensed frequencies for Indian rural areas. It has been observed that IEEE 802.11b physical layer is suitable for such areas. Further, IEEE 802.11 MAC has been analyzed and proved to be inefficient for a distribution service that needs to maximize capacity for subscribers and maintain QoS. Therefore, a new MAC has been proposed for such scenarios while retaining IEEE 802.11b PHY. The new MAC is very much similar to IEEE 802.16 MAC. Hence, performance of IEEE 802.16 MAC over IEEE 802.11b PHY can be analyzed in order to predict its effectiveness for such a system. This study will be very useful for real deployment of a wireless communication system for Indian rural areas.

# Bibliography

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

[3] 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*, 40(6):98–107, June 2002.

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

[5] Abhay K. Parekh and Robert G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Trans. Netw.*, 2(2):137–150, 1994.

[6] GuoSong Chu, Deng Wang, and Shunliang 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*, 1:435–439, June 2002.

[7] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, 1993.

[8] Mohammed Hawa and David W. Petr. Quality of Service Scheduling in Cable and Broadband Wireless Access Systems. *Tenth IEEE International Workshop on Quality of Service*, pages 247–255, May 2002.

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

[10] Alejandro Quintero, Yacine Elalamy, and Samuel Pierre. Performance evaluation of a broadband wireless access system subjected to heavy load. *Computer Communications*, 27(9):781–791, June 2004.

[11] Yaxin Cao and Victor O. K. Li. Scheduling Algorithms in Broad-Band Wireless Networks. In *Proceedings of the IEEE*, volume 89, pages 76–87, January 2001.

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

[13] Songwu Lu, Vaduvur Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. *IEEE/ACM Transactions on Networking*, 7(4):473–489, 1999.

[14] S. Golestani. A self-clocked fair queuing scheme for broadband applications. In *Proc. IEEE INFOCOM '94*, pages 636–646, 1994.

# Acknowledgements

I take this opportunity to express my sincere gratitude and appreciation for my adviser **Prof. Sridhar Iyer** for his constant support and encouragement which has made this project work a success. **Prof Krishna Paul** spent lot of time with me in the initial stages of this project, I am grateful to her for setting me on the right course in this venture.

I would like to thank Anunay Tiwari, Arun Kumar S. Jadhav and Shailesh M. Birari for their valuable suggestions and helpful discussions.

I would also like to thank my family and friends for their encouragement and showing belief in me.

Finally, I would like to thank the entire KReSIT family for making my stay at IIT Bombay a memorable one.