CHAPTER 3

WiMAX SCHEDULING ARCHITECTURE

3.1 INTRODUCTION

The MAC layer is responsible for the usage of the air resources in an efficient manner and provides QoS differentiation. In general, the MAC layer is responsible for overall connection and session processing. The BS and SS MAC layers communicate with each other to set up, add, and delete services on a need basis. The IEEE 802.16 MAC protocol is designed to support two network communication models.

1. Point to Multipoint mode
2. Mesh mode

3.2 POINT TO MULTIPOINT

In the PMP mode, the nodes are organized into a cellular structure, consisting of a BS and some SSs. As shown in Figure 3.1, the communication link channel is classified into uplink (from SS to BS) and downlink (from BS to SS) channels. But all SSs should be within the transmission range. The IEEE 802.16 MAC protocol is connection oriented. In each time frame, the BS schedules the transmissions and indicates the uplink and downlink BW allocation in a map along with the physical layer parameters. When communication starts between the SS and BS, each SS has to create
connections over which data has to be transmitted to and from the BS. Here, scheduling plays a key role. Now the MAC layer has to schedule the usage of a scarce BW resource, and maintain QoS between the service classes.

![IEEE 802.16 connection oriented architecture](image)

**Figure 3.1** IEEE 802.16 connection oriented architecture

### 3.3 MESH NETWORK

In the case of a mesh mode, the nodes are organized in a random manner. If all the stations are close to one another, each node can act like a router to relay packets to its neighbors. There is no direct link between the SS and the BS in the mesh network. A node can choose the best quality channel to transmit the data with the help of the routing protocol; the traffic can be routed to the best possible route to avoid congestion. Only in this research work the PMP mode is considered.

In the PMP mode of communication, the uplink and downlink communication occurs in different time frames. In the case of the downlink sub frame, the BS transmits a burst of MAC PDUs. The transmission type is
broadcast, all the SSs have to listen to the data transmitted by the BS. But the SS has to process the PDUs which are addressed to it. In the case of an uplink sub frame, any SS can transmit a burst of MAC PDUs to the BS in a Time Division Multiple Access (TDMA) manner. The SS can be configured to be either in full duplex or half duplex mode.

If the SS wants to open a connection to the BS, it first sends a request. Upon receiving the message the BS performs admission control based on the requested traffic, QoS specification, and available resources. Once the connection is established, the SS may obtain the particular BW by sending a class specific request. The BS then aggregates all the requests and allocates the BW to each connection or SS, through an appropriate scheduling scheme.

In order to ensure slotted channel sharing the slots are allocated by the BS to various SSs in one uplink frame using TDD or FDD. This slot allocation information is broadcast by the BS through the UL-MAP at the beginning of each frame. The UL-MAP contains an information element, which includes the transmission opportunities and the time slots in which the SS can transmit during the uplink sub frame.

3.4 WiMAX PROTOCOL ARCHITECTURE

The connection oriented nature of MAC protocol makes the data
in Figure 3.2. The IP, Ethernet, and Asynchronous Transfer Mode (ATM) traffic are supported by the convergence sublayer. This layer converts the traffic into MAC data units.

![IEEE 802.16-2005 protocol stack](image)

**Figure 3.2 IEEE 802.16-2005 protocol stack**

The WiMAX network provides broadband access for services having diversified QoS requirements, and different traffic priorities. It is the responsibility of the MAC layer to schedule the traffic flows and to allocate the BW in such a manner that the QoS requirements of each flow are satisfied.

### 3.5 IEEE 802.16 QoS ARCHITECTURE

The IEEE 802.16 is expected to provide the QoS for fixed and mobile users. The QoS depends upon a number of implementation details like
scheduling, buffer management and traffic shaping. The responsibility of scheduling and BW management is to allocate the resources efficiently, based on the QoS requirement of the service classes. Each service flow is associated with the QoS parameters, which define the scheduling and transmission ordering in the air interface.

As mentioned earlier, the IEEE 802.16 standard has defined five service flow classes, which have different QoS requirements: Unsolicited Grant Service, real time Polling Service, non-real time Polling Service, enhanced real time Polling Service, and BE.

![Figure 3.3 QoS architecture of the IEEE 802.16 standard](image)

Each of the service classes is characterized by stringent QoS requirements, which are tailored to give the optimum guarantee, required by
the applications that the scheduling service is designed for. Figure 3.3 shows
the QoS architecture of the IEEE 802.16 WiMAX standard.

Admission control, traffic policing and scheduling schemes for nrtPS, rtPS and BE connection are undefined by the IEEE 802.16 standard. Uplink BW allocation scheduling resides in the BS to control all the uplink packet transmissions. The IEEE 802.16 MAC layer adopts a connection oriented architecture, in which a connection must be established before the data communication. Each connection is assigned a unique Connection IDentifier (CID) and it is associated with a service flow which defines the desired QoS level of the connection.

In a standard scheduling framework, the data packets arriving at the BS are classified into connections, which are then classified into service flows. Packets of the same service flow are placed in a queue, and then further classified based on their service priorities of the connection. For packets in multiple queues with different service requirements, a packet scheduler is employed to decide the service order of the packets from the queues. If properly designed a scheduling algorithm may provide the desired service guarantees. Each connection requests the BS for the desired BW. The BW may be allocated as per the connection or as per the SS. Actually, IEEE 802.16 standard defines the following:

1. Information exchange between the BS and SS, like connection set up, BW request and UL-MAP can be initialized by the signaling mechanisms.

2. The UGS service flow’s uplink scheduling algorithm.

QoS requirements for the IEEE 802.16 WiMAX standard are summarized in Table 3.1. The UGS is designed to support real time
applications with strict delay requirements, which generated fixed size packets at periodic intervals. UGS connections use unsolicited granting BW request mechanisms. Thus, UGS connections will never request BW. They are given periodic bandwidths without any polling or contention. The grant size is computed by the BS, based on the minimum reserved traffic rate, which is defined as the minimum amount of data transmitted on the connection on an average over time. If additional BW is required, the SS requests the BS to poll it to the allocated BW.

**Table 3.1 QoS Requirements and characteristics of the service classes**

<table>
<thead>
<tr>
<th>Service classes</th>
<th>Min BW</th>
<th>Max BW</th>
<th>Latency</th>
<th>Jitter</th>
<th>BW requests</th>
<th>Contention</th>
<th>Purpose</th>
<th>Polling Interval</th>
<th>Grant Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>UGS</td>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>VoIP</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>rtPS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>VoIP</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>rtPS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>Video</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nrtPS</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
<td></td>
<td>Web/FTP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BE</td>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The rtPS service is designed to support real time applications, which impose less stringent delay requirements. The rtPS service generates variable size data packets at regular intervals. MPEG video and VoIP are a few examples of rtPS services. There are two important parameters i.e. minimum reserved traffic rate, and maximum latency, for rtPS connections which define their QoS. The BS periodically grants unicast polls to rtPS
connections. The received packets have varied sizes and the rtPS service class is specially designed for applications which do not have any specific delay requirements.

The nrtPS service class is designed to support BW intensive applications like FTP. These connections reserve the minimum amount of BW, which will boost up the performance of the application. Both nrtPS and BE connections request their BW, by either responding to broadcast polls from the BS or by piggybacking a BW request on an outgoing PDU.

3.6 WIRELESS SCHEDULING ISSUES

The nature of a WiMAX multiservice environment is such that it is difficult to operate it in the following scenarios.

1. Various packet streams
2. Different QoS for different service classes
3. Varied traffic behaviors.

The packet schedulers at the MAC layers play a crucial role to maintain QoS delivery. Since various service classes impose different QoS requirements, a single scheduling algorithm may not be sufficient for each service class’s distinct QoS characteristic.

Since the scheduling for rtPS, nrtPS, BE is not defined in the IEEE 802.16 standard, numerous approaches have been addressing these issues. Some algorithms work, based on priority based on centralized scheduling. Most of the algorithms work by applying different algorithms for different service classes. A few of the past works consider scheduling algorithms for wired networks, and modify them to fit into wireless networks. But the nature
of a wireless channel imposes limitations due to channel interference and fading effect.

An ideal scheduler should consider the following important parameters.

1. The traffic service type.
2. The set of QoS requirements of the connections defined by the IEEE standard.
3. The capacity of the BW for data transmission.
4. The BW requirements from the connections.
5. Waiting time of the BW request in the system.

The ideal scheduler should be able to make the optimum use of the available BW, to reduce traffic delays and satisfy the QoS requirements to the best extent so as to reduce the packets’ drop rate and sustain the QoS support. The cross layer scheduling is one of the channel aware scheduling algorithms.

3.7 CROSS LAYER INTERACTION CONTRIBUTION IN WiMAX SCHEDULING ALGORITHM

The proposed cross layer scheduling algorithm is split into two parts, namely the BS part and the SS parts which are in BS and SS respectively and are shown in the following Figure 3.4. The BS part accepts (called layer abstraction) layer specific parameters, regarding channel conditions and QoS parameters of active connections, which are given by the BS physical and MAC layers. Depending upon this information the cross layer decision algorithm determines the most suitable modulation and coding scheme, and the traffic rate of each and every SS for both uplink and
downlink. This process is called layer optimization which is shown in Figure 3.4.

![Cross-layer functionality diagram](image)

**Figure 3.4 Cross-layer functionality**

The BS part informs the layers of any modifications, if required. This process is called layer reconfiguration. If the BS part’s decision involves traffic rate changes, it communicates with the SS part through the MAC layer of the SS which then communicates with the SS application layer. The SS only acts as a passive module because only the BS instructs the SS about the processing.

The main functionality of the cross layer scheduler is located at the BS. It has to collect all the information about the traffic, QoS requirements of active service classes, channel conditions, proper adjustments in modulation and coding scheme and traffic rate at each SS for the system’s throughput.
When a change in the traffic rate and modulation scheme from the BS is necessary, it has to communicate with the SS about the required changes. Now the SS has to communicate with its application layer for the proper data rate changes.

The BS part of the algorithm relies on two major QoS parameters i.e. packet loss rate and mean delay. It is important for the BS to maintain the channel condition information for both the uplink and downlink.

3.8 NS-2 SIMULATION MODEL

The cross layer scheduler proposed in this study, was implemented in the IEEE 802.16 module in the NS-2. The NS-2 is a widely used tool for the simulation of packet switched networks. It gives huge support for the simulation of TCP routing and MAC protocols over wired and wireless networks. The network elements in the NS-2 simulator are developed as classes in an object oriented manner. It has an Object Tool Command Language (OTCL) interpreter for easy user interface, and has an input model which is written in Tool Command Language (TCL) scripts.

As sown in Figure 3.5, NS is a discrete event simulator using OTCL script interpreter and network component object libraries. NS simulations scripts are developed using OTCL script language. An OTCL script typically initiates an event scheduler, sets up the network topology using the network objects and the supporting functions in the library, and programs the traffic sources to start/stop transmitting packets through the event scheduler.
3.8.1 Script Structure of NS-2

NS-2 architecture is designed to have separate data path and control path implementations. NS is written both on OTCL and in C++ also. The event scheduler and the basic network component objects in the data path are written and compiled using C++ and this approach significantly reduces the packet and event processing time. This newly compiled object is made available to the OTCL interpreter through an OTCL linkage. Each C++ class object has a matching OTCL object and the controlling functions and the variables function as member functions and member variables of the corresponding OTCL object.

Based on the functional requirements it is easy to add member functions and variables to a C++ linked OTCL object. The objects in C++ that do not need to be controlled in a simulation program or internally used by another object do not need to be linked to OTCL. Figure 3.6 show an object hierarchy example in C++ and OTCL and for each C++ object that has an OTCL linkage there is a matching OTCL object hierarchy.
3.8.2 NS-Miracle Library : Multi Interface Cross Layer Extension for NS

NS-MIRACLE enhances the functionalities offered by the NS-2 by providing an efficient and embedded engine for handling cross layer messages and also facilitates the option to have multiple modules within each layer of the protocol stack. For example multiple network, link, MAC or physical layers can be specified and accessed within the same node. This kind of framework facilitates the implementation and the simulation of modern communication systems in NS-2. In general it is possible to define multiple MAC/ARQ at layer 2 and so on for any other layer of the protocol stack. Additionally, the library contains a set of classes and methods for inter-layer/inter-protocol design and communication.

For instance, it allows for the exchange of any type of message/structure/command among modules/protocols. The library is intended to help researcher in implementing any type of cross layer and multi-technology solution.
3.8.2.1 Introduction

In fact, there are already several methods to give to NS-2 a more flexible structure to manage simultaneously multiple technologies at the same time, but all are uncoordinated. NS-MIRACLE introduce a standardization in the cross layer messaging in order to give to all the modules/algorithms the possibility of communicating to each other by means of ad hoc structured messages. All these features are introduced through a dynamic library for NS-2. This provides an efficient manner to make any change to the standard NS-2 installation.

3.8.2.2 Module

One of the main basic blocks of this structure is represented by the class Module. Module is a class designed to contain any protocols or modules that we usually place in stack in order to follow the OSI model in the designing process of the structure, an example of a possible architecture achievable is given in Figure 3.7. Since it has to be placed in a stack, its main function is to communicate with the other Modules placed in the above and bottom layers. Therefore NS-MIRACLE has implemented a channel for the communication between modules of adjacent layers such as defined in OSI structure, the Service Access Point (SAP). Due to the flexibility of the library, it is possible to define any number of modules in any layer and connect them to each other, by means of SAP.
This help us in the design of architecture in which are involved multiple technologies for example. SAP are also used to implement the tracing of simulation activity and tracing with an ad hoc format all the packets exchanged between modules at each desired level. Due to its own nature, we set up an ad hoc module for the channel, in fact it has to be common for all the nodes and have to store some knowledge about all of them. In order to gives to it such a kind of functionalities NS-MIRACLE has also defined an ad hoc channel to connect it to all the nodes that is called Channel SAP (ChSAP). For cross layer communication, there is a special channel called Cross-Layer SAP (CISAP), which is in charge of the delivering of the cross layer messages.
3.8.2.3 Cross layer message and node-core

Another important novelty introduced in this library is the communication between modules by means of Cross layer Messages, defined in the ClMessage class. These messages are exchanged through the CISAP and, since all modules are connected to each other’s with them, we may use ClMessage as cross layer message. The node-core with cross layer message exchange is shown in Figure 3.8.

Figure 3.8 Node core within the architecture
In fact, as mentioned in Module Section, each Module is equipped with a CISAP and this gives to it the connection to the node-core which is the structure in charge to dispatch all the CLMessage, i.e., we can consider node-core the cross layer bus which also has all the functionalities to manage it. By extending the CIMessage class, it is possible to obtain each ad hoc cross layer message is needed, i.e., it is possible to define new CIMessage with any structured parameter desires in order to implement the communication required.

There are two different types of communication: asynchronous and synchronous modes. In the first one is not requested any direct answer, the response is not mandatory and, if needed, it may be sent in another CIMessage when it is ready. While, in synchronous ClMessages, it is needed that the recipient answers directly in the same distance of the CIMessage received. This technique gives also to the programmer another important feature: the possibility to have variables shared by different modules. In fact, when a module receives a synchronou CIMessage, it is sharing the same instance of the CIMessage class with the source, so it may stores this instance and uses it as a variable to manage the state of common resources or to exchange information about the current module status.

3.8.2.4 Plug-in

Plug-in class is another feature introduced to gives more flexibility to the structures and to model new architecture. In fact, PlugIn is the parent class of Module and it is designed to attach any module direct to the node-core external from the stack of modules.
In Plug-in are defined only the cross layers communication functionalities of a Module, so it is equipped with the CISAP in order to be connected to the cross layer bus, placed in node-core. Plug-in structure in the NS-2 is shown in Figure 3.9.

This allows to Plug-In to contain all that features that is difficult to put in a fixed position of the stack. The class has been introduces in order to give to the programmer the possibility to place external intelligence that has to collaborate with several modules in the stack and so manage cross layer functionalities.

![Figure 3.9 Plug-in within the architecture](image-url)
3.8 SUMMARY

In this chapter the basic IEEE 802.16 WiMAX communication architecture has been discussed. The WiMAX QoS, protocol architecture has been presented. Since the research is focused on cross layered manner scheduling part of the MAC layer of the IEEE 802.16 wireless MAN, the cross layer contribution has been also presented.