An 802.16 Model for NS2 Simulator with an Integrated QoS Architecture

Ikbal C. Msadaa
Eurecom Institute
Mobile Communications Department
Sophia-Antipolis 06904, France
msadaa@eurecom.fr

Fethi Filali
Eurecom Institute
Mobile Communications Department
Sophia-Antipolis 06904, France
filali@eurecom.fr

Farouk Kamoun
ENSI
CRISTAL research team
La Manouba 2010, Tunisia
frk.kamoun@planet.tn

ABSTRACT

The IEEE 802.16 technology is emerging as a promising solution for BWA due to its ability to support multimedia services and to operate in multiple physical environments. Also, with data rates in excess of 120 Mbps, it provides a cost-effective alternative to wireline broadband access systems. Unfortunately no open-source simulation environment supporting this technology has been proposed so far. Therefore this work is meant to be a contribution to build an 802.16 simulation model for NS2, the most popular network simulator. In this paper, we present the details of design and implementation of the proposed simulation model. Our model includes a novel QoS architecture, and addresses 802.16 fixed BWA systems that use Orthogonal Frequency Division Multiplexing (OFDM) modulation and operate in TDD mode. The QoS architecture we propose consists of a call admission control (CAC) policy and a hierarchical scheduling algorithm that flexibly adjusts uplink and downlink bandwidth to serve unbalanced traffic. Both scheduling and CAC algorithms are based on an adaptive modulation and coding (AMC) scheme.

Categories and Subject Descriptors

I.6.m [**Computing Methodologies**]: Simulation, Modeling, and Visualization—*Miscellaneous*
; C.2.1.k [**Computer Systems Organization**]: Communication/Networking and Information Technology—*Network Architecture and Design, Wireless communication*

Keywords

802.16, NS2, OFDM, TDD, QoS, scheduling, CAC

1. INTRODUCTION

The IEEE 802.16 Standard [1] specifies the air interface for fixed BWA systems in the frequency ranges 10-66 GHz and below 11 GHz. The standard covers both the Media Access Control (MAC) and the physical (PHY) layers. The 802.16 MAC layer was designed to accommodate different PHYs and services, which address the needs of different environments. In this paper, systems of interest are those operating at frequencies below 11 GHz—where LOS is not required—and using Orthogonal Frequency Division Multiplexing (OFDM) modulation known as "WirelessMAN-OFDM" air interface.

The basic topology of an IEEE 802.16-based network consists of one Base Station (BS) and one or more Subscriber Stations (SSs). In point to multipoint (PMP), which is the only mode for sharing media considered in this paper, the SSs within a given antenna sector receive the same transmission broadcast by the BS—corresponding in general to the Internet Service Provider (ISP)—on the downlink channel (DL). Each SS is required to capture and process only the traffic addressed to itself (or to a broadcast or multicast group it is a member of). On the uplink channel (UL) however, the Time Division Multiple Access (TDMA) scheme is applied. DL and UL channels are duplexed using one of the two following techniques: Frequency Division Duplexing (FDD) and Time Division Duplexing (TDD). In this paper, we focus on 802.16 systems operating in TDD mode. Figure 1 shows an example of the OFDM frame structure in TDD mode. It is worth mentioning that the description of this structure is relevant for the remaining of the paper.

In the IEEE 802.16, the channel consists of fixed-length frames, as shown in Figure 1. Each frame is divided into a DL and an UL subframes. The standard [1] specifies that, when using TDD, the UL subframe and DL subframe durations shall vary within the same shared frame. The downlink subframe consists of one single PHY PDU while the uplink subframe consists of two contention intervals followed by multiple PHY PDUs, each transmitted by a different SS. The first contention interval is used for ranging which is the process of adjusting the Radio Frequency (RF). The second interval may be used by the SSs to request bandwidth since bandwidth is granted to SSs on demand. Two gaps separate the downlink and uplink subframes: transmit/receive transition gap (TTG) and receive/transmit transition gap (RTG). These gaps allow the BS to switch from the transmit to receive mode and vice versa.

The downlink PHY PDU consists of one or more bursts, each transmitted with a specific burst profile; a burst profile is a set of parameters describing the transmission properties (modulation type, forward error correction (FEC) type, etc.) corresponding to an interval usage code (IUC). Note that the standard supports an adaptive modulation and coding scheme (AMC); indeed each SS is required to adapt its IUC (a DIUC for the downlink and an UIUC for the uplink), and then the modulation and coding scheme in use based on measurements on the physical layer. The length of each burst is set by the BS. Indeed, at the beginning of each frame, the BS schedules the uplink and downlink grants—by mechanisms that are outside the scope of the Standard—and then broadcasts the DLFP, the

DL-MAP and the UL-MAP informing the SSs of its scheduling decisions. The DLFP describes the location and profile of the first downlink bursts (at most four). SSs using the same DIUC are advertised as a single burst. The DL-MAP, when sent, describes the location and profile of the other downlink bursts—if they exist. However the IEEE 802.16 Standard specifies that, at least one full DL-MAP must be broadcast within the Lost DL-MAP Interval even if there are less than five bursts in the current frame. The UL-MAP should be transmitted in each frame. It contains information elements (IE) that indicate the types and the boundaries of the uplink allocations directed to the SSs.

As can be seen from Figure 1, the UL-MAP, the DL-MAP (if one is transmitted) as well as the other broadcast MAC control messages are transmitted in the first DL burst. The profile of each downlink and uplink burst are specified in the Downlink Channel Descriptor (DCD) and Uplink Channel Descriptor (UCD), respectively. The BS broadcast the DCD and the UCD messages periodically—every DCD/UCD Interval. Referring to Figure 1, we note that each burst consists of one or more MAC PDUs. Each MAC PDU begins with a fixed-length MAC header followed by a payload and a CRC field. The burst may also contain padding bytes since each burst must consist of an integer number of OFDM symbols. UL bursts begin with a preamble used for PHY synchronization.

Thanks to its ability to support multimedia services (as we will see in Section 2) and to operate in multiple physical environments, the IEEE 802.16 technology emerges as a promising solution for BWA. Also, with data rates in excess of 120 Mbps [1], it provides a cost-effective alternative to wireline broadband access systems. Unfortunately no open-source simulation environment supporting this technology has been proposed so far. Therefore this work is meant to be a contribution to build an 802.16 simulation model for NS2, the most popular network simulator. In this paper, we present the details of design and implementation of the proposed simulation model. Our model includes a novel QoS architecture, and addresses 802.16 fixed BWA systems that use OFDM modulation and operate in TDD mode. The QoS architecture we propose consists of a CAC policy and a hierarchical scheduling algorithm that flexibly adjusts uplink and downlink bandwidth to serve unbalanced traffic. Both scheduling and CAC algorithms are based on an AMC scheme.

The remaining of this paper is organized as follows. Section 2 gives an overview of the IEEE 802.16 standard QoS related issues. The design and implementation details of our simulation model are presented in Section 3. In Section 4, we describe the scheduling and CAC algorithms, and the design of the proposed QoS architecture. In Section 5, we provide simulation results of our proposal. Finally, Section 6 concludes the paper and presents the related work.

2. QOS-SUPPORT IN IEEE 802.16

The standard defines a connection-oriented MAC protocol where all the transmissions occur within a context of a unidirectional connection. Each connection, identified by a unique Connection ID (CID), is associated to an admitted or active service flow (SF) whose characteristics provide the QoS requirements to apply for the PDUs exchanged on that connection. There are three types of service flows: (a) provisioned SFs for which the QoS parameters are provisioned for example by the network management system, (b) admitted SFs for which resources, mainly bandwidth, are reserved and (c) active SFs which are activated to carry traffic using resources actually provided by the BS. Each service flow is uniquely identified by a SFID; admitted and active service flows have also a CID. Service flows may be dynamically managed; they may be created, changed or deleted using DSA, DSC and DSD MAC management messages, respectively. The SF management procedure consists actually in exchanging DSx-REQ, DSx-RVD—sent by the BS when the transaction is SS-initiated—DSx-RSP and DSx-ACK messages, between the BS and the SS. Note that initiating the creation of a new service flow is a mandatory capability for a BS and an optional one for an SS.In order to facilitate the MAC SDUs delivery with the appropriate QoS constraints, the IEEE 802.16 Standard defines a classification process by which a MAC SDU is mapped to the associated connection and so to the SF related to that connection. The classification procedure is performed by classifiers consisting of a set of protocol-specific matching criteria applied to each MAC SDU entering the network.

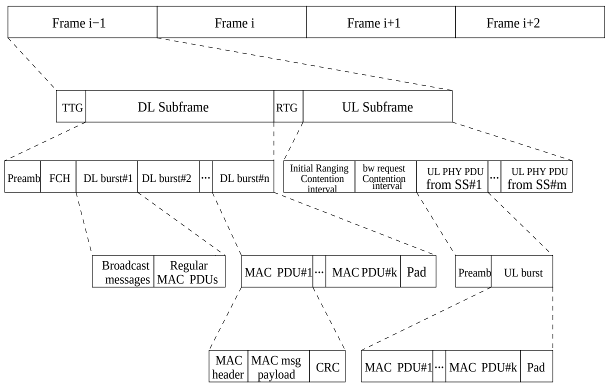


Figure 1: OFDM Frame Structure with TDD

Depending on the service to be tailored to each user application, the connection is associated with one of the following scheduling services supported by the 802.16 MAC protocol: Unsolicited Grant Service (UGS), Real-time Polling Service (rtPS), Non-real-time Polling Service (nrtPS), and Best Effort (BE)[1]. Each scheduling service is designed to meet the QoS requirements of a specific category of applications. The QoS parameters that must be specified when establishing a new service flow are listed in Table 1.

To inform the BS of its uplink bandwidth requirement, the SS may send a stand-alone bandwidth request header or just piggyback the request on a PDU, which is an optional capability. Other mechanisms such as bandwidth stealing and the use of poll-me (PM) bit[2] are also specified by the IEEE 802.16 Standard. It is important to mention that, whatever is the bandwidth request mechanism in use, bandwidth is always requested by an SS on a per-connection basis, nevertheless it is granted by the BS to an SS as an aggregate of grants. Therefore, since the SS receives the allocated bandwidth as a whole in response to a per-connection requests, it cannot know which request is honoured. The SS can then use the grant—specified in a Data Grant IE—either to send data or management messages or even to request bandwidth for any of its connections.

3. AN 802.16 MODEL FOR NS2

The 802.16 simulation model we propose is based on ns-2.29 version. It consists of new objects, simulating the interface queue and MAC layer, developed in accordance to what has

[1]The Extended rtPS (ertPS) scheduling service specified by the IEEE 802.16e-2005 [2] is out of the scope of this work.
[2]A field of a specific subheader of a MAC PDU, used by the SS to request a bandwidth poll for a non-UGS connection.

	UGS	rtPS	nrtPS	BE
Maximum Sustained Traffic Rate	X	X	X	X
Minimum Reserved Traffic Rate	—	X	X	—
Maximum Latency	X	X	—	—
Tolerated Jitter	X	—	—	—
Traffic Priority	—	—	X	X
Tx/Rx Policy	X	X	X	X

Table 1: Mandatory QoS parameters for each scheduling service

been specified by the IEEE 802.16 standard [1]. As channel object, we use the WirelessPhy class.

3.1 The interface queue (IFq)

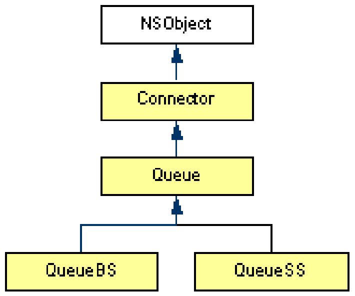
The *Queue* object, simulating the queue interface in ns-2, has two main functionalities: packet scheduling and buffer management. Packet scheduling is defined in [6] as the decision process used to choose the packets to be serviced and those to be dropped while the buffer management process refers to the specific discipline used to regulate the occupancy of a particular queue.

However in the specific case of 802.16, the packet scheduling functionality cannot be provided at the interface queue level. Indeed the scheduling decisions are made in the beginning of each frame to plan the transmissions during the whole frame. These decisions concern not only the data packets waiting in the buffers but also the MAC control messages to be transmitted in the current frame. Moreover to schedule UL transmissions, the BS should take into account the bandwidth requests formulated by the SSs during the last frame intervals and which are available only at the MAC level. For all these reasons, we have decided that the packet scheduling process should be guaranteed by the MAC layer while keeping the buffer management functionality at the IFq level.

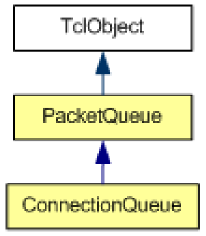
However to schedule transmissions on the DL, the *Mac* object at the BS needs to be aware of the backlog status. The same problem exists for UL transmissions at the SS. Therefore we have decided to add a module that consists of two main structures managed by both the *Queue* and *Mac* objects and whose role consists first in reflecting the backlog status (*QueueStatus* structure)—giving the sizes of the packets awaiting in each queue—and secondly in transmitting to the IFq the scheduling decisions made at the MAC level (*Grant* structure)—giving the amount of data to be serviced from each connection during the current frame interval. Note that, in order not to affect legacy ns-2 classes we didn't add any primitive between *Mac* and *Queue* classes; also we had conserved the callback mechanism to implement queue blocking when necessary. Further details on the relationship between the IFq and the MAC layer in the proposed model are given in 3.2.

Since the Queue class is designed by ns-2 to be used as a base class, we developed, for our 802.16 simulation model, two classes *QueueBS* and *QueueSS* derived from the *Queue* class (as illustrated by Figure 2(a)) and implementing the *enque* and *deque* virtual functions. Each class maintains a list of *ConnectionQueue* objects derived from the *PacketQueue* class, as can be seen in Figure 2(b). The *ConnectionQueue* implements the low-level operations necessary to hold on an ordered set of packets in a particular queue.

Note that a queue is allocated for each active MAC connection. A classification process is performed at the interface queue to map the MAC SDU to the corresponding CID and thus to



(a) The Queue Class



(b) The PacketQueue Class

Figure 2: QueueBS, QueueSS, and ConnectionQueue Classes

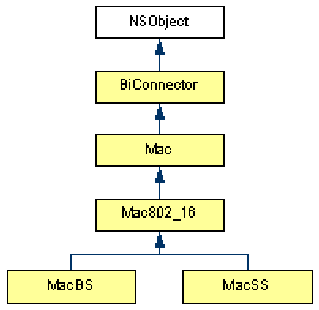


Figure 3: MacBS and MacSS Classes

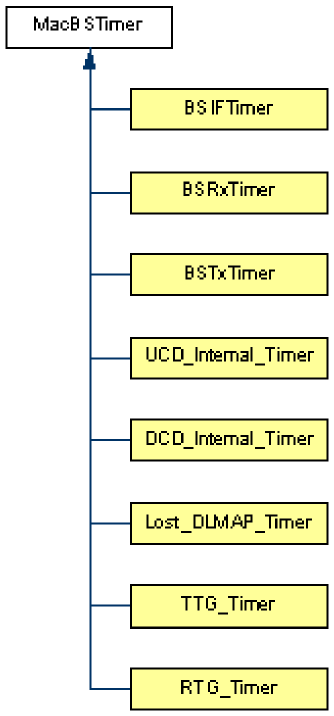


Figure 4: The MacBSTimer Class

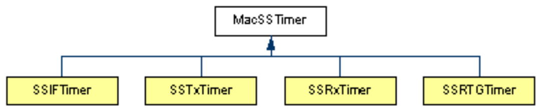


Figure 5: The MacSSTimer class

the appropriate queue. This mapping is fulfilled based on a matching table that is kept up-to-date by the MAC layer. Like *QueueStatus* and *Grant* structures, this matching table is visible for both IFq and MAC components.

3.2 The MAC layer

In the MAC layer, our model consists of a virtual class *Mac802_16* and two derived classes *MacBS* and *MacSS*. The class hierarchy existing between the mentioned classes and legacy ns-2 classes is shown in Figure 3. In addition to the classical sending and receiving mechanisms, *MacBS* and *MacSS* contain a *DSxManagement* and a *Scheduling* modules.

The *DSxManagement* module guarantees a dynamic management of service flows. Actually, only active service flows are supported by our simulation model so far. As we will see through examples in Section 5, connection addition and deletion may be initiated either by *MacBS* or by *MacSS*. All connection addition transactions are nevertheless processed by the *Admission Control* Module, existing at the BS side, and whose procedure is described in Section 4.
For each *Mac* class (*MacBS* and *MacSS* classes), we define a set of MAC states and timers whose expiration causes the transition from one state to another. These transitions reflect the frame structure described in Figure 1. Before moving to the description of the transmission state diagrams, let us first present the different timers used by *MacBS* and *MacSS*, respectively.

We have derived two virtual classes *MacBSTimer* and *MacSSTimer* from the virtual class *Handler*. These classes constitute the base classes for timers used by *MacBS* and *MacSS*, respectively. Several timers are defined by the 802.16 standard [1]. However only the most important ones were integrated in our simulation model. As shown in Figure 4, many classes are derived from *MacBSTimer*:

- *TTG_Timer* and *RTG_Timer*: correspond to the transmit/receive transition gap (TTG) and receive/transmit transition gap (RTG) interval timers. At the expiration of these timers, the *tx_state_* and *rx_state_* of *MacBS* transit from an active to a MAC_IDLE state, respectively.
- *DCD_Interval_Timer*, *UCD_Interval_Timer*, and *Lost_DLMAP_Timer*: defined for the DCD, UCD, and Lost DL-MAP intervals. At the expiration of these timers the flags *DcdToSend*, *UcdToSend* and *DlmapToSend* are set to 1 indicating the necessity of transmitting, during the current frame interval, a DCD, an UCD or a DL-MAP management message, respectively.

Other timers necessary to guarantee the transmit and receive functionalities are used:

- *BSIFTimer* and *SSIFTimer*: derived from *MacBSTimer* and *MacSSTimer*, respectively, these interface timers keep tracks of how long the interface will be in transmit mode—i.e.. it is actively transmitting bits into the air.
- *BSRxTimer* and *SSRxTimer*: these timers are started when the first bit of a packet is received at *MacBS* or *MacSS*, respectively. They are set to a value emulating the reception time of the whole packet. When these timers expire, *recvHandler()* calls the *recv_timer()* method which is responsible for identifying the type of the received packet (e.g. an UL-MAP message) and calling the appropriate method (in this case for example *recvUL_MAP()*).
- *BSTxTimer* and *SSTxTimer*: these timers are started by the *transmit()* method. When they expire the method *send_timer()* is called via the *sendHandler()* function. *send_timer()* is responsible for identifying the current MAC transmission state (by the means of the *tx_state_* value) and then deciding about the next step (depending on the value of the *tx_state_*). As we will see in the next paragraph, if we consider for instance the case of *MacBS*, after sending a DL-MAP message, the tx_state_ is MAC_DLMAP which means that the call of *send_timer()* would entail the transmission of an UL-MAP message and pass the tx_state_ from MAC_DLMAP to MAC_ULMAP.

More details about the whole process involving these timers and the different MAC states are given herein. At the beginning of each frame interval, the scheduling module, at *MacBS*, shares the available bandwidth among the management messages to send in the current frame, the DL transmissions (based on the backlog of DL connections) and the UL transmissions (in response to the bandwidth requests expressed by the SSs during the last frame intervals). This scheduling procedure starts when the RTG_Timer expires indicating the end of the MAC_IDLE mode, as presented in Figure 6 and the beginning of the active transmitting mode for *MacBS*.
First a DLFP is generated and transmitted on the air interface. It reflects the scheduling decisions made for the four DL bursts. A time emulating the transmission of a long preamble is nevertheless taken into account before the transmission of the DLFP message. Depending on the value of *DlmapToSend*, *MacBS* decides whether a DL-MAP should be sent during the current frame in which case it generates the corresponding message and broadcasts it to the SSs belonging to the network. The *tx_state_* is then set to MAC_DLMAP indicating that a DL-MAP message has been transmitted. The same tests are done for DCD and UCD messages. The flags *DcdToSend*, *UcdToSend*, and *DlmapToSend* are set to 0 after transmitting the corresponding messages and the timers *DCD_Interval_Timer*, *UCD_Interval_Timer*, and *Lost_DLMAP_Timer* are started.

Once the management MAC messages are transmitted, *MacBS* checks if a data packet has been transmitted by *QueueBS* and if the number of packets scheduled for the current frame is not null, it passes to the state MAC_DATA. A particular processing is reserved to the first data packet transmitted since the network entry. *MacBS* remains at the state MAC_DATA until the transmission of all the packets scheduled for a particular DL burst; in which case *tx_state_* is set to MAC_PAD indicating that padding bits are being transmitted to reach an integer number of OFDM symbols for the current DL burst. When all the packets scheduled for the current DL subframe are transmitted, *MacBS* enters in an idle state corresponding to the transmit/receive transition gap (TTG); the *tx_state_* remains at MAC_IDLE state till the next frame interval while the receiving state passes from an idle to an active state after the expiration of the *TTG_Timer*.

Figure 7 presents the transmission diagram of *MacSS*. The transmission process is much easier than in the case of *MacBS*. Indeed, at the expiration of the *SSRTGTimer*, the *tx_state_* of

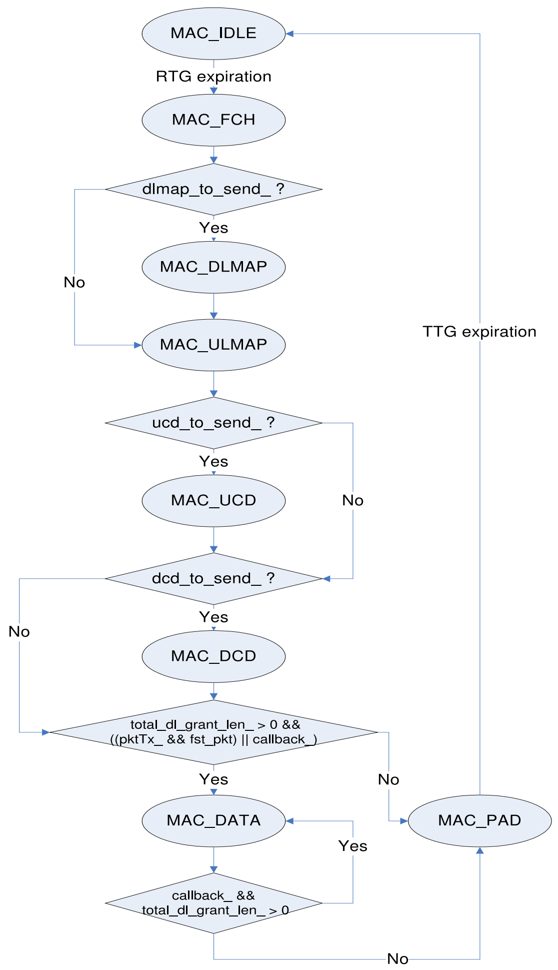


Figure 6: `MacBS` transmission states diagram

MacSS transits from MAC_IDLE to MAC_DATA state. Once the scheduled data packets are transmitted, *MacSS* transmits the bandwidth requests planned for the current frame. Moreover since all the packets transmitted by a particular SS are sent in the same burst, *MacSS* enters once in the MAC_PAD state to send all the necessary padding bits and then enters in an idle transmission state.

4. INTEGRATION OF A NOVEL 802.16 QOS ARCHITECTURE IN NS2

In this section, we present the design of our QoS architecture. Figure 9 illustrates the different elements and modules that constitute the proposed architecture as well as the interactions that exist between them. In this figure, we can see the proposed MAC logical structures for both BS and SS. Note that entities having the same name appear in both sides. In general, they play the same role. Differences will nevertheless be shown and discussed while explaining the role of each component.

- `Classifier`: As mentioned in Section 1, when a MAC SDU is received, it should be mapped onto a particular connection. In the architecture we propose, this task is accomplished by the `Classifier` based on a set of matching criteria such as the 5-IPv4 tuple[3]. As can be seen from Figure 9, the classification process is applied by both BS and SS to packets they are transmitting. When a BS or an SS receives an SDU, it refers to a matching table; if the considered SDU matches the criteria relative to a specific CID, it is then transmitted to the `Buffer Manager` (see Figure 9).

[3]IP source and destination addresses, source and destination ports, and QoS type field

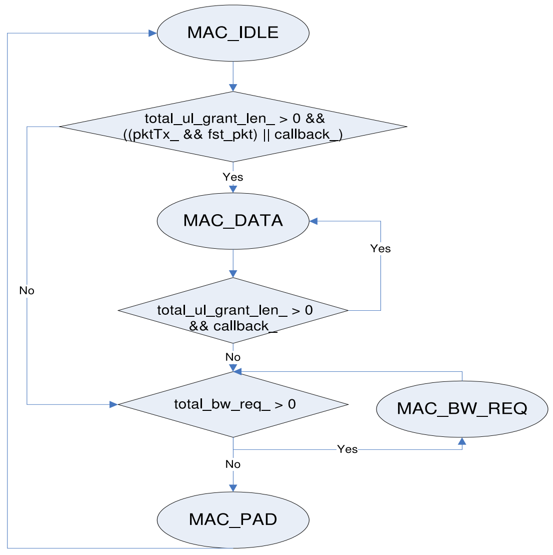

MAC_IDLE
total_ul_grant_len_ > 0 && ((pktTx_ && fst_pkt) || callback_)
Yes
No
MAC_DATA
total_ul_grant_len_ > 0 && callback_
Yes
No
total_bw_req_ > 0
Yes
No
MAC_BW_REQ
MAC_PAD

Figure 7: `MacSS` transmission states diagram

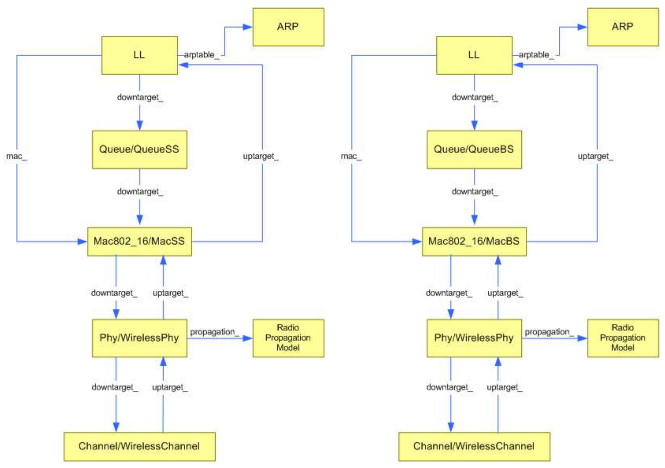


Figure 8: BS and SS node structures

It is possible that the SDU cannot be associated to any active connection. The `Classifier` checks then whether the packet can be mapped to a provisioned service flow (a SF that does not correspond to any CID), in which case the `Classifier` asks the `DSx Manager` to activate the corresponding SF. The `DSx Manager` may either respond by specifying a new CID or reject the Classifier demand—generally as a result of `Admission Control Module` decision (see Figure 9). In the latter case, as well as when the MAC SDU cannot be associated to any matching table entry, the packet is dropped.

- `Buffer Manager`: The `Buffer Manager` is responsible for managing the queues. It allocates a separate queue for each active MAC connection. The motivation behind organizing the queues in such a manner is that we simplify

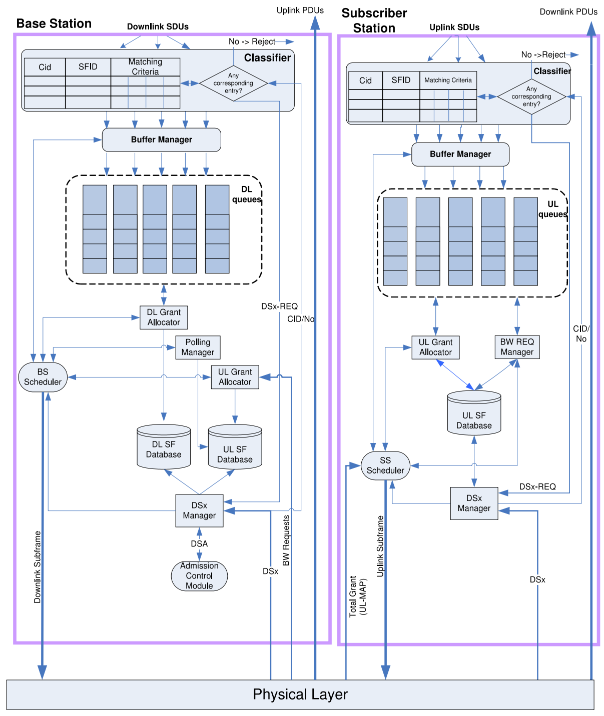


Figure 9: QoS architecture Design

the scheduling process since we need to know the queue size of each connection to perform bandwidth allocation. The *Buffer Manager* operates as follows. When it receives a MAC SDU from the *Classifier*, it puts it into the corresponding connection queue based on the CID already determined by the *Classifier*. It may also discard the packet if the buffer capacity of the corresponding connection is exceeded. As we will see later, the *Buffer Manager* is also responsible for delivering a specific amount of data from each connection queue according to what has been decided during the scheduling procedure.

- *DL/UL SF Database*: This database contains the QoS parameters of each DL/UL service flow. These parameters depend on the service flow type: whether it is associated to a UGS, rtPS, nrtPS or BE connection. The *DL* and *UL SF Database*s are used, during the scheduling procedure, to get the DL and UL QoS constraints, respectively. These databases are managed and maintained up-to-date by the *DSx Manager*.

- *DSx Manager*: The *DSx Manager* receives and treats all the messages exchanged during a service flow creation, deletion or modification procedure. For example, when the BS receives a DSD-REQ message, the *DSx Manager* treats the request and informs the *BS Scheduler* that a DSD-RVD message should be sent during the next frame interval. DSA-REQ messages, and in some cases DSC-REQ messages should be handled differently. In fact, since the purpose behind sending such messages is to create a new active service flow (DSA-REQ) or to activate an existing provisioned service flow (DSC-REQ), the request shall be handled by the *Admission Control Module* (see Figure 9) which decides whether it can be accepted or not.According to the request and to the *Admission Control Module* decision—when considering the case of a service flow creation or activation—the *DSx Manager* updates the *DL/UL SF Database* by entering, modifying or deleting the QoS characteristics of the considered service flow.

- *Admission Control Module*: Note that, although it exists only in the BS side (see Figure 9), the *Admission Control Module* is applied for both SS and BS-initiated connections addition requests. Its role is to check whether a request to create a new active service flow or to activate an existing one can be honored; other cases like using a more robust modulation technique may also require *Admission Control Module* action. The analytical procedure to follow will be detailed in Section 4.3. The decisions made by the *Admission Control Module* should be communicated to the *DSx Manager* which is responsible for planning and applying them.

- *Polling Manager*: As mentioned in Section 1, [1] proposes many bandwidth request mechanisms. Apart from UGS which does not need any explicit bandwidth request, polling can be applied to all types of scheduling services. Therefore, we integrate in the proposed architecture design the *Polling Manager* (see Figure 9) whose role consists in granting bandwidth request opportunities. The *Polling Manager* needs to have access to the *UL SF Database* to determine the SSs that are concerned by this technique, which are those having at least one non-UGS connection. As specified in the IEEE 802.16 standard [1], polling should be periodic for rtPS connections and regular for nrtPS connections. However, since the duration of the polling interval is not specified by the standard, it may be adapted to the QoS requested by each connection or just set to one or two frame intervals for instance. In all cases, during every polling interval, the *Polling Manager* should provide to the SSs having at least one rtPS or nrtPS connection, an opportunity to request bandwidth for each (n)rtPS connection.

- *DL Grant Allocator*: The *DL Grant Allocator* allocates bandwidth for DL connections according to the scheduling policy whose analytical basis will be explained in Section 4.2. To accomplish this task, the *DL Grant Allocator* needs to know the current status of DL queues, the QoS constraints of each DL service flow and the amount of remaining bandwidth—a parameter that is maintained by the *BS Scheduler*. To get the DL SF parameters, the *DL Grant Allocator* refers to the *DL SF Database*.

- *UL Grant Allocator*: As shown in Figure 9, this entity exists in both BS and SS structures.
  - At the BS, bandwidth is allocated based on bandwidth requests sent by the SSs for rtPS, nrtPS and BE connections. For UGS connections, the grants are made according to the QoS constraints of the associated service flow. In all cases, the *UL Grant Allocator* needs to have access to the *UL SF Database* to get the QoS constraints of UL service flows. After allocating bandwidth, the *UL Grant Allocator* informs the *BS Scheduler* of its grants decisions.
  - At the SS, the *UL Grant Allocator* operates in the same manner as the *DL Grant Allocator* at the BS with the exception of dealing with UL connections instead of DL connections.

- *BW REQ Manager*: As mentioned above, the *Polling Manager* has to allocate bandwidth to an SS specifically for the purpose of requesting bandwidth for its non-UGS UL connections. These allocations are then used by the *BW REQ Manager*, at the SS, to send bandwidth requests; they may optionally be used to send data. More generally, the

SS may use any uplink allocation to send data or bandwidth requests [1]. Therefore, since the `BW REQ Manager` does not know the exact amount of bandwidth to be used for requests, it should refer to the `SS Scheduler` which is the only component being able to make such decisions and to have information on bandwidth availability. Also, the `BW REQ Manager` needs to check the UL queues and `UL SF Database` in order to plan requests.

- `BS Scheduler`: The `BS Scheduler` represents the main element of the proposed architecture. In fact, it is responsible for synchronizing the work of the `DL Grant Allocator`, the `UL Grant Allocator` and the `Polling Manager` since it maintains information on the amount of the remaining bandwidth after each scheduling step. As we have chosen to apply a simultaneous scheduling between UL and DL, the `BS Scheduler` alternates between these three entities. Thus, after each grant decision, made either by the `DL/UL Grant Allocator` or by the `Polling Manager`, the `BS Scheduler` must update the amount of the remaining bandwidth. Besides, the `BS Scheduler` should remain informed of the `DSx Manager` decisions in order to plan the DSx management messages, such as DSA-REQ, DSC-RVD, DSD-RSP, to be sent in the current frame. Based on all the collected information, the `BS Scheduler` first generates the DLFP, the UL-MAP, and optionally the DL-MAP messages. These messages hold the scheduling decisions made by the BS (more specifically by the `DL Grant Allocator`, the `UL Grant Allocator` and the `Polling Manager`). Secondly, it either asks the `Buffer Manager` to transmit data according to what has been specified in the DLFP message, or just generates the appropriate management messages and send them to the SSs.

- `SS Scheduler`: As far as the SS is concerned, the `SS Scheduler` is the main element in the proposed QoS architecture. The `SS Scheduler` interacts with the BW-REQ Manager and the `UL Grant Allocator` in order to use the whole grant assigned by the BS. Note that only the `SS Scheduler` knows the amount of bandwidth that was granted to the SS's Basic CID. It has thus the responsibility of updating this parameter after each scheduling decision. Furthermore, the `SS Scheduler` has to share the whole granted bandwidth between the SS-initiated active connections.

4.1 Hierarchical scheduling structure

Let us consider the two-layer hierarchical scheduling structure proposed by Chen *et al.* [3]. In first layer scheduling, the authors have used two policies. The first one is a transmission direction based priority where they choose to attribute to DL a higher priority than UL. The second policy is a service class based priority applying the following scheme: $rtPS > nrtPS > BE$. The authors have then combined these policies using a strict priority scheme which assigns strict priority from highest to lowest to: DL_{rtPS}, UL_{rtPS}, DL_{nrtPS}, UL_{nrtPS}, DL_{BE}, and UL_{BE}. For DL and UL UGS connections, they have chosen to apply a fixed bandwidth allocation strategy. In second layer scheduling, they have proposed the use of Deficit Fair Priority Queuing (DFPQ) algorithm. In the scheduling structure we propose, we adopt the hierarchical aspect of scheduling proposed by Chen *et al.* [3]—and particularly the simultaneous scheduling between DL and UL—while avoiding the use of cyclic algorithms like DFPQ. The idea behind avoiding this approach is that when we consider a more realistic context, the BS does not have enough time to perform such a cyclic scheduling algorithm. As for the SS, the scheduling procedure is easier since the only connections to be managed are those it establishes with the BS in the UL direction.

4.2 The BS scheduling algorithm

Let S_{pkt}, S_{gmh}, and S_{crc} denote the size (in bytes) of an IP packet, a generic MAC header, and a CRC field, respectively. For a given connection j, N_i is the number of packets that are transmitted during the i last frame intervals and B_{max} and B_{min} stand for the Maximum Sustained, and the Minimum Reserved Traffic Rate, respectively. Let Q_i and R_i denote the number of packets that are waiting in the queue of connection j and the amount of its requested bandwidth (in bits). The use of R_i is meaningless in the case of an UGS connection. In the beginning of each frame interval i, the number of packets n_i to transmit per connection j should be calculated given N_{i-1}, B_{max}, S_{pkt}, and Q_i when j is a DL connection or R_i when j is a non-UGS UL connection. Indeed to compute n_i, we shall consider the three following cases:

- **Case 1: j is a DL connection:** The idea is that the scheduling policy tries to offer to connection j the possibility of transmitting a number n_i of packets big enough to guarantee to connection j the maximum rate allowed by the CAC module. Obviously, this does not imply that n_i can be bigger than the number of packets waiting in the queue of connection j:

$$n_i = \min\left(\left\lceil \frac{B_{max} * i * T_{frame}}{S_{pkt} * 8} \right\rceil - N_{i-1}, Q_i\right)$$

- **Case 2: j is an UL UGS connection:** In this case the scheduling policy should offer to connection j the possibility of transmitting a number n_i of packets big enough to guarantee to connection j to reach the Maximum Sustained Traffic Rate specified in its service flow:

$$n_i = \left\lceil \frac{B_{max} * i * T_{frame}}{S_{pkt} * 8} \right\rceil - N_{i-1}$$

- **Case 3: j is a non-UGS UL connection:** The bandwidth requirements of non-UGS connections must be explicitly formulated by the SS scheduler, which is the better-informed of the UL queues status. This bandwidth request, corresponding here to the parameter R_i, is formulated during the (i-1)th frame interval and represents the amount of bandwidth needed during the i^{th} frame interval. However, the BS scheduler must check whether R_i exceeds the Maximum Sustained Traffic Rate; in which case, it performs a kind of shaping by choosing the minimum between what has been requested by the SS scheduler and what would normally be planned by the BS scheduler in order to guarantee the maximum rate allowed by the CAC module for connection j for the i last frame intervals:

$$n_i = \min\left(\frac{R_i}{S_{gmh} + S_{pkt} + S_{crc}}, \left\lceil \frac{B_{max} * i * T_{frame}}{S_{pkt} * 8} \right\rceil - N_{i-1}\right)$$

Once n_i is computed for a given connection j, the BS scheduler calculates the resulting overhead and checks whether the available bandwidth—after deducting the overhead corresponding to gaps, preambles, control messages, contention intervals, and potential padding—allows the transmission of n_i packets otherwise n_i is decremented until reaching a feasible solution or $n_i$is null. In the latter case, the scheduler considers the next connection according to the priority scheme described in 4.1. The procedure stops when there is no more available bandwidth or all the bandwidth needs are satisfied. In this case, the remaining bandwidth is equally shared among SSs having at least one non-UGS connection.

4.3 Admission Control Policy

The admission control mechanism is performed when an SS or a BS attempts to establish a new active connection. It consists in the following procedure.

1. All BE connections addition requests are accepted since they don't have any QoS requirements.
2. For non-BE connections, the Admission Control Module checks whether the Minimum Reserved Traffic Rate can be guaranteed for existing connections and for the one attempting to be established. To do that :
   (a) it first computes the ceiling number n of packets to serve per frame for the new connection as follows: $n = \left\lceil \frac{B_{min} * i * T_{frame}}{S_{pkt} * 8} \right\rceil$,
   (b) then it estimates the overhead resulting from the transmission of these n packets. If the available bandwidth allows such grants—while considering the Minimum Reserved Traffic Rate for all existing non-BE connections—then the new connection is accepted otherwise it is accepted.

In both CAC and scheduling procedures, the time necessary to transmit n packets is computed based on the modulation and coding scheme in use i.e. on the number of bits per OFDM symbols.

5. USING THE IMPLEMENTATION

In this section, we will show how to make use of the 802.16 model we developed for ns-2 simulator. We will first describe the main parts of a tcl script, then we will present some basic simulation results aiming at validating the implemented model and evaluating the performances of the proposed QoS architecture.

5.1 Tcl simulation script

Figures 10, 11, and 12 present the main parts of a simple script. The first part of an 802.16 tcl script consists, as shown in Figure 10, in providing information about the different layers. Nodes are then configured using the specified parameters. As mentioned in Section 3, to simulate the queue interface and MAC layer, we use *QueueBS* and *MacBS* classes for a BS node and *QueueSS* and *MacSS* classes for an SS node, respectively. Both BS and SS nodes structures are presented in Figure 8. SSs are then attached to the corresponding BS using *base-station* command. Recall that the network entry process is not taken into account in our simulation model. Indeed SSs are explicitly attached to the BS and provided three pairs of management CIDs—using *set-mngt-cid* command. *set-diuc* and *set-uiuc* commands allow the specification of the DL and UL burst profiles to be used by a given SS. As specified by [1], the possible values are between 1 and 11 for a DIUC and between 5 and 12 for an UIUC corresponding to a burst profile. Each value corresponds in our model to a specific predefined modulation and coding scheme.

Even though we do not implement DSx management messages, our simulation model offers the possibility of adding and deleting active connections. Indeed as can be seen in Figure 11, a

```
set opt(chan)  Channel/WirelessChannel
set opt(netif) Phy/WirelessPhy
set opt(ifq)   Queue/QueueBS
set opt(mac)   Mac/Mac802_16/MacBS

$ns_ node-config -... \
                 -macType     $opt(mac) \
                 -ifqType     $opt(ifq) \
                 -...

set bs_ [$ns_ node]
$bs_ random-motion 0

#set BS position
...

set opt(ifq)   Queue/QueueSS
set opt(mac)   Mac/Mac802_16/MacSS

$ns_ node-config -macType $opt(mac) \
                 -ifqType $opt(ifq)

for {set i 0} {$i < $opt(nbr_ss) } {incr i} {
     set ss_($i) [$ns_ node]
     $ss_($i) random-motion 0
     $ss_($i) base-station [$bs_ node-addr]

}

#set SSs positions
...

set mac_bs [$bs_ getMac 0]

for {set i 0} {$i < $opt(nbr_ss) } {incr i} {
     set mac_ss($i) [$ss_($i) getMac 0]
     $mac_bs set_mngt_cid $mac_ss($i)
}

#set burst profiles
$mac_ss(1) set-diuc 3
$mac_ss(1) set-uiuc 5
```

Figure 10: Node configuration

```
set cid_(1) [$mac_bs add-connection $mac_ss(1) UGS $r_max $lat $jit $rxtx_pol $pkt]
set sce_cls 3
set cid_(2) [$mac_ss(1) add-connection $mac_bs $sce_cls $pkt]
$mac_bs delete-connection $cid_(1)
```

Figure 11: Connection management

```
set src_udp1 [new Agent/UDP]
$src_udp1 set class_ 1
$src_udp1 set prio_ 1
$src_udp1 set fid_ $cid_(1)
$src_udp1 set packetSize_ $pkt
set dst_udp1 [new Agent/Null]
$ns_ attach-agent $bs_ $src_udp1
$ns_ attach-agent $ss_(1) $dst_udp1
set app1 [new Application/Traffic/CBR]
$app1 set packetSize_ $pkt
$app1 set rate_ $r_max
$app1 attach-agent $src_udp1
$ns_ connect $src_udp1 $dst_udp1
$ns_ at 0.02 "$app1 start"
```

Figure 12: Generation of a CBR traffic

parameter's name	parameter's value
T_{frame}	10 ms
BW	20 MHz
T_{ttg}, T_{rtg}	$2 * T_{sym}$
T^{rng}	24 OFDM symbols
T^{bw}	20 OFDM symbols

Table 2: Simulations parameters

connection can be created in two manners; either by specifying all the required SF parameters (like for the first connection) or just by giving the identifier of a predefined service class. The CID assigned to the added connection depends on the admission control module decision. More precisely, a null value corresponds to a rejection decision. To make the difference between DL and UL connections, we consider the SS-initiated created connections as UL connections (e.g. the second connection in Figure 11) while assuming the BS-initiated ones as DL connections. To delete an active connection, only the CID needs to be specified. Note that, when creating a new connection, a "*$pkt*" parameter is specified. It corresponds to the traffic packet size in the case of an UGS connection—the packet size is fixed—or to the average packet size in the case of non-UGS connection. This parameter is necessary for the Admission Control module as we have explained in Section 4.

Once the nodes and the connections are configured, the traffic can be generated. Figure 12 illustrates an example of a CBR traffic generated over the DL UGS connection created before. We need first to make sure that the considered connection was accepted by the Admission Control module; i.e. the CID is not null. At this level, a mapping should be performed between the CBR traffic and the associated UGS MAC connection which corresponds to the classification process. The classifier we propose in our simulation model is based on the IP flow ID parameter which allows a simple identification of the corresponding traffic. Moreover, in order to make the classification process easier, we set the *fid* (flow identifier) to the CID value.

Other parameters such as frame duration, channel bandwidth, TTG, RTG and contention intervals duration have the default values given in Table 2. Nevertheless they may be set in the simulation script to different values.

5.2 Simulation results

Some basic simulation scenarios are presented in this section. They aim at validating the proposed simulation model and evaluating the performances of 802.16 networks. The first scenario studies the impact of the modulation and coding schemes on the maximum throughput that can be achieved in 802.16 networks. The second scenario investigates the performances of the proposed scheduling algorithm in terms of throughput, delay, and jitter. Different service types are considered in this scenario. Finally we study, in the third scenario, the variation of the instantaneous rate. The motivation behind running such scenario is seeing the effect of the absence of fragmentation on the frame-by-frame performances.

5.2.1 Variation of the average throughput for different modulation and coding schemes

In this first scenario, we are interested in knowing the maximum IP throughput that may be reached in 802.16 networks depending on the modulation and coding scheme in use. Therefore we consider an UL UGS MAC connection carrying CBR traffic. The maximum sustained rate specified for this connection is increased progressively till reaching a maximum value at which the admission control module would reject the connection adding request. In order to avoid a possible irregular behaviour of the simulator, due to the absence of DL traffic, we added a DL UGS connection carrying CBR traffic transmitted at a low rate (10 Kbps). We consider the same scenario for three different modulation and coding schemes: QPSK 1/2, 16-QAM 3/4 and 64-QAM 2/3. The packet size of 1500 bytes is considered in three cases.

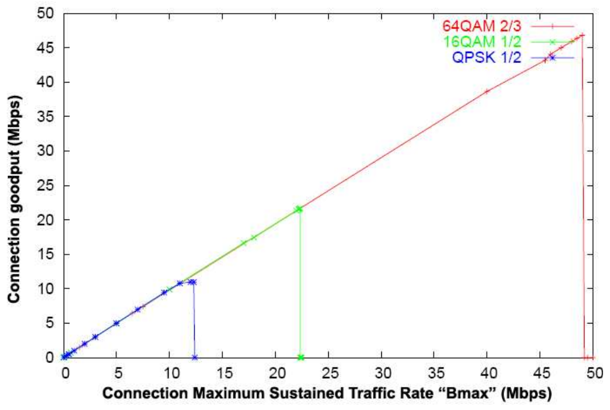


Figure 13: The average IP throughput for different modulation and coding schemes

Figure 13 depicts the obtained IP throughput for the three cases. The x axis corresponds to the maximum sustained rate specified when configuring the SF of the UL UGS connection. The y axis corresponds to the IP throughput—which is also equivalent to the MAC goodput—of the considered connection. The linear aspect of the three curves corresponding to the three modulation and coding schemes shows the efficiency of the proposed scheduling mechanism since it offers to the connection the desired rate, making use of the whole frame interval and not only a proportion dedicated for UL transmissions. The maximum values of IP throughput obtained with QPSK 1/2, 16-QAM 3/4 and 64-QAM 2/3 are 10.959 Mbps, 22.35 Mbps, and 46.768 Mbps. Figure 13 shows that specifying a maximum sustained traffic rate that exceeds these values leads to a rejection by the Admission Control module.

5.2.2 Variation of the average throughput, delay and jitter for different service types

It is true that delay and jitter constraints were not taken into account in the proposed scheduling algorithm. Nevertheless we are interested in seeing the effect of the proposed priority-based queuing discipline on them. Therefore, we consider a simple scenario involving three connections that differ in terms of service types (UGS, rtPS, and nrtPS) but configured with the same maximum sustained traffic rate. The same scenario is run for three different values of rate (20 Kbps, 300 Kbps, and 700 Kbps) in order to see the impact of increasing the traffic rate on delay and jitter. The obtained results for throughput, delay, and jitter are reported in Tables 3(a), 3(b) and 3(c), respectively.

The average throughput values reported in Table 3(a) confirm the results discussed above (in the first scenario). Indeed the IP throughput—resulting from the proposed scheduling strategy—is almost equal to what has been specified as maximum sustained traffic rate in the associated SF. Table 3(b) shows that the average delay is quiet short—does not exceed 7.25 ms— and almost the same for UGS and rtPS connections;

	UGS	rtPS	nrtPS
20 kbps	19.98	19.98	19.98
300 kbps	299.52	299.53	299.59
700 kbps	699.24	699.21	699.42

(a): Average throughput (Kbps)

	UGS	rtPS	nrtPS
20 kbps	5.61	5.79	393.61
300 kbps	6.72	6.76	23.29
700 kbps	7.25	7.02	11.44

(b): Average delay (ms)

	UGS	rtPS	nrtPS
20 kbps	1.61	1.56	1.08
300 kbps	4.22	4.67	4.39
700 kbps	2.47	2.34	2.37

(c): Average jitter (ms)

Table 3: The average throughput (a), delay (b), and jitter (c) for different service types

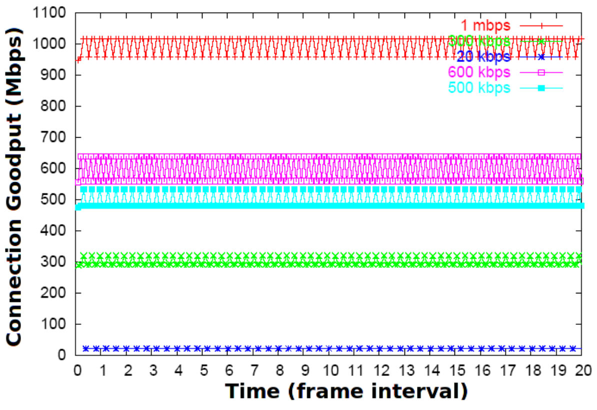


Figure 14: The IP instantaneous throughput

recall that this parameter is not relevant to nrtPS service type. We notice also that it is proportional to the traffic rate. As for jitter (3(c)), only UGS connection is concerned by this constraint. Nevertheless we notice that the obtained values are almost the same for the three service types. This may be explained by the fact that the packets are transmitted in bursts whatever is the associated service type.

5.2.3 Variation of the instantaneous throughput for different traffic rates

Through the previous simulation scenarios, we have shown that the scheduling strategy we propose meets the QoS requirements in terms of average throughput. In this scenario, we focus on instantaneous throughput. Therefore we consider five UGS connections with different rate requirements (from 20 Kbps to 1 Mbps). The motivation behind this scenario is to see the impact of absence of fragmentation on the frame by frame throughput performances. As depicted by Figure 14, the curve corresponding to the variation of throughput in time is not as smooth as it can be expected from that of the average throughput. Indeed in each curve, the throughput fluctuates between two values. Note that the throughput value is sampled with a period of 10 ms; If we had done the same every 20 ms ($2 * T_{frame}$), we would not have notice this curves behaviour. This can be justified by the fact that we do not implement fragmentation and thus the ceiling number of packets that should be scheduled in each frame interval corresponds to more or less the quantity of data that should be transmitted to reach the required rate.

6. DISCUSSION AND CONCLUSION

A WiMAX module, based on the OFDMA PHY specifications, has been developed by Chen *et al.* [4] for ns-2 simulator. The module includes a weighted round-robin (WRR) scheduling method in which each service type is associated with a percentage parameter corresponding to the ratio of bandwidth dedicated to that service type. For OFDM systems, a simulation model has been proposed by the National Institute of Standards and Technology (NIST) [8]. It does not include any CAC or QoS scheduling features. Nevertheless it offers the possibility of adding a new scheduler. Also the model was extended to support scanning and handover. A more basic simulation model, or at least its preliminary released version, was implemented by Yeom and Kim [9]. In this version, only the best effort service is available with a simple RR scheduling among the different flows. The authors do not implement any admission control or bandwidth request mechanism. Their main objective is to evaluate the performance of Wibro networks, and mainly TCP flows, based on IEEE 802.16. Unlike in [8] and [9], where we need to specify the DL ratio in advance, our model flexibly adjusts the DL/UL transmissions according to the bandwidth needs on a frame-by-frame basis. Thanks to this adaptive scheduling scheme, we avoid bandwidth waste and insure more efficient use of available resources to serve unbalanced traffic. Also, we do not need to estimate or set any bandwidth ratio for a specific service, as it was suggested by Chen *et al.* in [4].

In this paper, we have presented the details of design and implementation of an OFDM-based 802.16 simulation model for NS2. Our model includes a novel QoS architecture that consists of a CAC policy and a hierarchical scheduling algorithm that flexibly adjusts uplink and downlink bandwidth to serve unbalanced traffic. The simulation results presented in this paper reveal the efficiency of the proposed QoS architecture that makes efficient use of all the available resources to satisfy the bandwidth needs of the active connections while considering the modulation and coding scheme in use.

7. REFERENCES

[1] IEEE Std 802.16-2004. IEEE Standard for Local and metropolitan area networks- Part 16: Air Interface for Fixed Broadband Wireless Access Systems. 2004.

[2] IEEE Std 802.16e 2005. IEEE Standard for Local and metropolitan area networks Part 16-Amendment 2 and Corrigendum 1. 2005.

[3] J. Chen *et al.*. A Service Flow management Strategy for IEEE 802.16 Broadband Wireless Access Systems in TDD Mode. In *IEEE International Conference on Communications (ICC2005)*, 2005.

[4] J. Chen *et al.*. The design and implementation of WiMAX module for ns-2 simulator. *In 2006 workshop on ns-2: the IP network simulator WNS2 '06*, Oct. 2006.

[5] I. Msadaa. Intégration de la Norme IEEE 802.16 dans l'environnement de simulation NS2. Computer Science Master's thesis, Ecole Nationale des Sciences de l'Informatique, Tunisia, Dec. 2006.

[6] K. Fall and K. Varadhan. *ns Notes and Documentation.* Dec. 2005.

[7] J. Robinson. 802.11 MAC code in NS2 (version 2.28). http://www.ece.rice.edu/˜jpr/ns/docs/802_11.html.

[8] National Institute of Standards and Technology. http://www.antd.nist.gov/seamlessandsecure/doc.html.

[9] S. Kim and I. Yeom. http://cnlab.kaist.ac.kr/802.16/ ieee802.16.html