WiMAX-RBDS-Sim: An OPNET Simulation Framework for IEEE 802.16 Mesh Networks

Gustavo Vejarano
Wireless and Mobile Systems Laboratory
Electrical and Computer Engineering
University of Florida
gavafj@ufl.edu

Janise McNair
Wireless and Mobile Systems Laboratory
Electrical and Computer Engineering
University of Florida
mcnair@ece.ufl.edu

ABSTRACT

In this paper, a simulation model for IEEE 802.16 (WiMAX) wireless mesh networks with distributed scheduling is developed. It provides a framework for the evaluation of reservation-based distributed scheduling (RBDS) policies at the medium access control (MAC) layer. The simulation model, called WiMAX-RBDS-Sim, is developed under the OPNET event-driven simulation environment[1]. It provides interfaces for the integration of RBDS policies and link-establishment algorithms. As an example of the use of WiMAX-RBDS-Sim, a new RBDS policy called Sliced-GM-RBDS is proposed and evaluated. This policy is based on the GM-RBDS policy proposed in [27]. The WiMAX-RBDS-Sim simulation results show that Sliced-GM-RBDS outperforms GM-RBDS in terms of network stability. A link-establishment algorithm is also evaluated with WiMAX-RBDS-Sim to determine the time required for the completion of link establishments across the network. Finally, the performance of WiMAX-RBDS-Sim is evaluated in terms of simulation speed and memory usage.

Categories and Subject Descriptors

I.6.7 [**Simulation and Modeling**]: Simulation Support Systems—*environments*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; G.1.0 [**Numerical Analysis**]: General—*Stability (and instability)*

General Terms

Performance, Algorithms, Design

Keywords

wireless mesh networks, simulation, distributed scheduling, OPNET, IEEE 802.16

[1]WiMAX-RBDS-Sim is freely available through the OPNET University Program. It can be downloaded at http://plaza.ufl.edu/gavafj/opnet.html

1. INTRODUCTION

The problem of data scheduling in Wireless Mesh Networks (WMN) [2] has received special attention during the last few years due to its effects on key performance characteristics of WMNs. The scheduling problem consists of the calculation of a subset of links that are activated simultaneously for the transmission of data so that some or all of the network performance characteristics are optimized. These characteristics include the network capacity (i.e., the maximum amount of information the network can carry), end-to-end delay, and end-to-end throughput. The solution to the scheduling problem may enable the provisioning of Quality of Service (QoS) at the MAC layer by focusing on certain performance characteristics. For example, a scheduling scheme may focus on guaranteeing required levels of end-to-end delay and throughput.

The IEEE 802.16 standard [16] defines the physical and MAC layers for WMNs. Although the standard provides the architecture necessary for the implementation of any scheduling policy, no policy is specified by the standard. The MAC layer is based on time-division multiple access (TDMA), where time is divided into frames that are simultaneously used by non-interfering links. Each frame is divided into a control subframe and a data subframe. Scheduling messages are exchanged in control subframes containing scheduling information such as requests for use of data subframes and grants of data subframes. An election algorithm determines the subset of nodes that access the control subframe such that the transmissions of scheduling messages do not interfere with each other. The scheduling messages, control subframes, and the election algorithm enable the implementation of any given scheduling policy.

The standard also defines the procedures that nodes need to follow for joining an 802.16 WMN and for establishing physical links with other nodes. These procedures need to be finished before a node begins scheduling data packets.

In this paper, an OPNET [26] simulation framework, called WiMAX-RBDS-Sim, is proposed. Its purpose is to enable the evaluation of distributed scheduling policies and the link-establishment process in IEEE 802.16 WMNs. It implements the scheduling messages, election algorithm, and link-establishment and provides an interface for the integration of different distributed scheduling algorithms. To the best of our knowledge, WiMAX-RBDS-Sim is the first OPNET-based simulator for IEEE 802.16 WMNs with distributed scheduling. In Section 2, the related work and contributions are discussed. An overview of IEEE 802.16 WMNs is presented in Section 3. In Section 4, the architecture

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8679
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8679

of WiMAX-RBDS-Sim is explained in detail using the proposed algorithm Sliced-GM-RBDS as an example. In Section 5, the OPNET implementation of the WiMAX-RBDS-Sim architecture is described. In Section 6, the simulation results obtained with WiMAX-RBDS-Sim for the proposed algorithm are provided and discussed. The performance of WiMAX-RBDS-Sim is evaluated in Section 7. Finally, the paper is concluded in Section 8.

2. RELATED WORK

The research community has recently investigated the challenges of centralized and distributed scheduling in 802.16 WMNs. In this paper we focus on the distributed scheduling problem[2].

Distributed-scheduling research can be classified into two groups: election-algorithm-based policies and RBDS policies. In [6, 11], the performance of the election algorithm is evaluated theoretically and by means of simulation. In [33, 28, 3, 17], the performance is improved in terms of number of collisions and control-subframe utilization by dynamically adjusting the parameters of the algorithm. In [14, 32, 31], a QoS differentiation scheme is proposed based on the adjustment of the election-algorithm parameters so that a node transmits scheduling messages more often when its data transmissions have higher priority. In [20], a congestion control mechanism is proposed in which one of the election-algorithm parameters is adjusted by each node according to the congestion measured locally. The effect of this adjustment is that a node waits longer for sending scheduling messages when the local congestion is high.

The second group for distributed-scheduling research considers RBDS policies. These policies are based on the reservation of future data subframes so that no two interfering links are assigned the same data subframes. In [9], the nodes prioritize the traffic flows going through them according to the weights of each of the flows, and the number of data subframes reserved for each flow is proportional to its weight. The data subframes eligible for reservation are finite in the sense that only the frames that are a certain number of frames away from the current one can be included in the reservations. This set of data subframes is known as the schedule horizon. In [29], data-subframe utilization is improved by allowing every reservation to contain non-contiguous data subframes that are all assigned to the same link. In this way, the number of wasted data subframes (i.e., data subframes that are never assigned to any link) is reduced. In [19], the reservations for each link are calculated based on the statistical characteristics of the data traffic generated at the source node in order to reduce the scheduling overhead. In [12], an end-to-end reservation scheme is proposed for constant-bit-rate flows such as voice-over-IP. Other simpler reservation-based distributed scheduling schemes are proposed in [18, 8, 21], and a broadcasting algorithm that aims to minimize the number of reservations per broadcast is proposed in [34].

2.1 Distributed-Scheduling Simulators

The IEEE 802.16 standard defines two operating modes for metropolitan access networks. These are the Point-to-Multipoint (PMP) and Mesh modes. In the literature, research has been mostly focused on simulators for the PMP mode [4, 23, 24]. To the best of our knowledge, there are two simulators for IEEE 802.16 WMNs with distributed scheduling that have been discussed in the literature. These are WiMsh [10], which is a patch for the Network Simulator 2 (ns-2) [1], and NCTUns [15], which is a network simulator and emulator that includes a simulation tool for IEEE 802.16 WMNs with distributed scheduling.

WiMAX-RBDS-Sim is a packet-level simulation model for the OPNET discrete event simulator [26]. It is an implementation of the MAC Common Part Sublayer given in the standard [16]. The architecture of WiMAX-RBDS-Sim was conceived for the simulation of the RBDS policies defined in [27] which are based on the input/output queue concepts.

Although, WiMAX-RBDS-Sim and WiMsh have similar functionalities, there are differences between these two simulators in terms of architecture and network topology. These differences are as follows.

The architecture of WiMAX-RBDS-Sim is based on a theoretical framework that focuses on a stability analysis for wireless multi-hop networks under any RBDS policy. WiMAX-RBDS-Sim provides the means for obtaining simulation results for such policies in order to compare them with the theoretical results obtained for the same policies under the theoretical framework.

WiMAX-RBDS-Sim implements the link establishment algorithm, which is not implemented in WiMsh, and provides the framework for the implementation and evaluation of improvements to the link establishment process.

In WiMAX-RBDS-Sim, the links established between the nodes depend on the location of the nodes, the physical channel model selected for the simulation, and the nodes' antennas. Therefore, the topology of the network is calculated based on those parameters. In WiMsh, the links are given as a simulation parameter by specifying the network topology from a set of predefined topologies.

2.2 Contributions

In this paper, the design and implementation of WiMAX-RBDS-Sim is presented. WiMAX-RBDS-Sim has the following characteristics:

- It provides a framework for the implementation and evaluation of algorithms that adjust the parameters of the election algorithm for each node dynamically. WiMAX-RBDS-Sim allows the evaluation of these algorithms in terms of scheduling delay, control-subframe utilization, and number collisions in control subframes.
- It provides a framework in which scheduling policies can be implemented and evaluated in terms of capacity (i.e., the set of maximum data input rates that the policy can handle while guaranteeing the stability of all the queues in the network), throughput, and delay.
- It provides a framework for the implementation and evaluation of link-establishment algorithms. This evaluation is performed in terms of link-establishment delay (i.e., the time required by two nodes to establish links between them).

3. IEEE 802.16 WMN OVERVIEW

In an 802.16 WMN there are two types of nodes: Base Stations (BS) and Subscriber Stations (SS). The 802.16 WMN is

[2]For more information on centralized scheduling, please refer to [22, 7].

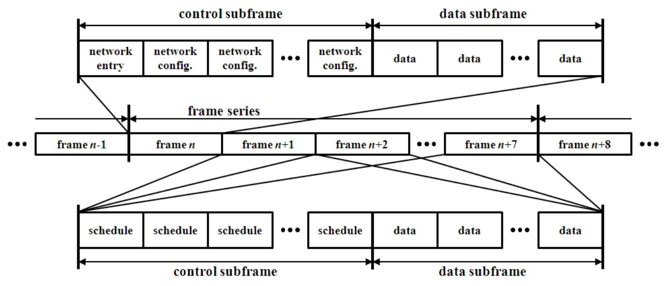

Figure 1: Frame Structure of the IEEE 802.16 Mesh Mode

connected to external networks such as the Internet through BSs. Unidirectional links can be established between any of these nodes, and the information is transmitted on a hop-by-hop basis. Each link is uniquely identified in the network with both the node ID and the link ID. The link ID identifies the link among the set of outgoing links of a node, and the node ID identifies the node from which the link originates. The system access follows a frame-based approach which is shown in Figure 1. Each channel is divided in time into series of frames. In Figure 1, each of these series consists of eight frames (i.e., from frame n to frame $n+7$). A frame is further divided into two subframes: a control subframe and a data subframe. The control subframes are used for establishing links and scheduling data. The data subframes are used for data transmission. Control subframes are divided into slots, which are used for transmitting scheduling packets and link-establishment packets. The control subframe of the first frame of a series (e.g., frame n in Figure 1) is used for transmitting link-establishment packets[3]. The control subframes of the following frames of the series (e.g., frames $n+1$, $n+2$, ..., $n+7$ in Figure 1) are used for transmitting scheduling packets only. Data subframes are divided into slots which are used for transmitting data packets.

Nodes access the control slots (i.e., the slots in control subframes) using the IEEE 802.16 election algorithm [6]. The election algorithm guarantees that whenever a node transmits, all the other nodes in its 2-hop neighborhood stay quiet, where the 2-hop neighborhood of a node consists of all the nodes that are at most 2 hops away from it. In this way, collisions in control slots are avoided and link-establishment and scheduling packets are successfully received.

Nodes access the data slots (i.e., slots in data subframes) using a scheduling policy that is implemented by exchanging scheduling packets in the control subframe. The goal of the scheduling policy is to avoid unused data slots and collisions of data packets while optimizing one or more metrics of the network such as its capacity.

3.1 Data-Slot Scheduling

The scheduling policy for accessing data slots is not specified in the IEEE 802.16 standard. The standard only defines the Mesh-Distributed-Scheduling (MSH-DSCH) message, which contains scheduling information and is carried by scheduling packets, so that different distributed scheduling policies can be implemented. The information contained in an MSH-DSCH message is organized in Information Elements (IE) as follows.

- *Request IE*: A node can make several requests simultaneously on a one-request-per-link basis. The information included in a request is the link ID, number of requested data slots per data subframe, and number of requested data subframes. The number of data subframes may be infinite so that streams of information can be transmitted in the link.
- *Availability IE*: A node notifies its 1-hop neighbors[4] of the data slots it has available for reservation. This IE specifies a set of available data slots with a start frame number, number of frames, start data slot number, number of data slots, direction (i.e., the minislots may be available for transmission, reception, or both of them), and the channels the available data slots belong to.
- *Grant IE*: This IE includes the same parameters specified for the Availability IE. However, these are used for specifying a set of data slots that have been *assigned to* a link.

The scheduling procedure follows a three-way handshake. First, a node sends an MSH-DSCH message to one of its 1-hop neighbors requesting a set of data slots. In the message, the node also includes the set of data slots that it has available for reservation. The 1-hop neighbor grants the request by replying with another MSH-DSCH message that specifies a set of data slots that satisfies the availability of data slots at both nodes. Finally, the first node confirms the reservation of such set of data slots by echoing the grant in another MSH-DSCH message. By following this three-way handshake, the 1-hop neighbors of the two nodes become aware of the data slot reservation so that the data slots in the grant become unavailable for them.

3.2 Link Establishment

In order to exchange data packets in both directions, two 1-hop neighbors need to establish two links (i.e., one link in each direction). This is achieved by means of a three-way handshake performed with Mesh Network Configuration (MSH-NCFG) messages which are transmitted in link-establishment packets. The handshake is initiated by the node with lowest ID. First, the lowest-ID node sends a *challenge* to its new 1-hop neighbor. Second, the 1-hop neighbor replies with a *challenge-response* and the ID for its outgoing link (i.e., the incoming link of the lowest-ID node). Finally, the lowest-ID node replies with an *accept* and the ID for its outgoing link (i.e., the incoming link of the 1-hop neighbor). The link-establishment information that the nodes exchange during the handshake (i.e., *challenge*, *challenge-response*, *accept*, and link IDs) is written on Link-Establishment IEs, and these are attached to MSH-NCFG messages.

4. WIMAX-RBDS-SIM ARCHITECTURE

The WiMAX-RBDS-Sim architecture is shown in Figure 2. The architecture consists of the physical, MAC, and logical-link layers. The nodes all share the same architecture. There are 10 radio channels in the physical layer. The MAC layer

[3]These packets carry network-configuration information too, but this is out of the scope of this paper.

[4]Two nodes are 1-hop neighbors if and only if they both can receive packets from each other directly.

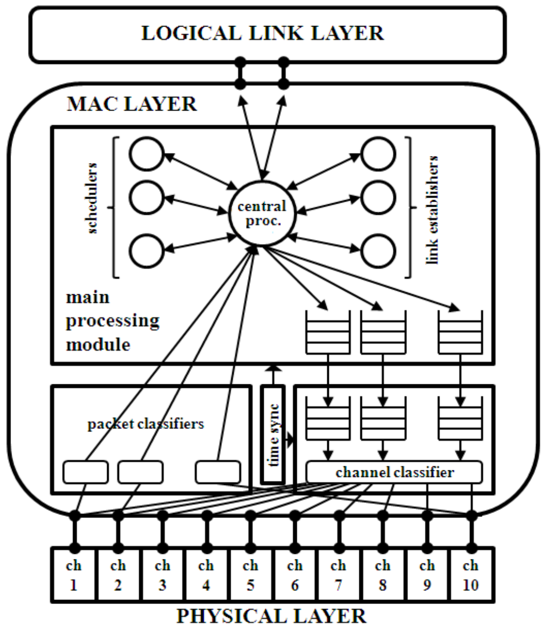

Figure 2: WiMAX-RBDS-Sim Architecture.

implements the option for distributed scheduling in wireless mesh networks of the IEEE 802.16 standard [16][5]. It consists of four main functional modules. These are the main-processing module, the packet-classifier module, the output-queue and channel-classifier module, and the time-synchronization module. The logical-link layer is the node's source and sink of data packets. It does not implement any logical-link protocol.

Each of the radio channels can be set at the configuration modes specified in the standard [16]. These modes determine the frame length, channel bandwidths, and forward-error-correction and modulation schemes. They are specified as simulation parameters.

In order to make comparisons between theoretical and simulation results, the WiMAX-RBDS-Sim architecture follows the 1-hop traffic model[6] (i.e., the destination node of every packet is always 1 hope away from the source node). The logical-link layer can be set at some data-packet generation rate, and the data-packet source generates data-packets at such rate for each of the node's 1-hop neighbors.

In the MAC layer, the packet-classifier module classifies all the received packets into scheduling, link-establishment, and data packets and forwards them to the main-processing module. In this module, there is one classifier per radio channel. The output-queue and channel-classifier module queue packets that have already been scheduled for transmission and forward them to the right radio channel at the transmission time indicated by their respective schedules. In this module, there is one output-queue per radio channel.

The main-processing module performs the following tasks:

- establishes incoming and outgoing links with each of the node's 1-hop neighbors (i.e., the links in the WiMAX-RBDS-Sim architecture are unidirectional)
- queues unscheduled data packets and forwards them to the output-queue and channel-classifier module once they are scheduled
- schedules for transmission data-packets generated at the logical-link layer
- forwards to the logical-link layer the received data-packets that are destined to the node

The main-processing module consists of several processors that are dedicated to specific tasks and a set of queues where unscheduled data packets are stored temporarily. There are three types of processors: main processor, scheduler, and link establisher. There is only one main processor in the module. The main processor controls the flow of packets within the module and creates and destroys scheduler and link-establisher processes. The schedulers are created and destroyed at the onset and at the end of the simulation respectively. There is one scheduler per channel. A scheduler generates and processes for its assigned channel the Request, Availability, and Grant IEs transmitted and received by the node respectively. The link-establisher processes are created and destroyed dynamically by the main processor during the simulation. A link establisher is created when a 1-hop neighbor is detected for the first time. The link-establisher generates and destroys the Link-Establishment IEs transmitted and received by the node for establishing the incoming and outgoing links with the detected 1-hop neighbor. The main processor destroys the link-scheduler once these links have been established.

The flow of packets within the main-processing module consists of two directions. These correspond to the transmission and reception of packets. In the direction of transmission, data packets are generated by the logical-link layer and delivered to the main processor. The main processor stores them in the module's queues. These are the input queues. There is one input queue per outgoing link. Depending on the lengths of the input-queues, the main processor invokes the schedulers which generate the scheduling IEs (i.e., Request IE, Availability IE, and Grant IE). With these IEs and based on the election algorithm, the main processor generates scheduling packets and calculates their schedules. The scheduling packets are forwarded to the output queues along with their schedules. When the schedulers are finished scheduling any packets in the input queues, the main processor removes those packets from the input queues and forwards them along with their schedules to their corresponding output queues. Also, the main processor generates link-establishment packets with the Link-Establishment IEs generated by the link-establishers and calculates their schedules based on the election algorithm. These packets are also forwarded along with their schedules to their corresponding output queues. In the direction of reception, the main processor first checks whether the received packets are directed to the node it belongs to. If that is the case, the main processor forwards data packets to the logical-link layer,

[5]Specifically, the MAC layer implements the MAC Common Part Sublayer for IEEE 802.16 wireless mesh networks with distributed scheduling.

[6]The efficiency of link scheduling algorithms is usually evaluated with the maximum throughput that the algorithm can support while guarantying the stability of the network (i.e., the queues return to their empty state with some probability greater than zero). The theoretical results obtained for the link-scheduling efficiency are usually obtained under the assumption of 1-hop traffic as shown in the simulation results in [25].

reads IEs from scheduling and link-establishment packets, and destroys these types of packets (i.e., scheduling and link-establishment packets). Scheduling IEs are passed to their corresponding schedulers, and Link-Establishment IEs are passed to their corresponding link-establishers.

4.1 The Link-Establishers

The link-establisher process model is shown in Figure 3. The model is a state transition diagram in which actions are taken at every state, and state transitions take place when the main processor invokes the link-establisher. The link-establisher communicates with the main processor by writing on and reading from a buffer that can be accessed by both processes only.

The link-establisher is able to perform two different roles present in the three-way handshake. These are the challenger and the replier. The challenger corresponds to the link-establisher at the 1-hop neighbor with lower ID which initiates the handshake by sending a *challenge*. The replier corresponds to the link-establisher at the 1-hop neighbor with higher ID which replies to the *challenge* by sending a *challenge-response*. In the initial state, named *init* (see Figure 3), the process decides its role. The challenger jumps from *init* to *prep-challenge*, and the replier jumps from *init* to *prep-response*.

Upon arrival to *prep-challenge*, the link-establisher prepares the Link-Establishment IE with the *challenge* that initiates the handshake and waits at state *wait* until the main processor is ready to process the Link-Establishment IE. When the main processor is ready, the link-establisher jumps to *tx-challenge*, writes the Link-Establishment IE on the buffer, and sets a timer for the retransmission of the Link-Establishment IE in case it receives no response from the 1-hop neighbor. Then, the link-establisher jumps to *wait-response* where it waits for a response. If the response is not the expected *challenge-response*, it jumps to *abort* where it aborts the handshake and destroys itself. In this case, the main processor will later create another link-establisher for establishing the links with the 1-hop neighbor. If the response is the expected *challenge-response*, the link-establisher, at state *prep-accept*, reads the incoming-link ID assigned by the 1-hop neighbor, and creates another Link-Establishment IE with an *accept* and outgoing-link ID. When the Link-Establishment is written on the buffer at state *tx-accept*, the link-establisher jumps to *finish*, notifies the logical-link layer that a new outgoing link has been established and destroys itself.

Upon arrival to *prep-response* (i.e., when the link-establisher takes the replier role), the link-establisher creates a Link-Establishment IE with a *challenge-response* and outgoing-link ID and waits at state *wait* until the main processor is ready to process the Link-Establishment IE. When the main processor is ready, the link-establisher jumps to *tx-response*, writes the Link-Establishment IE on the buffer, and sets a timer for a retransmission in case no response is received from the 1-hop neighbor. When the response is received, the link-establisher checks whether it is another *challenge*, the expected *accept*, or an unexpected response (i.e., *reject*, *challenge-response*). If it is another *challenge*, the link-establisher assumes that the previously transmitted *challenge-response* was not received and the Link-Establishment IE is retransmitted. If it is the expected *accept*, the link-establisher jumps to *get-LinkID* where it reads the incoming-link ID and jumps to *finish*. At state *finish*, it notifies the logical-link layer that a new outgoing link has been established and destroys itself. If it an unexpected response, the link-establisher jumps to *abort*.

4.2 The Schedulers

The schedulers implement the distributed-scheduling algorithm used for scheduling the data packets stored in all the input queues. To illustrate the scheduling process, we propose the Sliced-Greedy-Maximal-RBDS (Sliced-GM-RBDS) algorithm, based on the GM-RBDS policy proposed in [27]. The Sliced-GM-RBDS algorithm uses sets of data slots like the ones shown in Figure 4. In order to specify these sets, the data subframes are all divided or *sliced* into a given number of data-slot groups of the same size. For example, in Figure 4, there are 12 data slots per frame numbered from 4 to 15 which are divided into 4 data-slot groups. These groups correspond to the data slots numbered 4-6, 7-9, 10-12, and 13-15. A set of data slots is specified with a data-slot group and a frame range. For example, if the group corresponds to data slots numbered from 7 to 9 and the frame range consists of frames numbered from 11 to 20, the set of data slots consists of 30 data slots (i.e., 10 frames and 3 data slots per frame). This set is shown in Figure 4 along with other three different sets. In WiMAX-RBDS-Sim, there are 256 minislots per frame as specified in the standard [16], and the size of the data-slot groups is a simulation parameter.

The Sliced-GM-RBDS algorithm is based on the following scheduling policy. Whenever a node transmits a scheduling packet,

- for every outgoing link, request a set of data slots that covers the longest frame range that can be entirely covered with unscheduled data packets
- grant the longest requests that do not overlap with each other
- for every request and grant made, set its start frame (i.e., the first frame of the frame range covered by the request/grant) at the earliest possible frame such that overlaps are avoided

In the algorithm, a Request IE (defined in Section 3.1) is always generated along with an Availability IE. The Request IE specifies the size of a set of data slots (i.e., the number of requested data slots). The Availability IE specifies a set of data slots whose size is at least equal the size of the set of data slots specified in the Request IE. The Availability IE's set indicates all the data slots that the node's 1-hop neighbor is allowed to include in the grant for the request. Therefore, this set should not include any of the data slots that belong to any of the grants heard by the node sending the request, otherwise the grant generated at the node's 1-hop neighbor may overlap one or more of such grants, and the node will not be able to confirm the grant. On the other hand, the 1-hop neighbor is able to grant the request if it finds a set of data slots within the Availability IE's set that does not overlap any of the grants that it has heard and that has the size specified in the Request IE.

The scheduler process model is shown in Figure 5. It consists of two states. These are *init* and *schedule*. In state *init*, the scheduler initializes itself according to simulation parameters such as the number of data-slot groups. This

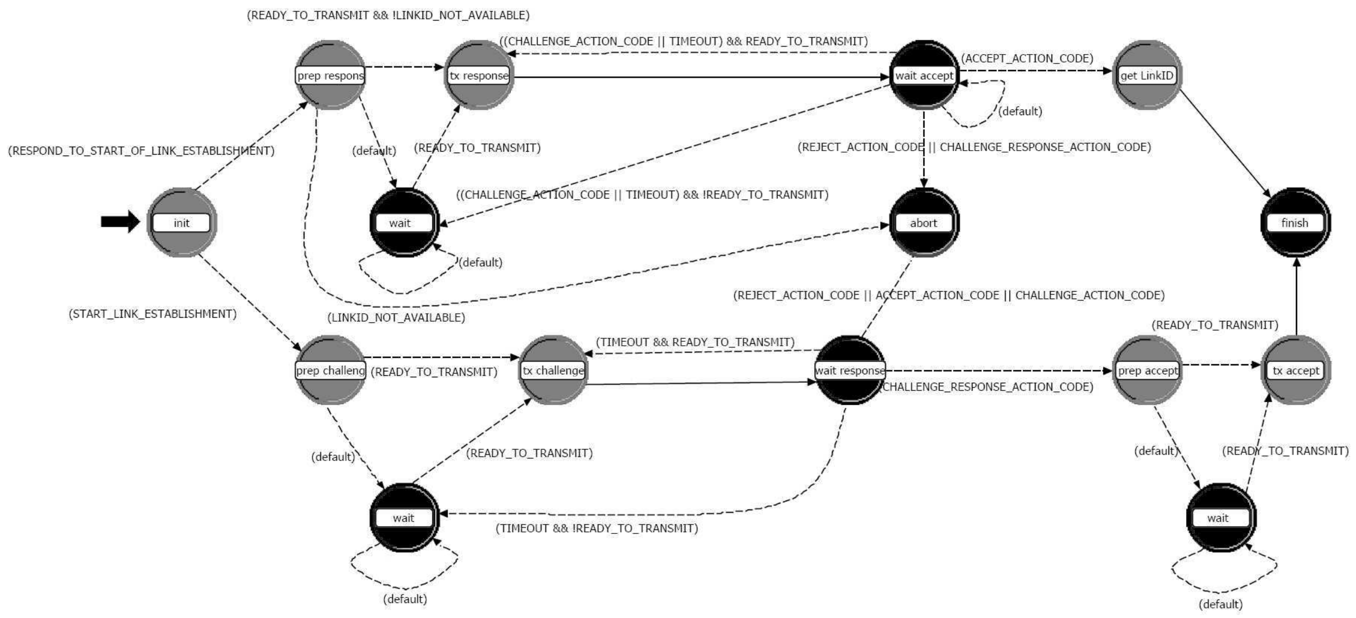
(READY_TO_TRANSMIT && !LINKID_NOT_AVAILABLE)
((CHALLENGE_ACTION_CODE || TIMEOUT) && READY_TO_TRANSMIT)
(ACCEPT_ACTION_CODE)
prep respons
tx response
wait accept
get LinkID
(default)
(RESPOND_TO_START_OF_LINK_ESTABLISHMENT)
(READY_TO_TRANSMIT)
(REJECT_ACTION_CODE || CHALLENGE_RESPONSE_ACTION_CODE)
((CHALLENGE_ACTION_CODE || TIMEOUT) && !READY_TO_TRANSMIT)
init
wait
abort
finish
(REJECT_ACTION_CODE || ACCEPT_ACTION_CODE || CHALLENGE_ACTION_CODE)
(START_LINK_ESTABLISHMENT)
(LINKID_NOT_AVAILABLE)
(TIMEOUT && READY_TO_TRANSMIT)
prep challeng
tx challenge
wait response
prep accept
tx accept
(CHALLENGE_RESPONSE_ACTION_CODE)
(TIMEOUT && !READY_TO_TRANSMIT)

Figure 3: Link-Establisher Process Model

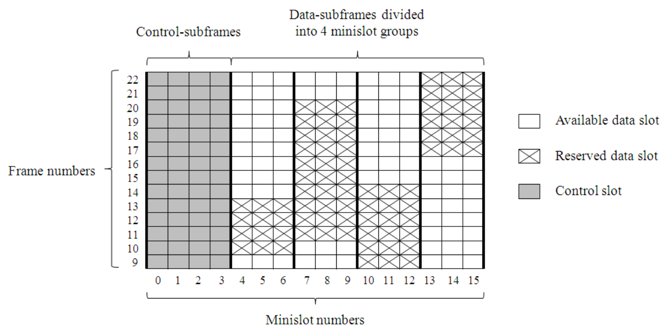

Figure 4: Data-slot Reservation in the Sliced-GM-RBDS Algorithm

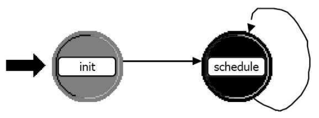

Figure 5: Scheduler Process Model

initialization takes place only once when the scheduler is created by the main processor at the onset of the simulation. The tasks performed in state *schedule* take place every time the scheduler is invoked by the main processor. The main processor invokes the scheduler every time a scheduling packet is transmitted or received in order to generate and process MSH-DSCH messages respectively. The contents of the messages are generated and processed using the Sliced-GM-RBDS algorithm.

When the scheduler is invoked for generating the contents of an MSH-DSCH that is going to be transmitted, it generates Grant, Request, and Availability IEs[7].

[7]The Sliced-GM-RBDS algorithm uses timers for discarding requests that expire.

Grant-IE Generation: For every pending request, the set of overlapping requests is found. If the request is the longest in this set, it is granted and all the other requests in the set are discarded (i.e., they are not longer eligible for any grant). The grant for each of the granted requests is assigned a set of data slots that is determined as follows. The set's data-slot group is the one specified in the request's Availability IE. The size of the set's frame range is the one specified in the Request IE, and the number of the set's start frame is the maximum among the frame numbers of the following frames: the Availability IE's start frame, the frame following the last frame granted in the data-slot group indicated by the Availability IE, and the frame that is a pre-specified number of frames ahead of the current one. In this way, it is assured that the grant does not overlap any of the grants heard by the node and the 1-hop neighbor that made the request. Also, it is assured that the 1-hop neighbor is able to confirm the grant before the grant's start frame becomes the current frame.

Request and Availability IEs Generation: For every outgoing link, if there is a pre-specified minimum of unscheduled data packets in its input-queue, a request is made for a set of data slots whose length is the longest that can be entirely covered with unscheduled data packets. The availability of each request is such that

- It does not overlap any of the grants heard by the node.
- If there is a data-slot group which has not been included in any of the availabilities heard by the node or any of the availabilities that were previously calculated for other outgoing links, that data-slot group is assigned to the availability being calculated, and the availability's start-frame number is the maximum among the frame numbers of the following frames: the frame following the last frame granted in the data-slot group, the frame when the next MSH-DSCH packet is transmitted by this node.

- If there is not such data-slot group, the data-slot group that has been assigned to the lowest number of availabilities among all the availabilities heard by the node and the availabilities previously calculated for other outgoing links is assigned to the availability, and the availability's start-frame number is the maximum among the frame numbers of the following frames: the frame following the last frame granted across all the data-slot groups, the frame when the next MSH-DSCH packet is transmitted.
- The number of the last frame of the availability's frame range is always made equal to infinity. In this way, the node that grants the request is able to use for the grant any of the frames following the availability's start frame. This is possible because none of the data slots that belong to the availability's data-slot group and to the frames that follow the availability's start frame has been granted according to the previous two items.

When the scheduler is invoked for processing the Grant IEs of an MSH-DSCH message that has been received, it performs the following actions. For every Grant IE in the MSH-DSCH message, the scheduler checks whether the grant is directed to the node. If it is, the scheduler looks for the ID of the outgoing link that connects to the 1-hop neighbor that sent the grant. Then, it schedules data packets that are in the input queue assigned to the outgoing link. These packets are scheduled in the set of data slots specified in the Grant IE. Finally, it generates a Grant IE for confirming the received grant. This grant confirmation is a copy of the received Grant IE.

5. OPNET IMPLEMENTATION

The OPNET implementation of the WiMAX-RBDS-Sim architecture is shown in Figure 6. This implementation matches the WiMAX-RBDS-Sim architecture shown in Figure 2. The physical layer consists of a radio-receiver module and a radio-transmitter module. Each module is configured with 10 radio channels and any of the configuration modes (i.e., bandwidth, forward error correction, and modulation schemes) specified in the standard [16]. Also, each module is connected to 10 packet streams, which are OPNET elements used for communicating packets across modules. There is 1 packet stream per channel. The radio-receiver module connects to 10 packet-classifier modules through the packet streams, and each packet-classifier module connects to the main-processing module through 3 packet streams. Each of these packet streams carries packets of only one type (i.e., link-establishment, scheduling, or data packet). The main-processing module connects to the data-packet-source-sink module of the logical-link layer through 2 packet streams. The outgoing packet stream carries received data packets and notifications, in the form of packets, of new outgoing-link establishments. The incoming packet stream carries data packets generated by the data-packet-source-sink module. The main-processing module connects to 10 output-queue-channel-classifier modules with 1 packet stream per module. There is 1 of each of these modules per channel, and they are used for temporarily storing any type of scheduled packets. Finally, each of the output-queue-channel-classifier modules connects to the radio-transmitter module through 1 packet stream that carries scheduled packets of any type when they need to be transmitted.

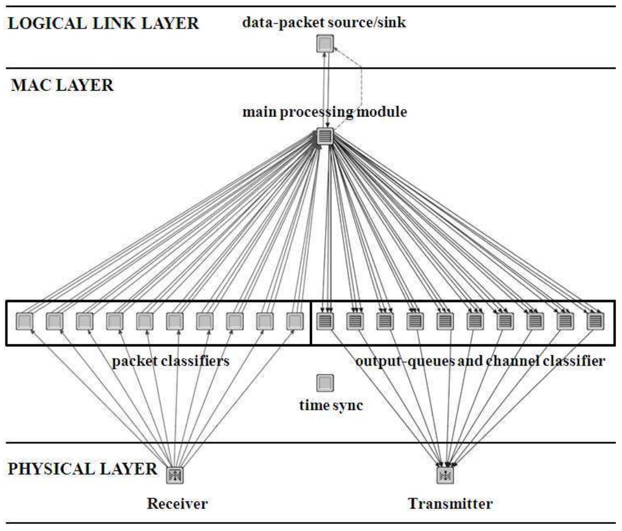

Figure 6: OPNET Implementation of the WiMAX-RBDS-Sim Architecture

The main-processing module implements the main processor, link-establishers, and schedulers as shown in Figure 2. Also, it implements the input-queues in which data-packets generated at the logical-link layer are stored temporarily while they are scheduled.

The time-sync module synchronizes the main-processing and output-queue-channel-classifier modules with the frame structure shown in Figure 1. It interrupts these modules at the onset of every control and data slot and informs them of the current-time information such as the current frame number, current type of subframe (i.e., control or data subframe), and the current control or data slot number.

6. SIMULATION RESULTS

An IEEE 802.16 mesh network with a grid topology of 7×8 nodes was simulated[8]. The number of slots in the control subframe was set to 9. The interference model considered was the *protocol model* [13] in which a reception is successful if the transmitter is closer to the receiver than any other node that transmits simultaneously. The traffic load was varied from 0 to 256 pk/s while accounting for the number of interfering links of every link. For example, if a link had 1 or 3 interfering links and the traffic load was set at 256 pk/s, the traffic generated for that link was 128 pk/s or 64 pk/s respectively (i.e., $\frac{256}{1+1}$ pk/s and $\frac{256}{1+3}$ pk/s). A total of 3000 frames were simulated.

The results obtained with WiMAX-RBDS-Sim are shown[9] in Figures 7, 8, 9, 10, 11, and 12.

Figure 7 shows the histogram of the control-subframe access delay. This is the delay a node experiences to gain access to the control subframe. When an IEEE 802.16 mesh node accesses the control subframe (i.e., it transmits either a link-establishment or scheduling packet), it competes for a future control slot to transmit its next link-establishment

[8]The size of this network is representative of real WMNs [5].
[9]The WiMAX-RBDS-Sim framework also allows the analysis of the data-packet-delivery delay, throughput, and collision probability in the control and data subframes, but these results are not discussed here for the sake of brevity.

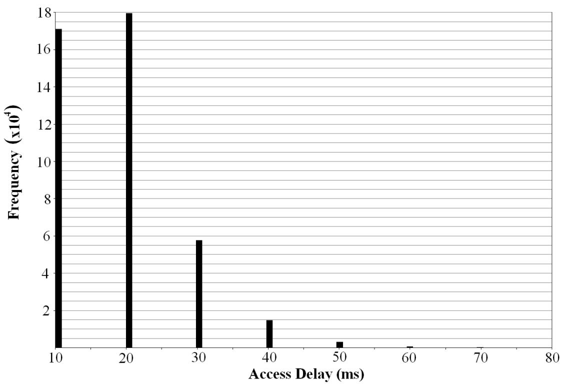

Figure 7: Control-Subframe Access Delay Histogram

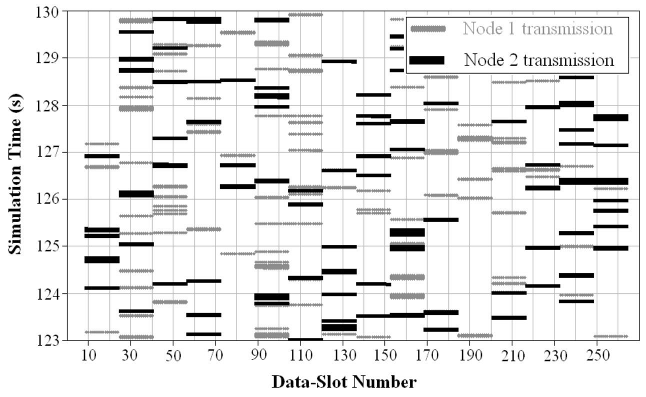

Figure 8: Data-Slot Reservation of two 1-hop Neighbors in a Network with Grid Topology of 7×8 Nodes

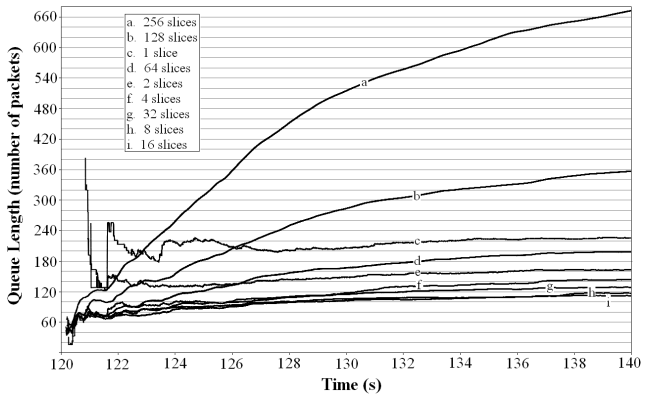

Figure 9: Average Output-Queue Length for Increasing Number of Slices (Traffic Load = 64 pk/s)

or scheduling packet. The time between two consecutive accesses to the control subframe is the access delay, and it directly affects the time required to finish any handshake in control subframes. Therefore, the scheduling-handshake delay, which is the time required to perform the three-way handshake specified in Section 3.1 for negotiating a data-slot reservation, is affected by the shape of such histogram (i.e., the distribution of the control-subframe access delay). This shape can be modified by dynamically controlling the parameters of the election algorithm, which is the technique used in [33, 28, 3, 17, 14, 32, 31, 20] for the implementation of different QoS, collision avoidance, and congestion control schemes. Therefore, WiMAX-RBDS-Sim is a tool that can be used for the evaluation of such schemes by facilitating a simulation framework in which they can be readily integrated.

Figure 8 shows the data-slot reservations of two 1-hop neighbors during an interval of 7 seconds. These reservations are calculated by the Sliced-GM-RBDS Algorithm so that overlaps between them are avoided. The algorithm was configured with 16 slices[10]. The blank spaces correspond to sets of data slots that were reserved for other nodes in the network. Ideally, there should not be any blank spaces nor overlaps when all the nodes that interfere with the two 1-hop neighbors are considered. Figures 8 is an example of the type of results that can be obtained for RBDS algorithms. In this example, the Sliced-GM-RBDS algorithm is being considered. However, other RBDS algorithms, such as the ones presented in [9, 29, 19, 12, 18, 8, 21, 34], can be integrated as scheduler processes (see Figure 2) and evaluated.

Figure 9 shows the average length across the network of the output queues for a traffic load of 64 pk/s when the number of slices increases. The average output-queue lengths clearly depend on the number of slices. Therefore, the capacity of the network can be increased by configuring it with the optimal (i.e., minimum average output-queue length) number of slices. When the data subframes are not partitioned (i.e., when there is only 1 slice), the average output-queue length is not minimized. This is the configuration used in the GM-RBDS algorithm presented in [27]. Therefore, the Sliced-GM-RBDS algorithm is an improved version of the GM-RBDS algorithm in terms of network capacity. The Sliced-GM-RBDS algorithm is able to maintain lower queue lengths, and as a consequence lower data-packet-delivery delays, than the GM-RBDS algorithm does. Specifically, the Sliced-GM-RBDS minimizes the average output-queue length when the number of slices is 16. Figure 10 shows the average output-queue length for different traffic loads when the number of slices is fixed at 16. The output-queue lengths are finite as long as the traffic load is below 128 pk/s. For higher traffic loads, the output-queue lengths increase indefinitely with time.

Figure 11 shows the instantaneous length of the input and output queues in the network when the traffic load is maximum (i.e., 256 pk/s). As expected from the results in Figure 10, the output queues are unstable (i.e., they increase indefinitely with time). However, the input queues are stable (i.e., they always return to the empty state at some points in time). Therefore, given that the input queues are always stable and the output queues are stable for traffic loads lower than 128 pk/s, the Sliced-GM-RBDS algorithm guarantees the stability of the network (i.e., all the queues in the network are stable) for the simulation scenario considered when the traffic load does not exceed 128 pk/s.

As a final example of the results and analysis that can be performed with WiMAX-RBDS-Sim, the histogram of the link-establishment delay is shown in Figure 12. This delay is

[10]The data-slot reservation example shown in Figure 4 considers only 4 slices.

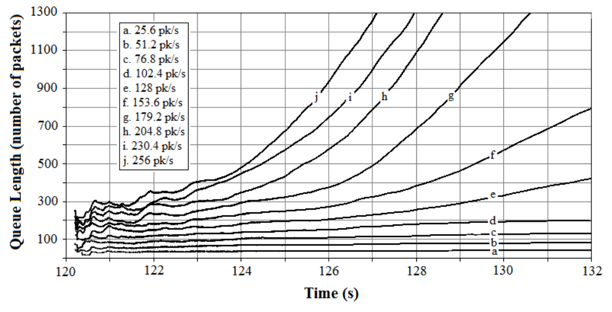

Figure 10: Average Output-Queue Length for Increasing Traffic Loads (Number of Slices = 16)

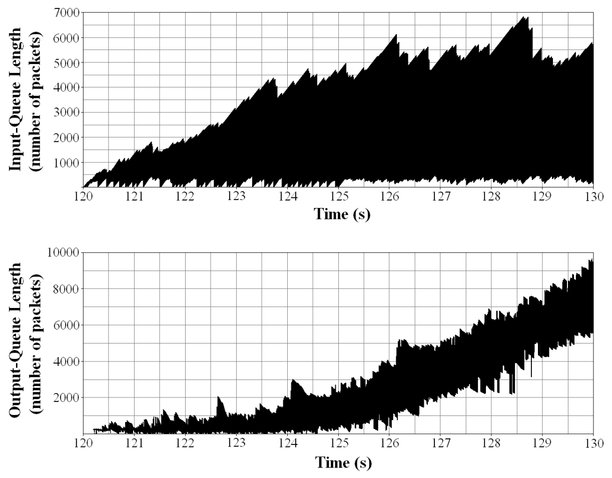

Figure 11: Input-Queue and Output-Queue Length Comparison for Maximum Traffic Load

the time required for establishing the incoming and outgoing links between 1-hop neighbors with the three-way handshake specified in Section 3.2. There is a total of 97 incoming-outgoing link pairs in the 7 × 8 grid topology. The results show that the link-establishment algorithm establishes a bidirectional link in no more than 2.6 s. This result is also affected by the distribution of the control-subframe access delay (see Figure 7) given that the link-establishment handshake is performed with the exchange of packets in the control subframe. A scheme for reducing the link-establishment delay was proposed in [30].

7. PERFORMANCE EVALUATION

The performance of WiMAX-RBDS-Sim in terms of simulation speed and memory usage was evaluated for different network setups. The simulated network setups included 4 different grid topologies with increasing number of nodes (i.e., 4 × 8, 5 × 8, 6 × 8, and 7 × 8) with the same configuration used in Section 6. The simulations were performed in the sequential mode in which there is only one thread of execution[11]. Figures 13 and 14 show that the simulation speed decreases and the memory usage increases approximately linearly when the number of nodes increases.

[11]The platform used in the simulation included a processor Intel Core Duo at 3 GHz and 3 GB of RAM at 3 GHz.

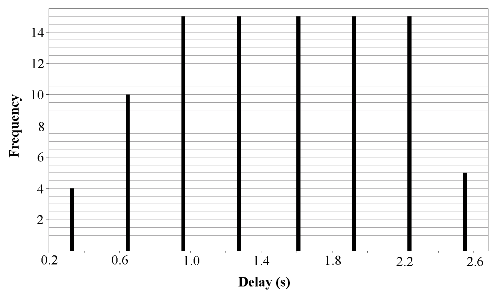

Figure 12: Link-Establishment Delay Histogram

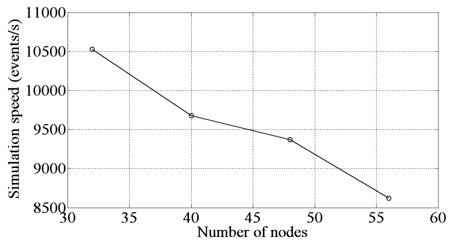

Figure 13: Simulation Speed

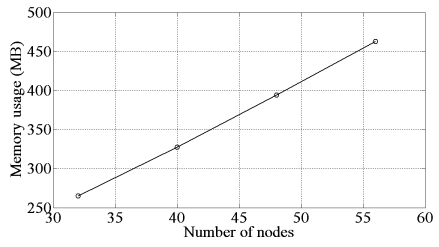

Figure 14: Memory Usage

8. CONCLUSIONS

A new simulation model for IEEE 802.16 mesh networks with distributed scheduling (i.e., WiMAX-RBDS-Sim) was developed in OPNET. It provides interfaces for the implementation and evaluation of distributed scheduling policies and link-establishment algorithms. The policies may be based on the dynamic variation of the election-algorithm parameters or on the reservation of future data slots.

As an example of the use of WiMAX-RBDS-Sim, a new RBDS policy (i.e., Sliced-GM-RBDS) was implemented and evaluated with it. This policy is based on the concepts of input and output queues. The results showed that the Sliced-GM-RBDS policy outperformed the GM-RBDS policy in terms of capacity since the system of queues was stable for higher traffic loads when the Sliced-GM-RBDS was used. Also, a link-establishment algorithm was implemented and integrated with WiMAX-RBDS-Sim, and its performance was evaluated in terms of delay.

Finally, the performance of WiMAX-RBDS-Sim was evaluated in terms of simulation speed and memory usage for an increasing number of nodes.

9. REFERENCES

[1] Network simulator 2. Network Simulation Software.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8679
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8679

[2] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487, March 2005.

[3] N. Bayer, B. Xu, V. Rakocevic, and J. Habermann. Improving the performance of the distributed scheduler in ieee 802.16 mesh networks. In *VTC2007-Spring*, pages 1193–1197, April 2007.

[4] A. Belghith and L. Nuaymi. Design and implementation of a qos-included wimax module for ns-2 simulator. In *Simutools'08*, pages 1–8, March 2008.

[5] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Mobicom'05*, pages 31–42, August 2005.

[6] M. Cao, W. Ma, Q. Zhang, and X. Wang. Analysis of ieee 802.16 mesh mode scheduler performance. *IEEE Trans. Wireless Commun.*, 6(4):1455–1464, April 2007.

[7] L. W. Chen, Y. C. Tseng, Y. C. Wang, D. W. Wang, and J. J. Wu. Exploiting spectral reuse in routing, resource allocation, and scheduling for ieee 802.16 mesh networks. *IEEE Trans. Veh. Technol.*, 58(1):301–313, January 2009.

[8] S.-M. Cheng, P. Lin, D.-W. Huang, and S.-R. Yang. A study on distributed/centralized scheduling for wireless mesh network. In *IWCMC'06*, pages 599–604, July 2006.

[9] C. Cicconetti, I. F. Akyildiz, and L. Lenzini. Feba: A bandwidth allocation algorithm for service differentiation in ieee 802.16 mesh networks. *IEEE/ACM Trans. Netw.*, 17(3):884–897, June 2009.

[10] C. Cicconetti, I. F. Akyildiz, and L. Lenzini. Wimsh: a simple and efficient tool for simulating ieee 802.16 wireless mesh networks in ns-2. In *Simutools'09*, pages 1–10, March 2009.

[11] C. Cicconetti, A. Erta, L. Lenzini, and E. Mingozzi. Performance evaluation of the mesh election procedure of ieee 802.16/wimax. In *MSWiM'07*, pages 323–327, October 2007.

[12] C. Cicconetti, V. Gardellin, L. Lenzini, and E. Mingozzi. End-to-end bandwidth reservation in ieee 802.16 mesh networks. In *MASS'07*, pages 1–6, October 2007.

[13] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Inf. Theory*, 46(2):388–404, March 2000.

[14] H. Hu, Y. Zhang, and H.-H. Chen. An effective qos differentiation scheme for wireless mesh networks. *IEEE Network*, 22(1):66–73, January 2008.

[15] S.-M. Huang, Y.-C. Sung, S.-Y. Wang, and Y.-B. Lin. Nctuns simulation tool for wimax modeling. In *WICON'07*, pages 1–6, October 2007.

[16] IEEE. Standard for local and metropolitan area networks - part: 16 air interface for fixed broadband wireless access systems. IEEE Std 802.16-2004.

[17] B. C. Kim, D. G. Kwak, H. S. Lee, and J. S. Ma. An adaptive holdoff algorithm based on node state for ieee 802.16 mesh mode with coordinated distributed scheduling. In *PIMRC'08*, pages 1–5, September 2008.

[18] P.-Y. Kong, H. Wang, Y. Ge, C.-W. Ang, J. Pathmasuntharam, W. Su, M.-T. Zhou, and H. Harada. Distributed adaptive time slot allocation for wimax based maritime wireless mesh networks. In *WCNC'09*, pages 1–6, April 2009.

[19] M. S. Kuran, G. Gur, T. Tugcu, and F. Alagoz. Cross-layer routing-scheduling in ieee 802.16 mesh networks. In *Mobilware'08*, pages 1–6, February 2008.

[20] Y. Li, D. Wei, H. Zhuang, H. Wang, and P. Wang. A new congestion control method for ieee 802.16 mesh mode. In *ICCSN'09*, pages 726–730, February 2009.

[21] H.-M. Lin, W.-E. Chen, and H.-C. Chao. A dynamic minislot allocation scheme based on ieee 802.16 mesh mode. In *FGCN'08*, pages 288–293, December 2008.

[22] S. Liu, S. Feng, W. Ye, , and H. Zhuang. Slot allocation algorithms in centralized scheduling scheme for ieee 802.16 based wireless mesh networks. *Computer Commun.*, 32(5):943–953, March 2009.

[23] I. C. Msadaa, F. Filali, and F. Kamoun. An 802.16 model for ns2 simulator with an integrated qos architecture. In *Simutools'08*, pages 1–10, March 2008.

[24] A. Sayenko, O. Alanen, H. Martikainen, V. Tykhomyrov, O. Puchko, and T. Hamalainen. Winse: Wimax ns-2 extension. In *Simutools'09*, pages 1–10, March 2009.

[25] G. Sharma, C. Joo, and N. B. Shroff. Distributed scheduling schemes for throughput guarantees in wireless networks. In *Allerton'06*, pages 1–10, September 2006.

[26] O. Technologies. Opnet modeler. Network Simulation Software.

[27] G. Vejarano and J. McNair. Reservation-based distributed scheduling in wireless networks. In *Submitted to WoWMoM'10*, pages 1–9.

[28] S.-Y. Wang, C.-C. Lin, H.-W. Chu, T.-W. Hsu, and K.-H. Fang. Improving the performances of distributed coordinated scheduling in ieee 802.16 mesh networks. *IEEE Trans. Veh. Technol.*, 57(4):2531–2547, July 2008.

[29] S.-Y. Wang, C.-C. Lin, and K.-H. Fang. Improving the data scheduling efficiency of the ieee 802.16(d) mesh network. In *Globecom'08*, pages 1–5, December 2008.

[30] S.-Y. Wang, C.-C. Lin, K.-H. Fang, and T.-W. Hsu. Facilitating the network entry and link establishment processes of ieee 802.16 mesh networks. In *WCNC'07*, pages 1842–1847, March 2007.

[31] Y. Zhang, J. Zheng, and W. Li. A simple and effective qos differentiation scheme in ieee 802.16 wimax mesh networking. In *WCNC'07*, pages 3216–3220, March 2007.

[32] Y. Zhang, M. Zhou, S. Xiao, and M. Fujise. An effective qos scheme in wimax mesh networking for maritime its. In *Proceedings of the 6th International Conference on ITS Telecommunications*, pages 612–616, June 2006.

[33] H. Zhu, Y. Tang, and I. Chlamtac. Unified collision-free coordinated distributed scheduling (cf-cds) in ieee 802.16 mesh networks. *IEEE Trans. Wireless Commun.*, 7(10):3889–3903, October 2008.

[34] A. Ziller, P. S. Mogre, M. Hollick, and C. Schwingenschlogl. Reliable broadcast mechanism for the ieee 802.16 mesh extension. In *Globecom Workshops*, pages 1–5, December 2008.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8679
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8679