Technical Report (Rev. 03/11/2010): OPNET Modeling and Evaluation of IEEE 802.16 Wireless Mesh Networks with Distributed Scheduling

Gustavo Vejarano, Graduate Student, PhD.
Advisor: Janise McNair

Wireless and Mobile Systems Laboratory
URL: http://www.wam.ece.ufl.edu/
Department of Electrical and Computer Engineering
University of Florida

Summary

An 802.16 mesh mode with distributed scheduling wireless mesh network (WMN) simulation model is developed for the evaluation of scheduling algorithms at the medium access control (MAC) layer. The simulation model is developed under the OPNET event-driven simulation environment. It provides interfaces for the integration of scheduling and link-establishment algorithms. It can simulate up to ten channels, and it assumes omni-directional antenna beam patterns, the protocol interference model, and perfect synchronization at the carrier, symbol, bit, and frame levels.

2. IEEE 802.16 Mesh Mode Review

The 802.16 standard [1] specifies the physical and MAC layers for a fixed and mobile Broadband Wireless Access System. The MAC layer is divided into three sublayers: Service-Specific Convergence Sublayer, MAC Common Part Sublayer, and Security Sublayer. Here, we focus on the MAC Common Part Sublayer, which will be called MAC layer, when it operates in the mesh mode. The MAC layer performs tasks for system access, bandwidth allocation, connection establishment, and connection maintenance.

A WMN can be developed based on the mesh mode specified in the standard. In an 802.16 WMN there are two types of nodes: Base Stations (BS) and Subscriber Stations (SS). Links can be established between any of these nodes, and the information can be transmitted on a hop-by-hop basis between the nodes. The 802.16 WMN is connected to external networks such as the Internet through BSs. In this way, Internet traffic can be carried to an SS that is several hops away from the BS on a multi-hop basis through intermediate SSs. The frequencies used in the network are below 11GHz so that Line of Sight (LOS) is not necessary between 1-hop neighbors. Depending on the licensing-type of the band (i.e., licensed bands, unlicensed bands), the air interface designated to the WMN can be WirelessMAN-OFDM and WirelessHUMAN for the licensed and unlicensed cases respectively. These two interfaces are specified in the standard [1].

The 802.16 mesh mode supports nodes with omni-directional and directional antennas. The maximum number of antenna directions is eight. It also supports multi-channel nodes and multi-rate links. The links are bidirectional, and the duplexing scheme is TDD. 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 the node, and the node ID identifies the node from which the link originates. The system time synchronization is achieved by providing some of the nodes with a time reference such as GPS and by synchronizing the others to these references, which are distributed across the network.

2.1. Frame Structure

The system access follows a frame-based approach which is shown in Fig. 1.

Each channel is divided in time into series of frames. In Fig. 1, each of these series consists of eight frames (i.e., from frame n to frame n+7). The number of frames in a series is defined during the network creation process. A frame is further divided into two subframes: control subframe, data subframe. The control subframes are used for carrying the information necessary to control the system access, bandwidth allocations, connection establishments, and connection maintenance. The data subframes are used for carrying upper layers' packets (i.e., data packets). The control and data subframes are divided into minislots (see Fig. 1). These are used by the nodes for transmitting control and data packets.

The control subframe of the first frame of a series (e.g., frame n in Fig. 1) is used for the transmission of control packets that carry network-configuration messages. The control subframes of the following frames of the series (e.g., frames n+1, n+2, …, n+7 in Fig. 1) are used for the transmission of control packets that carry scheduling packets.

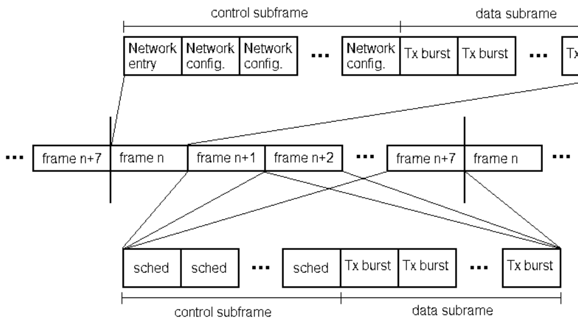

Fig. 1. 802.16 mesh mode frame structure

2.2. Network Configuration Control Subframe

The messages transmitted in the control subframe of the first frame of a series are dedicated exclusively to the connection establishment and connection maintenance functions.

Before joining the WMN, new nodes first need to listen to the messages transmitted by the nodes of the WMN in the network-configuration minislots. These messages contain the network configuration information necessary to join the network. Then, the new node selects a sponsoring node that will possibly establish a link with it so that it can start the required process for obtaining permission to join the network. A message to the sponsoring node is then sent during the network-entry minislot by the new node (see Fig. 1). This burst is exclusively used for the transmission of network-entry messages. In the message, the new node requests the creation of a link. Finally, if the sponsoring node decides to create the link, the new node communicates with a BS through the sponsoring node, and the BS gives permission and assigns a node ID to the new node.

The messages periodically transmitted by the nodes in the network-configuration minislots also carry information about the links of their 1-hop neighbors. Each node periodically broadcasts in one of the bursts the following information about each of its 1-hop neighbors.

- Link quality: It is a reliability measure of the incoming link that connects to the 1-hop neighbor.
- Burst profile: It is the set of parameters that defines the characteristics (e.g., modulation type, forward error correction type, preamble length) used for transmissions over the incoming link.
- 1-hop neighbor's transmission power.
- 1-hop neighbor's antenna direction for communicating with the reporting node.
- 1-hop neighbor's node ID, Holdoff Exponent (HE), and Next Transmission Time (NTx). The node ID identifies the node in the network, and the HE and NTx determine when the 1-hop neighbor accesses the control subframes.

Also, in the messages, the same information about node transmitting the messages (i.e., link quality, burst profile, transmit power, etc.) is sent to its 1-hop neighbors along with the number of hops between the node transmitting the messages and the closest node with a time reference (e.g., a node with GPS capability).

The messages for the connection establishment and connection maintenance are MSH-NENT (i.e., mesh network entry message) and MSH-NCFG (i.e., mesh network configuration message) respectively. These two messages are periodically broadcast in all of the available channels in the WMN to make sure that any new coming node has access to the network configuration information.

In the simulation model, only the MSH-NCFG messages are implemented. With these messages, the nodes establish links with their 1-hop neighbors before exchanging any data packets. The network entry process is omitted in the simulation. All the nodes are given their IDs and network configuration parameters such as the number of channels and bandwidth through the simulation parameters.

2.3. Scheduling Control Subframe

Two different scheduling mechanisms are specified for the scheduling of minislots in data subframes. These mechanisms are the centralized and distributed scheduling, and they are both based on the exchange of control packets in control subframes. The format and content of the scheduling messages carried by control packets depend on the scheduling mechanism used in the WMN. If the two scheduling mechanisms are used simultaneously in the network, the scheduling control subframe is divided into two parts. In the first part, the minislots are used to broadcast centralized scheduling messages, and in the second part, the minislots are used to broadcast distributed scheduling messages.

In centralized scheduling, the BS receives the resource requests of all the nodes that are within a specific hop distance from it and schedules the transmissions of each of these nodes according to a given scheduling policy in a tree-based topology. The BS is the root of the tree.

In distributed scheduling, each node periodically broadcasts lists of available minislots (i.e., minislots in data subframes that have not been reserved for future data packet transmissions in interfering links) to the nodes in its 1-hop or 2-hop neighborhood depending on the network configuration and schedules its data-packet transmissions based on the lists of available minislots it receives.

The access scheme for transmitting control packets in the minislots of control subframes is the same for both the centralized and distributed scheduling. This scheme is implemented by the election algorithm.

The simulation model considers the distributed scheduling only. There is only one message defined in the standard [1] for this scheduling; it is MSH-DSCH (i.e., mesh-distributed schedule). Therefore, in the simulation model, the nodes transmit MSH-NCFG messages in network-configuration control subframes and MSH-DSCH messages in scheduling control subframes. The MSH-NCFG messages are used for establishing links between the nodes, and the MSH-DSCH messages are used for scheduling data-packet transmissions in data subframes.

2.4. Multiple access scheme for the control subframe

The objective of the multiple access scheme for the control subframe is to allow simultaneous transmissions of packets in the minislots of control subframes while avoiding collisions. This is shown graphically in the examples in Fig. 2 and Fig. 3.

It is assumed that each of the nodes in the two WMNs (i.e., WMN of Fig. 2 and WMN of Fig. 3) is able to establish links with its six closest neighbors. Therefore, these nodes are the 1-hop neighbors of the node, and they are within the transmission range of their common neighbor. The transmission range is assumed to be the same for all the nodes. The arrows represent simultaneous transmissions of control packets during one minislot at the same channel.

In Fig. 2, the multiple access scheme is not being used. In fact, no scheme is assumed for this network. The nodes are free to transmit whenever they need to (i.e., slotted Aloha). Therefore, the hidden-terminal problem is present in the network, and the exposed-terminal problem is not. Transmissions t2 and t4 fail because they collide with t3 due to the hidden-terminal n3 (i.e., n1 and n2 are out of n3's coverage, so they cannot sense n3's transmissions to make sure that their packets can be received by the nodes inside n3's coverage). Transmission t6 is successful because n4 transmits its packet even though it is detecting n2's transmission.

In Fig. 3, the multiple access scheme implemented in the election algorithm is being used. This scheme guarantees that when a node transmits a control packet, all the nodes in the node’s 2-hop neighborhood stay silent (i.e., they do not perform any transmissions). Given that the control packets carry information that is relevant to all 1-hop neighbors, the transmissions are done on a point-to-multipoint basis (i.e., the control packets are broadcast to all 1-hop neighbors). By silencing the 2-hop neighborhood, the election algorithm avoids control-packet collisions. For example, n3's transmission silences n1, n2, and n4, so the transmissions of these nodes do not collide with n3's transmission as it occurs in Fig. 2. The nodes out of n3's 2-hop neighborhood (A1) can transmit without colliding with n3’s transmission. For example, n5 broadcasts a control packet to its 1-hop neighbors without colliding with n3's transmission.

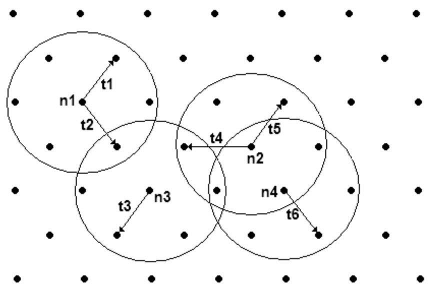

Fig. 2. Packet transmission in the control subframe with no multiple access scheme

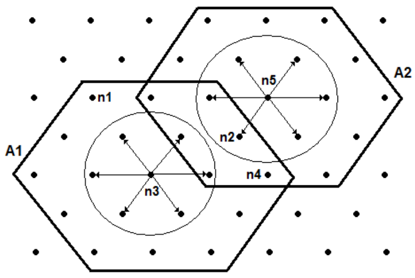

Fig. 3. Packet transmission based on the 802.16 mesh mode control subframe multiple access scheme

The election algorithm is based on the XmtHoldoffExponent (HE) and NextXmtMx (NTx) parameters. There is one instance of the election algorithm per node which performs the following operations. Each time a node[1] broadcasts a control packet, it calculates the schedule of the next control packet it transmits. This packet must be scheduled at least 2(HE+4) minislots away from the current control-packet transmission. This period of time is called Holdoff Time. In order to schedule the next control packet, the node establishes an eligibility interval that starts at the end of the Holdoff Time. The control-packet can be scheduled at any of the minislots that are within the eligibility interval only. The number of minislots in the interval is determined by HE and NTx. The node lists all its 2-hop neighbors whose eligibility interval overlaps with its interval, and from this set of nodes, it selects one of them as the winner. If the node is the winner, it schedules the next control-packet transmission at the first minislot following the end of the Holdoff Time. If the node loses, it delays its eligibility interval by one minislot and repeats the procedure (i.e., recalculates the set of nodes whose eligibility intervals overlap with its eligibility interval and finds the winner from such set). This procedure is repeated until the node is the winner.

2.5. Data transmission scheduling and QoS

The scheduling policy for accessing minislots in data subframes is not specified in the standard. The standard only defines the MSH-DSCH message for the exchange of scheduling information 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.

- Scheduling IE: This IE contains the node's and 1-hop neighbors' IDs, HEs, and NTxs. This information is used by the nodes for accessing the control subframe using the election algorithm.
- 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 minislots per data subframe, and number of requested data subframes. The number of data subframes may be infinity so that streams of information can be transmitted in the link.
- Availability IE: A node notifies its 1-hop neighbors of the minislots it has available for reservation. This information includes the start frame, number of frames, start minislot, number of minislots, direction (i.e., the minislots may be available for transmission, reception, or both of them), and the channels the available minislots belong to.
- Grant IE: This IE includes the same parameters specified for the Availability IE. However, these are used for assigning minislots to a node that had previously requested them.

The scheduling procedure follows a three-way handshake. First, a node sends an MSH-DSCH message to one of its 1-hop neighbors to request minislots in data subframes for the transmission of data packets. This node also includes in the message the set of minislots 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 minislots that satisfies the availability of minislots at both nodes. Finally, the first node confirms the reservation of such set of minislots 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 minislot reservation so that the minislots in the grant become unavailable for them.

2.6. Link establishment

The nodes transmit MSH-NCFG messages periodically. Therefore, a node discovers a 1-hop neighbor when it receives the first MSH-NCFG message transmitted by the neighbor. In order to exchange data packets in both directions, the two 1-hop neighbors need to establish two links between them (i.e., one link in each direction). This is achieved by means of a three-way handshake performed with MSH-NCFG messages. 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. This information is organized as follows.

- Link-Establishment IE: It has an action-code field for indicating whether it is challenge, challenge-response, accept, or reject, a link ID field, and two neighbor authentication values for security purposes. In the simulation model, the authentication values are used only for identifying the node that the Link-Establishment IE is directed to.

3. System Model

[1] Here we refer to the node as the entity performing the operations, but it is the node's election-algorithm instance that actually performs the operations.

The system model is based on the node model shown in Fig. 4. The node model can be instantiated as a BS or SS by setting the node object parameters accordingly. It supports up to ten radio channels. An 802.16 mesh network with distributed scheduling can be modeled by using several nodes and placing them within a spatial area such that they are capable of establishing links between them. For example, several nodes can be uniformly distributed as shown in Fig. 3.

The node model was developed with OPNET Modeler [2], and it consists of the following modules: 1 radio transmitter, 1 radio receiver, 11 queues, and 12 processors[2].

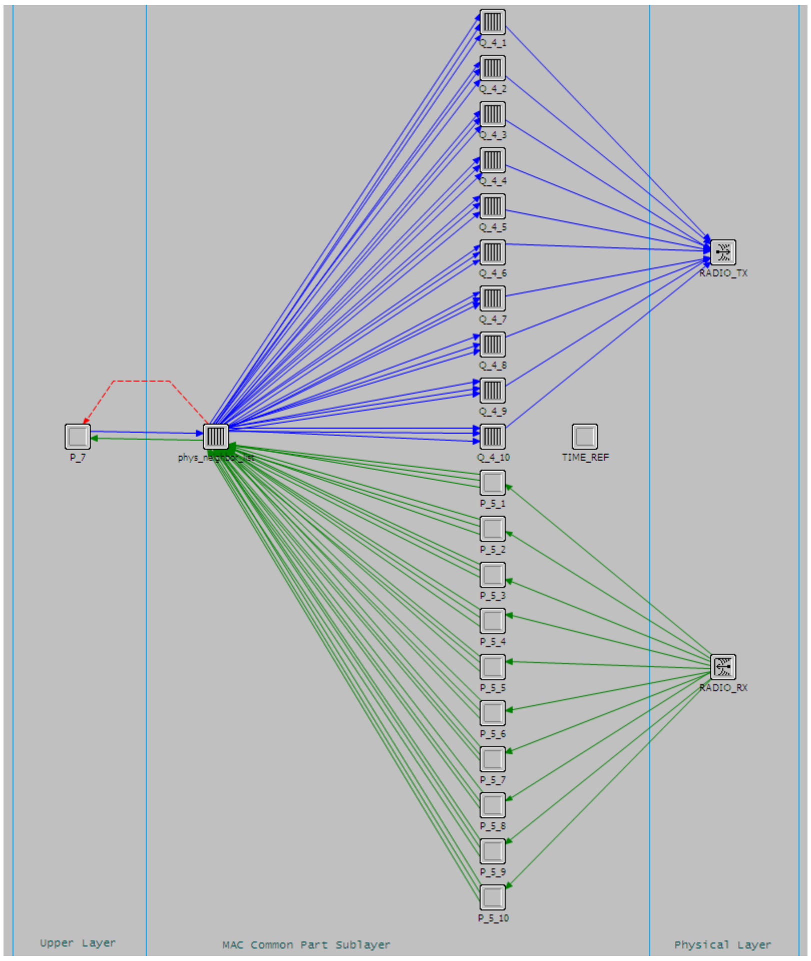

Fig. 4. 802.16 mesh mode node model

Queue modules Q_4_1, Q_4_2, …, Q_4_10 store temporally the packets (i.e., network configuration, scheduling, and data packets) to be sent to the radio-transmitter module for transmission at each of the ten channels according to the packet's schedules.

Processor modules P_5_1, P_5_2, …, P_5_10 classify the packets received through each of the channels into three categories: network configuration (i.e., MSH-NCFG), distributed scheduling (i.e., MSH-DSCH), and data, and forwards them to the queue module phys_neighbor_list.

The queue module phys_neighbor_list controls the generation and exchange of control packets that carry MSH-NCFG and MSH-DSCH messages in order to perform the link-establishment and distributed-scheduling procedures. Therefore, phys_neighbor_list is the source and sink of network-configuration (i.e., MSH-NCFG) and scheduling (i.e., MSH-DSCH) messages. It also serves as the interface to the upper layer through which data packets are sent back and forth between the upper-layer and the MAC layer, and it notifies the upper-layer of new link establishments with 1-hop neighbors. These notifications are directed to the data-packet source-sink process located at the upper layer (i.e., processor module P_7). The physical-layer configuration is performed by phys_neighbor_list too. It modifies the radio-transmitter's and radio-receiver's attributes according to the

[2] We refer to the modules in Fig. 4 using the following OPNET terminology. A processor module is a module capable of processing packets without performing any packet buffering. A queue module is a processor module with packet-buffering capability. Radio-transmitter and radio-receiver modules are capable of transmitting and receiving data packets only according to a given physical channel model, antenna, and transmission power.

simulation attributes[3]. These include modulation scheme, coding rate, and transmission power according to the standard [1].

Processor module P_7 is the node's source and sink of data packets. P_7 receives link-establishment notifications in the form of packets from phys_neighbor_list whenever new links are created with 1-hop neighbors. Once the notification is processed, P_7 updates the list of outgoing links. For each of the outgoing links, it generates data packets randomly according to the distribution and rate specified by the node's attributes. These data packets are sent to phys_neighbor_list which stores them temporarily until their schedules are calculated. Also, when phys_neighbor_list receives data packets directed to the node, these packets are sent to P_7, and P_7 finally destroys them.

Processor module TIME_REF synchronizes the node with the frame structure shown in Fig. 1.

The node model is based on the following assumptions: omni-directional antenna beam pattern, perfect synchronization at the carrier, symbol, bit, and frame levels, and static nodes (i.e., the nodes are not mobile)[4]. The model works as follows.

The radio-transmitter and radio-receiver modules perform all the physical-layer processing. The radio-transmitter module has ten input streams[5] (i.e., one input stream per channel). From each of these packet streams, the radio-transmitter module receives at the onset of every minislot the packet to be transmitted if there is such. These packets are transmitted with one of the modulation schemes given in the standard [1] and specified as a simulation attribute throughout the whole duration of the simulation (i.e., the physical-link-transmission rate is fixed). The radio-receiver module passes all the successfully received packets to processors P_5_1, P_5_2, …, P_5_10 through its output streams. There is one output stream per channel.

Each queue module Q_4_1, Q_4_2, …, Q_4_10 checks at the onset of every minislot whether there is any packet scheduled for transmission at that minislot. If there is such packet, the packet is sent to the radio-transmitter module. Processor modules P_5_1, P_5_2, …, P_5_10 classify the successfully received packets and pass them to the phys_neighbor_list module. The phys_neighbor_list module implements the scheduling policy for reserving minislots in data subframes for the transmission of data packets. It receives the data packets generated by P_7, stores temporarily while it calculates the schedules (i.e., channel, frame, and minislot numbers) for the transmission of each of them, and then sends them along with their schedules to queue modules Q_4_1, Q_4_2, …, Q_4_10. It is assumed that the outgoing link through which each packet needs to be transmitted is specified by the upper layer (i.e., routing decisions are made at upper layers). In order to calculate the schedules of data packets, phys_neighbor_list generates MSH_DSCH messages (i.e., scheduling messages) that are exchanged with the node's 1-hop neighbors, calculates the schedules for the control packets that carry the messages using the election algorithm, and sends the control packets along with their schedules to the corresponding queue module Q_4_1, Q_4_2, …, Q_4_10. Also, it processes the MSH-DSCH messages received by the node. In this way, the phys_neighbor_list is able to implement the three-way handshake for the request, grant, and grant confirmation of minislots in data subframes. The phys_neighbor_list module sends packets to queues Q_4_1, Q_4_2, …, Q_4_10 and receives packets from processors P_5_1, P_5_2, …, P_5_10 with three packet streams for each of them. Each packet stream is assigned to one of the three types of packets (i.e., MSH-NCFG, MSH-DSCH, and data). The data packets that are received from processor modules P_5_1, P_5_2, …, P_5_10 and that are directed to the node are forwarded to processor module P_7. The received data packets that are not directed to the node are destroyed. Phys_neighbor_list maintains a list of the nodes in the 2-hop neighborhood called physical-neighbor-list and a list of BSs. There is one of each of these lists per channel. The lists are updated by exchanging MSH_NCFG messages periodically with the node's 1-hop neighbors, and they are used by the election algorithm. Phys_neighbor_list implements the election algorithm for calculating the schedules of all the packets transmitted in control subframes. Process module TIME_REF synchronizes the node with the system time according to the frame structure shown in Fig. 1. It interrupts phys_neighbor_list, Q_4_1, Q_4_2, …, and Q_4_10 at the onset of every minislot and provides them with the current minislot number, frame number, and frame-structure parameters. Finally, process module P_7 models the upper layer. It generates data packets for the node's outgoing links and destroys the data packets received from the node's incoming links.

The data-packet scheduling and transmission processes previously described do not start until all the nodes have established links with their 1-hop neighbors. During the initialization period, which is specified as a simulation attribute, the process modules P_7 at all nodes do not generate any data packets. The nodes transmit packets with MSH-NCFG and MSH-DSCH messages only according to the standard [1]. Whenever a node receives an MSH-NCFG packet from a node whose ID is not in the physical-neighbor-list, a new entry is added to the list for the new 1-hop neighbor. Initially, the link IDs of the incoming and outgoing links have not been assigned (i.e., the links between the new 1-hop neighbors have not been established). Therefore, the

[3] Simulation attributes is the OPNET term for referring to user-specified simulation parameters.
[4] The static-node assumption is based on the fact that node locations in WMNs are carefully planned as in cellular networks.
[5] A stream is an OPNET inter-module communication element. It is used for sending only packets between modules.

phys_neighbor_list modules at the two nodes connected by the links start a link-establishment process each. The two link-establishment processes perform a three-way handshake by means of an MSH-NCFG-message exchange, which results in the assignment of IDs for the two new links (i.e., incoming and outgoing links). At the end of the initialization period, the phys_neighbor_list module at each node checks whether there are any entries in the physical-neighbor-list for 1-hop neighbors with unassigned link IDs. If all links have been assigned IDs, the data-packet generation, scheduling, and transmission processes take place, otherwise the simulation is aborted.

3.1. Process Models

The node model is based on seven process models[6] that specify all the required functionalities to implement the 802.16 mesh mode with distributed scheduling. These process models are: process_model_01, establish_incoming_link_connectivity, MSH_DSCH_scheduler, process_model_04, process_model_05, process_model_07, and time_ref.

3.1.1. Process_model_01

This is the process model used by queue module phys_neighbor_list. It connects to 30 input streams directed from P_5_1, P_5_2, …, and P_5_10, 30 output streams directed to Q_4_1, Q_4_2, …, and Q_4_10, and 1 output and 1 input stream directed to and from P_7 respectively. The sets of 30 input and 30 output streams are divided each into subsets of 3 streams (i.e., 3 streams per channel), and each of the streams in the subsets carries packets with one type of message only: MSH-NCFG, MSH-DSCH, or data. The output stream directed to P_7 carries data packets and link-establishment notifications in the form of packets. The input stream directed from P_7 carries data packets only. Process_model_01 creates link-establishment and scheduler child processes that implement the three-way handshakes for establishing links and scheduling data packets respectively. A link-establishment process exists only during the link-establishment three-way handshake with the 1-hop neighbor it is assigned to. Therefore, link-establishment processes are destroyed once the incoming and outgoing links to the 1-hop neighbor have been assigned IDs. Scheduler processes are always created and destroyed at the beginning and at end of the simulation respectively. There is one scheduler process per channel which performs all the three-way handshakes for scheduling data packets in its channel. The operations performed by process_model_01 are based on the 297 subqueues[7] of the phys_neighbor_list queue module. These subqueues are:

- Physical-neighbor-list subqueues: There are 10 of these subqueues (i.e., one subqueue per channel). For every channel, there is one packet in the subqueue per neighbor in the 2-hop neighborhood, and each packet contains the following information about the neighbor[8]: MAC address, distance in number of hops (i.e., 1-hop or 2-hop neighbor), node ID, HE, NTx, a flag for indicating that the entry has already been reported to the node's 1-hop neighbors since the last time it was updated, the distance in number of hops to the closest node with a time-reference system such as GPS, the process handle of the link-establishment process between the node and the neighbor, a pointer to the buffer used for communicating with the link-establishment process (i.e., OPNET parent-child memory[9] (PCM)), the incoming and outgoing link IDs, frame and minislot numbers of the last MSH-NCFG packet transmission, and frame and minislot numbers of the last MSH-DSCH packet transmission.
- BS-list subqueues: There are 10 of these subqueues (i.e., one subqueue per channel). For every channel, there is one packet in the subqueue per BS in the channel. Each packet contains the node ID of the BS and the distance (i.e., shortest path's length) in number hopes from the node to the BS[10].
- Grant-list subqueue: There are 10 of these subqueues (i.e., one subqueue per channel). For every channel, there is one packet in the subqueue per unexpired interfering grant. A grant interferes if it belongs to a link that interferes with any of the links of the node. A grant consists of the link ID, start-minislot, minislot-range-size, start-frame, persistence, direction, and channel number. The link ID identifies the link the grant belongs to. The start-frame is the first frame with minislots included in the grant. The persistence is the number of consecutive frames that have minislots included in the grant. The frames included in the grant share the same range of granted minislots. This range is specified with the start-minislot, which is the first minislot of the range, and the minislot-range-size, which is the

[6] In OPNET, process models are used to specify the behavior of processor and queue modules. A process model is described in terms of a state transition diagram. At each state, the process performs certain actions, and the process jumps from one state to another state every time it is interrupted by an event such as a packet arrival.
[7] In OPNET, a queue module may have one or more subqueues. Each suqueue is used for buffering packets which are pushed and pulled from the subqueue.
[8] The OPNET-packet-format file for the packets stored in the physical-neighbor-lists is phys_neighbor_list_entry.pk.
[9] In OPNET, a process may create and invoke child processes that perform certain actions for the parent process. The parent and child processes communicate by means of a shared memory that they both write to and read from. This memory is called parent-child memory.
[10] The OPNET-packet-format file for the packets stored in the BS-list is BS_list_entry.pk.

number of consecutive minislots in the range. The direction indicates whether it is a grant or a grant-confirmation, and the channel number identifies the channel of the link that the grant belongs to. A grant expires at the end of the frame range. Each of the packets stored in the grant-list subqueues contains all the previous parameters (i.e., grant parameters) plus the frame and minislot numbers when the grant was transmitted and the expiration time of the grant (i.e., the time instant of the end of the last frame included in the grant)[11].

- Input-queue-list subqueues: There are 256 of these subqueues (i.e., one subqueue per link ID[12]). However, only the subqueues for established outgoing links are used. These subqueues are called input-queues. Therefore, at every node, there are as many input-queues as many 1-hop neighbors the node has (i.e., one input-queue per outgoing link). Each of the packets stored in input-queues contain the following information[13]: a data packets generated by P_7 (i.e., payload), an indicator that the data packet is being scheduled, and the frame and minislot number fields of the schedule of the data packet.
- Request-list subqueues: There are 10 of these subqueues (i.e., one subqueue per channel). For every channel, there is one packet in the subqueue per unexpired interfering request. A request interferes in the same way that grants do (i.e., it interferes if it belongs to a link that interferes with any of the links of the node the request-table belongs to), and it consists of a link ID, demand-level, and demand-persistence. The link ID identifies the link the request belongs to. The demand-level determines the size (i.e., number of minislots per frame) of the requested minislot-range, and the demand-persistence determines the number of consecutive frames for which the demand-level is requested. It is assumed that a request is always made along with an availability that specifies the set of frames and minislots that the node making the request is able to reserve. An availability consists of the start-minislot, minislot-range-size, start-frame, persistence, direction, and channel number, which share the same definitions of the grant parameters. Each of the packets stored in request-list subqueues has fields for the parameters of a request and an availability, the ID of the node that sent the request and availability, the frame and minislot numbers when the request and availability were sent, and a flag for indicating when a request has been granted[14].
- Request-timer-list subqueue: There is only 1 subqueue of this type. The packets stored in this subqueue have one field only that is used for a pointer to a request-timer[15]. A request-timer is a structure with members for an event-handle, a request, and an availability. The event-handle references the future event that a request expires. There is always a request-timer per request in the request-list subqueues. Whenever a request expires, the scheduler process that is in charge of processing the request is interrupted.

Process_model_01 performs the following operations, defined in the exit executives of state st_1 (see Fig. 5), when an MSH-NCFG packet[16] is input through any of the input streams connected to any of the process modules P_5_1, P_5_2, …, and P_5_10.

- It updates the physical-neighbor-list if the node that sent the MSH-NCFG message or any of its 1-hop neighbors has not been included in the list or any of their parameters have changed.
- It updates the BS-list if the there are any BSs specified in the MSH-NCFG message that have not been included in the list or if any of their parameters has changed.
- It creates a link-establishment process for any 1-hop neighbor entry in the physical-neighbor-list for which the incoming and outgoing link IDs have not been assigned, and then it invokes the link-establishment process.

Process_model_01 performs the following operations, defined in the same exit executives, when an MSH-DSCH packet[17] is input through the same input streams.

- It updates the grant-list by adding to it all the grants included in the MSH-DSCH message.
- It updates the request-list by adding to it all the requests included in the MSH-DSCH message, creates a request-timer for each request, and stores the timer in the request-timer-list.
- It updates the current frame and minislot numbers in the PCMs used for communicating with the schedulers.
- It invokes the schedulers and passes to them the received MSH-DSCH packet.

[11] The OPNET-packet-format file for the packets stored in the grant-lists is scheduled_grant.pk.
[12] The IEEE 802.16 standard [1] defines a maximum of 256 link IDs.
[13] The OPNET-packet-format file for the packets stored in the input-queues is data_pk_and_schedule.pk.
[14] The OPNET-packet-format file for the packets stored in the request-lists is pending_request.pk.
[15] The OPNET-packet-format file for the packets stored in the request-timer-lists is timer_pk_fmt.pk.
[16] An MSH-NCFG packet consists of an MSH-NCFG message encapsulated in a MAC packet. The OPNET-packet-file for MSH-NCFG messages is MSH_NCFG.pk, and the OPNET-packet-file for MAC packets is MAC_PDU.pk.
[17] An MSH-DSCH packet consists of an MSH-DSCH message encapsulated in a MAC packet. The OPNET-packet-file for MSH-DSCH messages is MSH_DSCH.pk, and the OPNET-packet-file for MAC packets is MAC_PDU.pk.

Process_model_01 performs the following operations, defined in the same exit executives, when a data packet[18] is input through any of the same input streams.

- It unpacks the payload (i.e., upper-layer's data packet) from the received packet, packs it in another packet, and forwards this packet to the upper layer. The packet used for forwarding the payload has a command control field which is used for identifying packets with payloads from packets with link-establishment notifications.

Process_model_01 performs the following operations, defined in the same exit executives, when a data packet is input through the input stream connected to P_7[19].

- It unpacks the payload (i.e., upper-layer's data packet) from the received packet, packs it in another packet that has fields for its schedule (i.e., frame and minislot numbers) and a flag for indicating whether the data packet is being scheduled. Then, it stores the packet in the input-queue assigned to the outgoing link the upper-layer's data packet needs to be sent through.

Process_model_01 performs the following operations, defined in the same exit executives, at the onset of every minislot.

- It updates the current frame and minislot numbers.
- It deletes any expired grants from the grant-lists.
- It checks whether there is already a Link-Establishment IE to be attached to the MSH-NCFG message[20]. If there is no such Link-Establishment IE, it checks whether there are any new 1-hop neighbors with higher IDs with whom links have not been established. If there are any of such neighbors, an ID for an outgoing link is calculated, a link-establishment process is created[21], the link ID is written on the PCM, and the process is invoked. This process generates the Link-Establishment IE that includes the challenge that initiates the link-establishment three-way handshake with the new 1-hop neighbor.
- It checks whether the current frame and minislot numbers correspond to the schedule of an MSH-NCFG-packet transmission. If they do, it generates the MSH-NCFG packet that is transmitted at the next schedule for an MSH-NCFG packet transmission, and sends it to the corresponding Q_4 queue module[22]. In this way, every time the node detects that the current frame and minislot numbers correspond to one of its MSH-NCFG packet transmissions, it generates the next MSH-NCFG packet to be transmitted. Therefore, the MSH-NCFG packet that is transmitted at the current frame and minislot numbers was generated at the previous MSH-NCFG packet transmission and is already being transmitted by the radio-transmitter module. Process_model_01 generates the MSH_NCFG packet as follows. It creates a Network-Descriptor IE which is the IE that contains the configuration parameters necessary for new nodes to start a network-entry process[23], generates an MSH-NCFG message, attaches the Network-Descriptor and Link-Establishment IEs to the message, and creates the control packet (i.e., MSH-NCFG packet) that carries the message. Before sending the packet to its corresponding Q_4 queue module, it calculates the packet's transmission schedule (i.e., the schedule of the next MSH-NCFG packet transmission). Then, the packet and the schedule are sent to the corresponding Q_4 queue module[24].
- It checks whether the current frame and minislot numbers correspond to the schedule of an MSH-DSCH-packet transmission. If they do, it generates the MSH-DSCH packet that is transmitted at the next schedule for an MSH-DSCH packet transmission and sends it to the corresponding Q_4 queue module. Therefore, the MSH-DSCH packets are generated in the same way MSH-NCFG packets are (i.e., at the current MSH-DSCH-packet transmission, the MSH-DSCH packet for the next MSH-DSCH-packet transmission is generated and sent along with its schedule to its corresponding Q_4 queue module). In order to generate the MSH-DSCH packet, first, process_model_01 invokes the scheduler process so that the Request, Availability, and Grant IEs that need to be transmitted are

[18] A data packet consists of the payload encapsulated in a MAC packet. The OPNET-packet-file for payloads is Payload.pk, and the OPNET-packet-file for MAC packets is MAC_PDU.pk.

[19] The data packets output from P_7 consist of a payload encapsulated in a command-interface packet. The OPNET-packet-file for payloads is Payload.pk, and the OPNET-packet-file for command-interface packets is SAP_command_interface.pk.

[20] In the model, there can only be one Link-Establishment IE per MSH-NCFG message.

[21] Only one link-establishment process is created and invoked because only one Link-Establishment IE can be attached to an MSH-NCFG message. If there are more than one link-establishment processes that need to be created (i.e., there are more than one 1-hop neighbors with whom links have not been established), these will be created at the rate of one process per transmitted MSH-NCFG message.

[22] The MSH-NCFG packet to be transmitted at the current frame and minislot numbers is already at the radio-transmitter module. This packet was generated at the previous MSH-NCFG-packet-transmission schedule.

[23] A network-entry process is the process followed by nodes to acquire the network-configuration parameters and join the network. This process is not included in the simulation model. In the simulation model, all the nodes obtain the network-configuration parameters from the simulation attributes when they are at the initial state (i.e., st_0 in Fig. 5).

[24] The schedule is sent as an independent packet silently (i.e., without causing an interruption to the Q_4 queue module). The OPNET-packet-format file for schedules is PDU_schedule.pk.

calculated. Second, it calculates the schedule of the next MSH-DSCH-packet transmission, creates the MSH-DSCH packet, and attaches the Request, Availability, and Grant IEs to it. Third, it updates the grant-list and request-list by adding to them all the grants and requests included in the MSH-DSCH packet. Finally, the packet and the schedule are sent to the corresponding Q_4 queue module[25].

- It looks in all the input-queues for data packets that have already been scheduled. The scheduled data-packets are removed from their input-queues. The payloads of these packets (i.e., data packets generated by P_7) are extracted and MAC headers are attached to them. The schedules specified in the packets are extracted too. The data packets and their respective schedules are finally sent to their corresponding Q_4 queue modules[26].

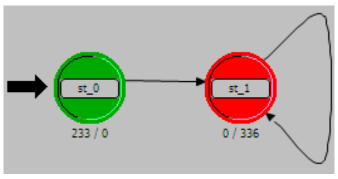

Fig. 5. Process_model_01 STM

Also, at the end of the initialization period, process_model_01 checks that all the links with the node's 1-hop neighbors have been established. If that is not the case, it aborts the simulation. This is done in order to assure that when P_7 begins to generate data packets (i.e., at the end of the initialization period), all the links with the node's 1-hop neighbors have already been established.

The process-model configuration is as shown in Table 1 (see Appendix I). The global attributes (Table 1) and node attributes (Table 6) define most of the configuration parameters of the node, which are set in the initial state of process_model_01 (i.e., state st_0 in Fig. 5). These attributes are defined as follows.

- The number of channels supported by the node
- The duration of frames in seconds
- The number of minislots in control-subframes
- The number of consecutive scheduling frames between every two network-configuration frames
- The center frequency of the lowest physical channel supported by the node
- The bandwidth of the physical channel
- The modulation scheme and coding rate of the physical layer
- The duration of the initialization period in seconds
- The node's interference radius in meters
- The node's transmission radius in meters
- NTx value
- HE value
- The required minimum number of unscheduled data packets to initiate scheduling handshakes[27]
- Minislot groups' size[28]

The local and global statistics in Table 1 (see Appendix I) share the same definitions. They can be used for monitoring the following characteristics of the node.

- The size of the two-hop neighborhood (i.e., the number of entries in the physical-neighbor-list)
- The size of the input-queues
- The data-packet-delivery delay (i.e., the time difference from the instant a data packet enters the input-queue at the source node until the instant the packet is delivered to processor module P_7 at the destination node)
- The node's throughput in packets per second (i.e., the total number of data packets per second received by processor module P_7)
- The node's throughput in bits per second (i.e., the total number of bits per second received by processor module P_7)
- The number of 1-hop neighbors
- The scheduling three-way handshake delay (i.e., the time difference from the instant a Request IE is

[25] The schedule is sent as an independent packet silently (i.e., without causing an interruption to the Q_4 queue module). The OPNET-packet-format file for schedules is PDU_schedule.pk.
[26] The schedules are sent as independent packets silently (i.e., without causing an interruption to the Q_4 queue module). The OPNET-packet-format file for schedules is PDU_schedule.pk.
[27] See Section 4.1.3 for the definition of this attribute
[28] See Section 4.1.3 for the definition of this attribute

transmitted until the confirmation Grant IE is transmitted)

- The access delay of schedueduling-control-subframes (i.e., the time difference between every two consecutive MSH-DSCH packet transmissions)
- The number of data packets the node receives with collisions
- The number of data packets the node receives without collisions

3.1.2. Establish_incoming_link_connectivity

This process implements the three-way handshake procedure used for establishing links between 1-hop neighbors. It is shown in Fig. 6. The process is created by process_model_01 every time a new 1-hop neighbor is added to the physical-neighbor-list. The process is able to perform the two different roles present in the three-way handshake. One of them is the challenger of the handshake, and it corresponds to the establish_incoming_link_connectivity process at the 1-hop neighbor with lower ID that initiates the handshake by sending a challenge. The other role is the replier of the handshake, and it corresponds to the establish_incoming_link_connectivity process at the 1-hop neighbor with higher ID which replies to the challenge with a challenge-response. In the initial state, the process decides its role according to the ID of its node and the ID of the 1-hop neighbor. Also, in this state, the process creates a list of the 1-hop neighbors that the node needs to send packets to. In Fig. 6, the challenger jumps from the initial state (i.e., st_0) to state st_7, and the replier jumps to state st_4.

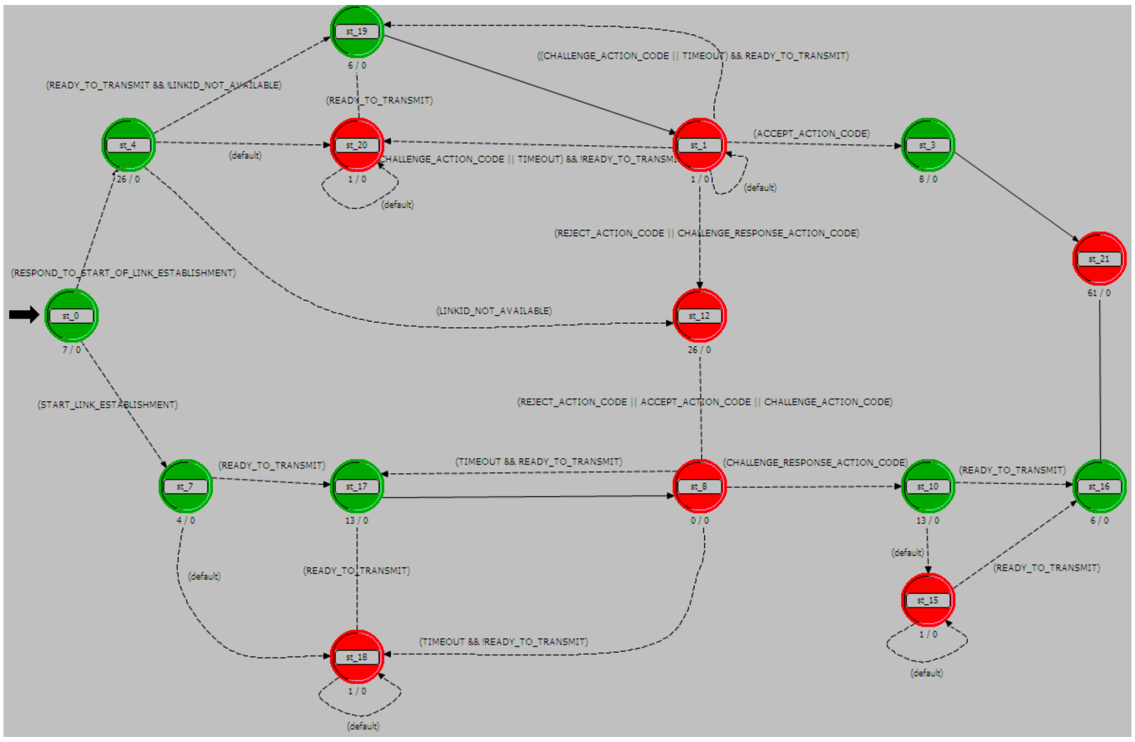
st_19
6 / 0
((CHALLENGE_ACTION_CODE || TIMEOUT) && READY_TO_TRANSMIT)
(READY_TO_TRANSMIT && !LINKID_NOT_AVAILABLE)
(READY_TO_TRANSMIT)
st_4
26 / 0
(default)
st_20
1 / 0
(default)
CHALLENGE_ACTION_CODE || TIMEOUT) && !READY_TO_TRANSMI
st_1
1 / 0
(default)
(ACCEPT_ACTION_CODE)
st_3
8 / 0
(REJECT_ACTION_CODE || CHALLENGE_RESPONSE_ACTION_CODE)
st_21
61 / 0
(RESPOND_TO_START_OF_LINK_ESTABLISHMENT)
st_0
7 / 0
(LINKID_NOT_AVAILABLE)
st_12
26 / 0
(START_LINK_ESTABLISHMENT)
(REJECT_ACTION_CODE || ACCEPT_ACTION_CODE || CHALLENGE_ACTION_CODE)
(READY_TO_TRANSMIT)
(TIMEOUT && READY_TO_TRANSMIT)
(CHALLENGE_RESPONSE_ACTION_CODE)
(READY_TO_TRANSMIT)
st_7
4 / 0
st_17
13 / 0
st_8
0 / 0
st_10
13 / 0
st_16
6 / 0
(default)
(READY_TO_TRANSMIT)
(default)
(READY_TO_TRANSMIT)
st_15
1 / 0
(default)
(TIMEOUT && !READY_TO_TRANSMIT)
st_18
1 / 0
(default)

Fig. 6. Establish_incoming_link_connectivity STM

Upon arrival to st_7, the process checks whether there is already a Link-Establishment IE in the PCM that needs to be attached to the MSH-NCFG message[29]. If there is such Link-Establishment IE, the process waits for certain time at st_18 for that Link-Establishment IE to be transmitted. Then, the process checks again whether it has already been transmitted. The process repeats this cycle until it finds that there is no Link-Establishment IE in the PCM. Then, it jumps to st_17. At this state, it writes on the PCM its Link-Establishment IE, which includes the challenge that initiates the handshake, and sets a timer for the retransmission of this Link-Establishment IE in case it receives no response from the 1-hop neighbor. Once this is done, it jumps to st_8 and waits for the response. If the response is not the expected challenge-response, it jumps to st_12, terminates the handshake, and destroys itself. In this case, process_model_01 will later create another establish_incoming_link_connectivity process for establishing the links with the 1-hop neighbor. If the response is the expected challenge-response, it reads the incoming-link ID assigned by the 1-hop neighbor from the PCM and creates another Link-Establishment IE that includes the accept for the challenge-response and the

[29] In the simulation model, only one Link-Establishment IE can be attached to an MSH-NCFG message.

outgoing-link ID[30]. Then, it follows the same procedure that was used for transmitting the Link-Establishment IE with the challenge (i.e., it waits until there is no Link-Establishment IE in the PCM that needs to be attached to the MSH_NCFG message) in order to transmit the Link-Establishment with the accept. Once the Link-Establishment with the accept is written on the PCM at st_16, the ID of the incoming link is written on the 1-hop neighbor's entry in the physical-neighbor-list. Finally, it jumps to st_21 where it sends a packet to P_7 to notify the new established outgoing link if the destination node is in the list of 1-hop neighbors created in the initial state (i.e., the list of 1-hop neighbors that the node needs to send packets to); then, the process destroys itself.

Upon arrival to st_4 (i.e., when the process takes the replier role), it calculates the ID for the outgoing link, writes it on the corresponding entry in the physical-neighbor-list, and creates the Link-Establishment IE that includes the challenge-response and outgoing link ID. Then, it checks whether there is already a Link-Establishment IE in the PCM. If there is, it waits at st_20 for that Link-Establishment IE to be transmitted, and if there is not, it jumps to st_19. At this state, it writes its Link-Establishment IE on the PCM and sets a timer for the retransmission of the Link-Establishment IE in case no response is received from the 1-hop neighbor. Then, it jumps to st_1 to wait for the response. When the response is received, it checks whether it is another challenge, the expected accept, or any other response (i.e., reject, challenge-response). If it is another challenge, it is assumed that the previously transmitted challenge-response was not received and the Link-Establishment IE is retransmitted. If it is the expected accept, it jumps to st_3 where it reads the incoming link ID from the response and writes it on the corresponding entry in the physical-neighbor-list, and then jumps to st_21 where it sends a packet to P_7 to notify the new established outgoing link and destroys itself. If the response is a reject or challenge-response, it jumps to st_12 where it terminates the handshake and destroys itself given that an unexpected response was received. In this case, process_model_01 will later create another establish_incoming_link_connectivity process for establishing the links with the 1-hop neighbor.

The PCM of each establish_incoming_link_connectivity process is a structure whose members contain the following information:

- Node ID: This is the ID of the node that the process belongs to. This parameter is passed by the parent process (i.e., process_model_01) to the establish_incoming_link_connectivity process, and this process uses it for determining the role it plays in the handshake (i.e., challenger, replier).
- Frame length: This is the frame duration. This parameter is passed by the parent process (i.e., process_model_01) to the establish_incoming_link_connectivity process, and this process uses it for setting the timers used for waiting for the 1-hop neighbor's responses.
- Channel number: This is the channel number of the incoming and outgoing links the process establishes. It is used for indexing the list of Link-Establishment-IEs (see the definition of this list below).
- Neighbor authentication values: There are two of these values which are used for authenticating the nodes according to the standard. However, in the simulation model, the values correspond to the IDs of the two nodes establishing the two links between them, and they are used by the establish_incoming_link_connectivity process for identifying the source and destination nodes of the Link-Establishment IEs only. This parameter is passed by the parent process (i.e., process_model_01) to the establish_incoming_link_connectivity process.
- Unassigned-link-IDs list: This is a list of the link IDs that have not been assigned to any of the outgoing links of the node. It is used by the establish_incoming_link_connectivity process when it takes the replier role for assigning a link ID to the outgoing link being established. This parameter is passed by the parent process (i.e., process_model_01) to the establish_incoming_link_connectivity process.
- Link ID: This is the link ID for the incoming link, which is assigned by the 1-hop neighbor. The parent process (i.e., process_model_01) reads from the received MSH-NCFG packet's Link-Establishment IE such link ID and writes it on the PCM before invoking the establish_incoming_link_connectivity process. The establish_incoming_link_connectivity process reads it from the PCM in order to assign it to the respective incoming link.
- Action code: This is the action code (i.e., challenge, challenge-response, accept, reject) of the received Link-Establishment IE. The parent process (i.e., process_model_01) reads it from the received MSH-NCFG packet and writes it on the PCM. The establish_incoming_link_connectivity process reads it from the PCM and determines the state it has to jump to when it is invoked by the parent process.
- Link-Establisment IE: The establish_incoming_link_connectivity process keeps an internal Link-Establishment IE that it prepares for transmission. Once the internal Link-Establishment IE is

[30] The parent process (i.e., process_model_01) calculates the outgoing-link ID and writes it on the PCM before the establish_incoming_link_connectivity process is created.

ready, the process checks whether there is already a Link-Establishment IE in the PCM to be attached to the next MSH-NCFG packet the node transmits. If there is not such Link-Establishment IE, the process copies its internal Link-Establishment IE to the PCM's Link-Establishment IE, which is later attached by the parent process (i.e., process_model_01) to the MSH-NCFG packet before it is transmitted.

- Link-Establishment-IEs list: There is one entry in this list per channel, and each entry points to the Link-Establishment IE that the parent process (i.e., process_model_01) attaches to the next MSH-NCFG packet the node transmits. When the parent process attaches a Link-Establishment IE to an MSH-DSCH packet, it empties the respective entry in the list. An empty entry indicates that the establish_incoming_link_connectivity process can write the copy of its internal Link-Establishment IE on the PCM. After writing the copy, the establish_incoming_link_connectivity process also needs to update the respective entry in the list.
- Channel bandwidth, modulation and coding scheme, number of minislots in control subframes, and node's transmission power: These are physical-layer parameters of the outgoing link that the establish_incoming_link_connectivity process establishes, and they are used for the new-outgoing-link notification that the process sends to P_7. The value of these parameters are set by the parent process (i.e., process_model_01) before it creates the establish_incoming_link_connectivity process.

3.1.3. MSH_DSCH_scheduler

This process implements the distributed-scheduling algorithm used for scheduling the data packets in all input-queues. The algorithm is based on the sets of minislots shown in Fig. 7. In order to specify these sets, the data subframes are all divided into a given number of minislot groups of the same size. For example, in Fig. 7, there are 12 minislots per frame numbered from 4 to 15 which are divided into 4 minislot groups. The minislots numbered 4-6, 7-9, 10-12, and 13-15 correspond to each of these groups. A set of minislots is specified with a minislot group and a frame range. For example, if the group corresponds to minislots numbered 7-9 and the frame range consists of frames numbered from 11 to 20, the set of minislots consists of 30 minislots (i.e., 10 frames and 3 minislots per frame). This set is shown in Fig. 7 along with other three different sets. In the simulation model, there are 256 minislots per frame as specified in the standard [1], and the size of the minislot groups is specified with a node attribute (see Appendix II).

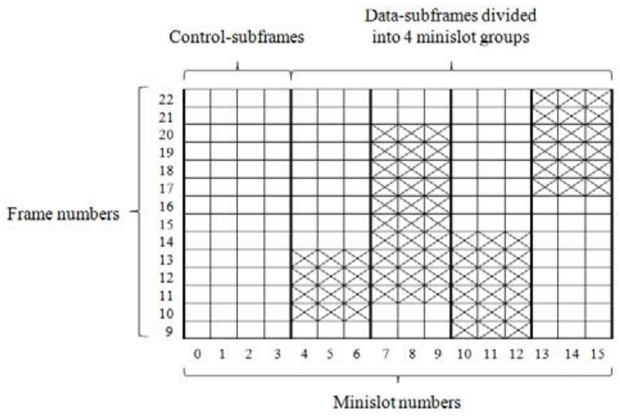

Fig. 7. Sets of minislots used in the distributed-scheduling algorithm

In the algorithm, a Request IE is always generated along with an Availability IE. The Request IE specifies the size of a set of minislots (i.e., the number of requested minislots) with the demand-persistence (i.e., the number of frames in the frame range covered by the set of minislots). The Availability IE specifies a set of minislots (i.e., minislot group and frame range) whose size is at least equal the size of the set of minislots specified in the Request IE. The Availability IE's set of minislots indicates all the minislots 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 minislots that belong to any of the grants in the node's grant-list, 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 minislots within the Availability IE's set that does not overlap any of the grants in its grant-list and that has the size specified in the Request IE.

The scheduling algorithm is based on the policy that nodes should grant only the longest [31]

[31] The length of a request is defined as the number of minislots included in the request.

non-overlapping[32] requests directed to them. In order to comply with this policy, each scheduler performs the following operations for granting and requesting minislots at every MSH-DSCH packet transmission[33]. These operations are performed in the exit executives of st_1 in Fig. 8.

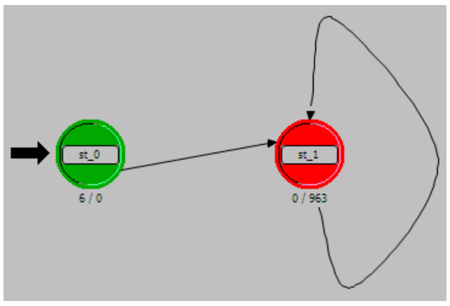

Fig. 8. MSH_DSCH_scheduler STM

- Grant IE generation: For every request in the request-list directed to the node, the set of overlapping requests is found first. If the request is the longest in this set, it is granted and all the other requests in the set are later ignored (i.e., they are not longer eligible for any grant). The grant for each of the granted requests is assigned a set of minislots that is determined as follows. The set's minislot group is the one specified in the request's Availability IE. The size of the set's frame range (i.e., grant's persistence) is the one specified in the respective Request IE (i.e., request's demand-persistence), and the number of the set's start-frame is the maximum between the following three frame numbers: the end-of-grant-list frame number[34] of the Availability IE's minislot group, the Availability IE's start-frame number, and the current frame number plus a pre-specified number of frames[35]. In this way, it is assured that the grant does not overlap any of the grants in the grant-lists of the node and 1-hop neighbor and that the 1-hop neighbor is able to confirm the grant in its next MSH-DSCH packet transmission before the grant's start-frame becomes the current frame. Also, the request-timer of every granted request is removed from request-timer-list and its scheduled interruption (i.e., the interruption directed to the scheduler process that occurs when the request expires) is cancelled.
- Request and Availability IEs generation: For every outgoing link, if there is certain minimum of unscheduled data packets in its input-queue, a request is made for the maximum number of minislots that can be covered with the unscheduled data packets and that is multiple of the minislot-groups' size. The availability of each request is such that
  - It does not overlap any of the grants in the node's grant-list.
  - If there is a minislot group which has not been included in any of the availabilities that are in the request-list or any of the availabilities that were previously calculated for other outgoing links, that minislot group is assigned to the availability being calculated, and the availability's start-frame number is the maximum of the following frame numbers: the end-of-grant-list frame number of the minislot group, the frame number when the next MSH-DSCH packet is transmitted.
  - If there is not such minislot group, the minislot group that has been assigned to the lowest number of availabilities among all the availabilities in the request-list and the availabilities previously calculated for other outgoing links is assigned to the availability, and the availability's start-frame number is the maximum of the following frame numbers: the maximum end-of-grant-list frame number across all minislot groups, the frame number when the next MSH-DSCH packet is transmitted.
  - The number of the last frame of the frame range is always made equal to infinity. This is possible because none of the minislots that belong to the availability's minislot group and to the frames that follow the availability's start-frame has been granted.

Also, the scheduler is invoked for processing any grants directed to the node at every MSH-DSCH packet reception. This processing includes scheduling data packets according to the grants, deleting any requests from

[32] Two requests overlap with each other if their availabilities have one or more minislots in common.
[33] The scheduler is invoked by process_model_01 before the creation of every MSH-DSCH packet.
[34] The end-of-grant-list frame number of a given minislot group is defined as follows. Consider all the set of minislots that are in the given minislot group and that are defined by the grants in the grant-list. Consider the maximum frame number included in such sets of minislots. The end-of-grant-list frame number of the given minislot group is that maximum frame number plus one (i.e., the frame following the last granted frame in the minislot group).
[35] The pre-specified number of frames grows exponentially with HE (i.e., 2^{HE}).

the request-lists for which there are grants or grant confirmations in the packet, destroying the request-timers of such requests, and cancelling the interruptions of these timers. These operations are performed as follows.

- For every Grant IE in the MSH-DSCH packet, the scheduler checks whether the grant is directed to the node. If it is, the scheduler looks in the physical-neighbor-list for the ID of the outgoing-link that connects to the 1-hop neighbor that sent the grant. This ID is later used for creating the grant confirmation. Then, it schedules data packets that are in the input-queue assigned to the outgoing-link. These packets are scheduled in the set of minislots 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 with the direction parameter set at the value that indicates a confirmation.
- For every request in the request-list assigned to the channel the MSH-DSCH packet was received from, the scheduler checks in the packet whether there is a grant directed to the node or a grant confirmation that matches the request. If there is such grant or grant confirmation, the scheduler looks for the request-timer in the request-timer-list by comparing the Request and Availability IE parameters of each request-timer with the Request and Availability IE parameters of the request. When the timer is found, it is destroyed and its scheduled interruption is cancelled. Then, the request is removed from the request-list.

Finally, at every request-timer expiration, the scheduler needs to change the flag of data packets in the corresponding input-queue from being-scheduled to not-being-scheduled, to remove the expired request from the request-list, remove the request-timer from the request-timer-list, and to destroy it. These actions are performed as follows.

- It looks in the request-timer-list for the request-timer that expired by comparing the event handle of each request-timer in the list with the event handle of the current event (i.e., request-timer expiration).
- It checks whether the request-timer belongs to a request it generated previously. If it does, it reads the number of requested minislots from the Request IE saved in the request-timer, and changes the flag of data packets. The number of data packets to which the flag change is performed is equal to the number of requested minislots.
- It looks in the request-table for the request that expired by comparing the Request-IE parameters of each table's entry with the Request-IE parameters saved in the request-timer, removes the request from the table, and destroys it.
- It deallocates the memory assigned to the request-timer, and destroys the timer.

There is one scheduler process per channel. The schedulers are all created by process_model_01 at its initial state only, and they exist for the whole duration of the simulation. Each scheduler is assigned to one of the channels and is in charge of scheduling the packets in the input-queues of the outgoing-links that are in its channel only. A PCM is assigned to each scheduler when they are created. The PCM of each scheduler is a structure whose members contain the following information.

- Process handle: This is the scheduler's process handle, and it is used only by the parent process (i.e., process_model_01) to invoke the scheduler.
- Number of minislots in control subframes: This frame-structure parameter (see Fig. 1) is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for indexing the first minislot of data-subframes (i.e., minislot 4 in Fig. 7) and for calculating the minislot group and start-frame of Availability IEs.
- Node's HE: This control-subframe-MAC parameter is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for calculating the start-frame of Grant IEs.
- Number of consecutive scheduling-frames between every two network-configuration frames: This frame-structure parameter (see Fig. 1) is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for calculating the start-frame of Grant IEs.
- Required minimum number of unscheduled data packets to initiate scheduling handshakes: This scheduling parameter is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for determining when to initiate a scheduling handshake for any of the node's outgoing links. There have to be at least this number of unscheduled and not-being-scheduled data packets in the input-queue of an outgoing link for the scheduler to initiate a scheduling handshake for that link by generating a Request IE.
- Minislot groups' size: This scheduling parameter (i.e., number of minislots per minislot group) is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for calculating the demand-level and demand-persistence of Request IEs.
- Node ID: The node ID is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for determining whether a request-timer belongs to a request it generated previously.

- Channel number: The channel number that the scheduler has been assigned to is passed by the parent process (i.e., process_model_01) to the scheduler. The scheduler uses it for indexing the subqueues of the queue module phys-neighbor-list (see Table 1) and specifying the channel-number field of Availability and Grant IEs.
- Minislot capacity: This is the maximum number of bits per minislot that can be transmitted according to the modulation scheme and coding rate. This parameter is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for determining whether the next MSH-DSCH packet the node transmits does not exceed the minislot capacity so that Request, Availability, and Grant IEs can still be attached to the packet.
- MAC-overhead size: This is the number of bits of overhead in all packets. This parameter is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for determining whether the next MSH-DSCH packet the node transmits does not exceed the minislot capacity.
- Current frame number: The current frame number is passed by the parent process (i.e., process_model_01) to the scheduler, and the scheduler uses it for calculating the start-frames of Availability and Grant IEs.
- Number of Request IEs: This is the number of Request IEs that the scheduler has generated for the next MSH-DSCH packet the node transmits. These Request IEs are added to a list. The number of Request IEs in this list is passed by the scheduler to its parent process (i.e., process_model_01), and the parent process uses it for attaching the list to the packet.
- Number of Availability IEs: This parameter is used for the Availability IEs generated by the scheduler in the same way that the number-of-Request-IEs parameter is used for the Request IEs generated by the scheduler too.
- Number of Grant IEs: This parameter is used for the Grant IEs generated by the scheduler in the same way that the number-of-Request-IEs parameter is used for the Request IEs generated by the scheduler too.
- Request-IEs list: This is a pointer to the first entry of the list of Request IEs that the scheduler generates for the next MSH-DSCH packet the node transmits. This parameter is passed by the scheduler to its parent process (i.e., process_model_01), and the parent process uses it for attaching the list to the packet.
- Availability IEs: This parameter is used for the list of Availability IEs that the scheduler generates for the next MSH-DSCH packet the node transmits in the same way that the Request-IEs-list parameter is used for the Request IEs.
- Grant IEs: This parameter is used for the list of Grant IEs that the scheduler generates for the next MSH-DSCH the node transmits in the same way that the Request-IEs-list parameter is used for the Request IEs.

3.1.4. Process_model_04

This process model defines the functionality of queue modules Q_4_1, Q_4_2, …, Q_4_10. Each of these queue modules is assigned to a specific channel. Three input streams and one output stream are connected to each of them. Each input stream carries packets with one type of message only (i.e., MSH-NCFG, MSH-DSCH, or data), and the output stream carries packets with any type of message that need to be transmitted. Each queue module has nine subqueues. These are:

- MSH_NCFG subqueue: Packets with MSH_NCFG messages are stored in this subqueue.
- MSH_NCFG scheduling subqueue: The schedules of the packets in the MSH_NCFG subqueue are stored in this subqueue. These schedules are stored in the form of packets.
- MSH_NCFG old subqueue: Copies of the packets with MSH_NCFG messages that are sent to the radio-transmitter module are stored in this subqueue. These copies are used for deallocating the memory used for the contents of the packet after the packet has already been processed at the destination node[36].
- MSH_DSCH subqueue: Same as MSH_NCFG subqueue but for packets with MSH_DSCH messages.
- MSH_DSCH scheduling subqueue: Same as MSH_NCFG scheduling subqueue but for packets with MSH_DSCH messages.
- MSH_DSCH old subqueue: Same as MSH_NCFG old subqueue but for packets with MSH_DSCH messages.
- Data subqueue: Same as MSH_NCFG subqueue but for data packets.

[36] When a packet is transmitted, all the nodes within the transmission range receive copies of it. However, the IEs carried by the transmitted packet are in memory that was allocated by the transmitting node. The received copies only carry pointers to that memory. In RBDS-Sim, memory is always deallocated by the same node that allocated it (i.e., the transmitting node).

- Data scheduling subqueue: Same as MSH_NCFG scheduling subqueue but for data packets.
- Data old subqueue: Same as MSH_NCFG old subqueue but for data packets.

Each queue module performs the following operations, defined in the exit executives of state st_1 (see Fig. 9), when a packet is input from any of the input streams.

- It classifies the incoming packets into MSH-NCFG, MSH-DSCH, and data packets according to the input stream the packets come from.
- It stores the incoming packets along with their schedules in their corresponding subqueues. The schedules are always sent in the form of packets by the queue module phys_neighbor_list silently. Therefore, they do not cause any interruptions on the queue modules. The arrival of a schedule is indicated with the input-stream interruption caused by the packet that the schedule belongs to. Every time phys_neighbor_list sends through any of the input-streams an MSH-NCFG, MSH-DSCH, or data packet to any of the queue modules, it also delivers silently one packet with the schedule of this packet.

And each queue module performs the following operations, defined in the exit executives of state st_1 (see Fig. 9), at the onset of every minislot.

- It checks whether the minislot corresponds to any of the schedules of the packets in the subqueues. If it does, it makes a copy of the packet, sends the packet to the radio-transmitter, deallocates the memory used in previously transmitted packets of the same type, and stores the copy in the corresponding subqueue (i.e., MSH_NCFG old subqueue, MSH_DSCH old subqueue, or data old subqueue).

The process-model configuration is as shown in Table 2. The local and global statistics named output_queue_length can be used for monitoring the number of data packets in the data subqueue. The local statistic named data_pk_tx_minislot_num can be used for monitoring the minislot numbers when data packets are transmitted. The global attribute named num_channels is the simulation attribute that indicates the number of channels.

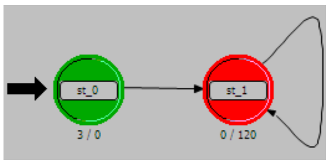

Fig. 9. Process_model_04 STM

3.1.5. Process_model_05

This process model defines the functionality of processor modules P_5_1, P_5_2, …, P_5_10. Each of these processor modules is assigned to a specific channel. One input stream and three output streams are connected to each of them. The input stream carries the packets received by the node. These packets are received by the radio-receiver module across all the channels. Depending on the specific channel that a packet is received from, the radio-receiver module sends the packet through one of its output streams (i.e., the input streams of the processor modules) to the respective processor module. Therefore, the packets that the processor modules receive through their input streams can be of any type (i.e., MSH-NCFG, MSH-DSCH, or data), and for a given processor module, all the packets come from the same channel (i.e., the channel that the processor module has been assigned to). The output streams of the processor modules carry packets of one type only. Each processor module has three output streams which carry MSH-NCFG, MSH-DSCH, and data packets each. Process_model_05 defines the only task these processors do: to classify all the received packets into the three categories and forward the classified packets to phys_neighbor_list module.

The processor modules only classify the packets the node receive into the three different types and forward the classified packets to the queue module phys_neighbor_list. Each processor module performs the following operations, which are defined in the exit executives of state st_1 (see Fig. 10), every time there is a packet arrival through the input stream.

- It checks whether the packet was transmitted by the node it belongs to. If that is the case, the packet should not be forwarded to the queue module phys_neighbor_list (i.e., the node does not need to process any of the packets it transmits after they have been transmitted). These packets are destroyed.
- If the packet was not transmitted by the node (i.e., it was transmitted by a 1-hop neighbor), it reads from the payload the type of packet, and according to this type, it sends it through one of the output streams.

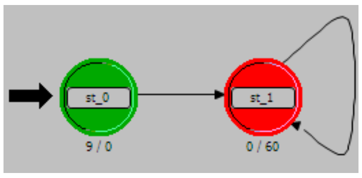

Fig. 10. Process_model_05 STM

The process-model configuration is shown in Table 3 (see Appendix I). The local and global statistics named num_rx_DSCH_pks_without_colls, num_rx_DSCH_pks_with_colls, num_rx_NCFG_pks_without_colls, and num_rx_NCFG_pks_with_colls can be used for monitoring the number of control packets (i.e., MSH-NCFG and MSH-DSCH) that arrive with and without collisions. The process-model writes a 1 to the corresponding statistic whenever a packet has one or more collisions or no collisions at all.

3.1.6. Process_model_07

This process model defines the functionality of processor module P_7. It has one input stream and one output stream. The input stream is for carrying the data packets and new-link-establishment notifications sent by the queue module phys_neighbor_list. The output stream is for carrying the data packets that it generates.

Process_model_07's main task is to serve as the node's source and sink of data packets. It generates data packets randomly with distribution and average rate for some or all of the outgoing links of the node. The distribution, average rate, and outgoing links for which data packets are generated are specified by node attributes (see Appendix II). The packets sent to queue module phys_neighbor_list consist of a payload (i.e., data) and control fields that include a command field and an outgoing-link-ID field. The command field is always used for indicating that the packet carries data[37], and the outgoing-link-ID field indicates the outgoing link that the data needs to be sent through. The packets sent by queue module phys_neighbor_list to processor module P_7 have the same control fields. In the simulation model, the command field of these packets is used for indicating whether the packets are data-packet receptions or new-link-establishment notifications.

In order to carry out its task, process_model_07 performs the following operations every time it generates a data packet. These operations are defined in state st_1 (see Fig. 11).

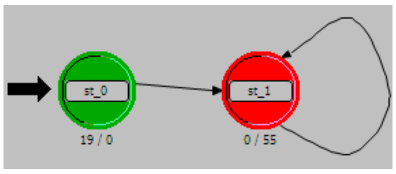

Fig. 11. Process_model_07 STM

- It calculates the time for the next data-packet generation according to the given average rate and distribution and schedules an interruption for itself at the calculated time.
- For every established outgoing link, it creates a data packet, encapsulates the packet into another packet with the control fields, sets the control fields (i.e., command and outgoing-link ID), and sends the packet to queue module phys_neighbor_list.

However, process_model_07 does not generate any data packets during the initialization period. In the initial state (i.e., state st_0 in Fig. 11), it schedules the first data packet generation at the end of the initialization period.

Also, process_model_07 performs the following operations, defined in state st_1, every time a packet is received from the input stream.

- It reads the command field from the packet. If the packet is a new-link-establishment notification, it reads the new outgoing-link ID and adds it to a list of established outgoing links.
- If the packet is a data packet, it destroys it.

The process-model configuration is shown in Table 4 (see Appendix I). The global attribute initialization_period specifies the length of the initialization.

3.1.7. Time_ref

This process model defines the functionality of processor module TIME_REF. It is shown in Fig.12. Its main task is to synchronize the node with the network's frame structure shown in Fig. 1. Therefore, it does not

[37] The command field can be used for specifying actions different from data-transmission requests, but in the simulation model, only such action is implemented.

process any packets nor has any input or output streams. It uses periodic self-interruptions that are scheduled at the onset of every minislot. At each of these interruptions, it interrupts other modules in the node to indicate the start of the current minislot and passes to each of them the timing information of the current minislot.

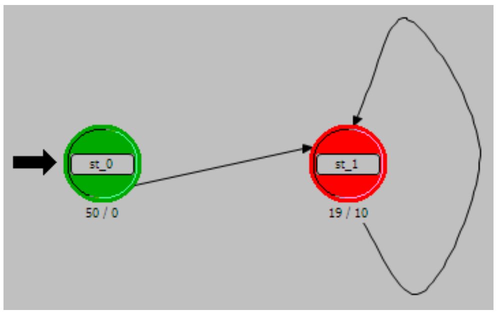

Fig. 12. time_ref STM

In the initial state (i.e., state st_0 in Fig. 12), time_ref calculates the duration of frames and minislots based on the simulation attributes that specify the frame structure timing (see Fig. 1). Also, it creates an OPNET Interface Control Information (ICI) structure for passing the current timing information to all the modules it interrupts. This ICI contains the following information.

- The number of consecutive scheduling-frames between every two network-configuration frames
- The total number of minislots per frame
- The number of minislots per control subframe
- The number of minislots per data subframe
- The current control-subframe type (i.e, network configuration, scheduling)
- The current frame number
- The current minislot number

Then, in the enter executives of state st_1, time_ref schedules interruptions at the current simulation time (i.e., at the onset of the current minislot) for the following queue modules: phys_neighbor_list, Q_4_1, Q_4_2, …, and Q_4_10. It calculates the time instant of the onset of the next minislot and schedules a self-interruption at such time.

When the self-interruption occurs, time-ref updates the current control-subframe type, frame number, and minislot in the ICI. This is done in the exit executives of state st_1. Finally, time_ref returns to st_1 by executing the enter executives once again, and this cycle repeats for every minislot covered by the total simulation time.

The process-model configuration is shown in Table 5 (see Appendix I).

Appendix I. Process-configuration parameters.

Table 1. process_model_01 configuration

Item	Name	Value
Model Attributes	none	none
Process Interfaces	begsim intrpt	enabled
Local Statistics	num_neighbors	single
	input_queue_length	single
	data_pk_delay	single
	throughput_pks_per_sec	single
	throughput_bits_per_sec	single
	num_onehop_neighbors	single
	num_sched_timeouts	single
	sched_handshake_delay	single
	sched_ctrl_subframe_access_delay	single
	num_rx_data_pks_without_colls	single
	num_rx_data_pks_with_colls	single
Global Statistics	num_neighbors	single
	input_queue_length	single
	data_pk_delay	single
	throughput_pks_per_sec	single
	throughput_bits_per_sec	single
	num_onehop_neighbors	single
	num_sched_timeouts	single
	sched_handshake_delay	single
	sched_ctrl_subframe_access_delay	single
	num_rx_data_pks_without_colls	single
	num_rx_data_pks_with_colls	single
Input Statistics	none	none
Global Attributes	num_channels	integer
	frame_length_code	integer
	MSH_CTRL_LEN	integer
	scheduling_frames	integer
	physical_channel0_center_frequency	double
	physical_channel_width	integer
	burst_profile	integer
	initialization_period	double
Output Packet Streams	CH0_NCFG_OUTPUT_STREAM_INDEX	0
	CH1_NCFG_OUTPUT_STREAM_INDEX	1
	CH2_NCFG_OUTPUT_STREAM_INDEX	2
	CH3_NCFG_OUTPUT_STREAM_INDEX	3
	CH4_NCFG_OUTPUT_STREAM_INDEX	4
	CH5_NCFG_OUTPUT_STREAM_INDEX	5
	CH6_NCFG_OUTPUT_STREAM_INDEX	6
	CH7_NCFG_OUTPUT_STREAM_INDEX	7
	CH8_NCFG_OUTPUT_STREAM_INDEX	8
	CH9_NCFG_OUTPUT_STREAM_INDEX	9
	CH0_DSCH_OUTPUT_STREAM_INDEX	10
	CH1_DSCH_OUTPUT_STREAM_INDEX	11
	CH2_DSCH_OUTPUT_STREAM_INDEX	12
	CH3_DSCH_OUTPUT_STREAM_INDEX	13
	CH4_DSCH_OUTPUT_STREAM_INDEX	14
	CH5_DSCH_OUTPUT_STREAM_INDEX	15
	CH6_DSCH_OUTPUT_STREAM_INDEX	16
	CH7_DSCH_OUTPUT_STREAM_INDEX	17
	CH8_DSCH_OUTPUT_STREAM_INDEX	18
	CH9_DSCH_OUTPUT_STREAM_INDEX	19
	CH0_DATA_OUTPUT_STREAM_INDEX	20
	CH1_DATA_OUTPUT_STREAM_INDEX	21
	CH2_DATA_OUTPUT_STREAM_INDEX	22
	CH3_DATA_OUTPUT_STREAM_INDEX	23
	CH4_DATA_OUTPUT_STREAM_INDEX	24
	CH5_DATA_OUTPUT_STREAM_INDEX	25
	CH6_DATA_OUTPUT_STREAM_INDEX	26
	CH7_DATA_OUTPUT_STREAM_INDEX	27
	CH8_DATA_OUTPUT_STREAM_INDEX	28
	CH9_DATA_OUTPUT_STREAM_INDEX	29
	P_7_OUTPUT_STREAM_INDEX	30
Input Packet Streams	CH0_NCFG_INPUT_STREAM_INDEX	0
	CH1_NCFG_INPUT_STREAM_INDEX	1
	CH2_NCFG_INPUT_STREAM_INDEX	2
	CH3_NCFG_INPUT_STREAM_INDEX	3
	CH4_NCFG_INPUT_STREAM_INDEX	4

	CH5_NCFG_INPUT_STREAM_INDEX CH6_NCFG_INPUT_STREAM_INDEX CH7_NCFG_INPUT_STREAM_INDEX CH8_NCFG_INPUT_STREAM_INDEX CH9_NCFG_INPUT_STREAM_INDEX CH0_DSCH_INPUT_STREAM_INDEX CH1_DSCH_INPUT_STREAM_INDEX CH2_DSCH_INPUT_STREAM_INDEX CH3_DSCH_INPUT_STREAM_INDEX CH4_DSCH_INPUT_STREAM_INDEX CH5_DSCH_INPUT_STREAM_INDEX CH6_DSCH_INPUT_STREAM_INDEX CH7_DSCH_INPUT_STREAM_INDEX CH8_DSCH_INPUT_STREAM_INDEX CH9_DSCH_INPUT_STREAM_INDEX CH0_DATA_INPUT_STREAM_INDEX CH1_DATA_INPUT_STREAM_INDEX CH2_DATA_INPUT_STREAM_INDEX CH3_DATA_INPUT_STREAM_INDEX CH4_DATA_INPUT_STREAM_INDEX CH5_DATA_INPUT_STREAM_INDEX CH6_DATA_INPUT_STREAM_INDEX CH7_DATA_INPUT_STREAM_INDEX CH8_DATA_INPUT_STREAM_INDEX CH9_DATA_INPUT_STREAM_INDEX P_7_INPUT_STREAM_INDEX	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Subqueues	Physical-neighbor-list BS-list Grant-list Input-queue-list Request-list Request-timer-list	0 to MAX_NUM_CHANNELS-1 MAX_NUM_CHANNELS to 2*MAX_NUM_CHANNELS-1 2*MAX_NUM_CHANNELS to 3*MAX_NUM_CHANNELS-1 3*MAX_NUM_CHANNELS to 3*MAX_NUM_CHANNELS+MAX_NUM_LINK_IDs-1 3*MAX_NUM_CHANNELS+MAX_NUM_LINK_IDs 3*MAX_NUM_CHANNELS+MAX_NUM_LINK_IDs+1 to 4*MAX_NUM_CHANNELS+MAX_NUM_LINK_IDs

Table 2. process_model_04 configuration

Item	Name	Value
Model Attributes	none	none
Process Interfaces	begsim intrpt	enabled
Local Statistics	data_pk_tx_minislot_num output_queue_length	single single
Global Statistics	output_queue_length	single
Input Statistics	none	none
Global Attributes	num_channels	integer
Output Packet Streams	OUTPUT_STREAM	0
Input Packet Streams	NCFG_INPUT_STREAM DSCH_INPUT_STREAM DATA_INPUT_STREAM	0 1 2
Subqueues	NCFG_SUBQUEUE_INDEX NCFG_SCHED_SUBQUEUE_INDEX DSCH_SUBQUEUE_INDEX DSCH_SCHED_SUBQUEUE_INDEX PAYLOAD_SUBQUEUE_INDEX PAYLOAD_SCHED_SUBQUEUE_INDEX OLD_NCFG_PKS_SUBQUEUE_INDEX OLD_DSCH_PKS_SUBQUEUE_INDEX OLD_DATA_PKS_SUBQUEUE_INDEX	0 1 2 3 4 5 6 7 8

Table 3. process_model_05 configuration

Item	Name	Value
Model Attributes	none	none
Process Interfaces	begsim intrpt	enabled
Local Statistics	num_rx_DSCH_pks_without_colls num_rx_DSCH_pks_with_colls num_rx_NCFG_pks_without_colls num_rx_NCFG_pks_with_colls	single single single single
Global Statistics	num_rx_DSCH_pks_without_colls num_rx_DSCH_pks_with_colls num_rx_NCFG_pks_without_colls num_rx_NCFG_pks_with_colls	single single single single
Input Statistics	none	none
Global Attributes	none	none
Output Packet Streams	NCFG_outstrm_index DSCH_outstrm_index payload_outstrm_index	0 1 2
Input Packet Streams	none	0
Subqueues	none	none

Table 4. process_model_07 configuration

Item	Name	Value
Model Attributes	none	none
Process Interfaces	begsim intrpt	enabled
Local Statistics	num_tx_data_pks num_rx_data_pks	single single
Global Statistics	none	none
Input Statistics	NUM_NEIGHS_INPUT_LOCAL_STAT_INDEX	1
Global Attributes	initialization_period	double double
Output Packet Streams	SAP_OUTPUT_STRM_INDEX	0
Input Packet Streams	SAP_INPUT_STRM_INDEX	0
Subqueues	none	none

Table 5. Time_ref configuration

Item	Name	Value
Model Attributes	none	none
Process Interfaces	begsim intrpt	enabled
Local Statistics	none	none
Global Statistics	none	none
Input Statistics	none	none
Global Attributes	frame_length_code MSH_CTRL_LEN scheduling_frames physical_channel_width	integer integer integer integer
Output Packet Streams	none	None
Input Packet Streams	none	none
Subqueues	none	none

Appendix II. Node attributes.

Table 6. Node attributes

Attribute name	Type	Value
is_node_a_BS	integer	No
node_ID	integer	1
radius_interference	double	1.0
radius_tx	double	1.0
XmtPower	integer	38
NextXmtMx	integer	3
XmtHoldoffExponent	integer	4
Min number of data pks to start sched	integer	10
Max number of sched minislots per frame per link	integer	32
Destination nodes list	string	0
Data packet arrival rates	string	0
Data-packet-generation distribution	string	exponential

References

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

[2] OPNET Technologies, OPNET Modeler. Network Simulation Software. URL: http://www.opnet.com