Computer and Information Science; Vol. 11, No. 1; 2018
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

A Simulation Model of IEEE 802.15.4 GTS Mechanism and GTS Attacks in OMNeT++ / MiXiM + NETA

Yasmin M. Amin[1] & Amr T. Abdel-Hamid[2]

[1] Former Teaching Assistant, Department of Networks Engineering, German University in Cairo, Cairo, Egypt XtremIO Storage Senior Recovery Analyst, Remote Reactive Services, Dell EMC, Cairo, Egypt

[2] Associate Professor, Department of Networks Engineering, German University in Cairo, Cairo, Egypt Senior Member, IEEE, Egypt

Correspondence: Yasmin M. Amin, Remote Reactive Services, Dell EMC, Cairo, Egypt. Tel: 20-10-9001-1344. E-mail: yasminmahmoudamin@hotmail.com

Received: December 21, 2017 Accepted: December 31, 2017 Online Published: January 27, 2018

doi:10.5539/cis.v11n1p78 URL: http://dx.doi.org/10.5539/cis.v11n1p78

Abstract

The IEEE 802.15.4 standard defines the PHY and MAC layer specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). With the proliferation of many time-critical applications with real-time delivery, low latency, and/or specific bandwidth requirements, Guaranteed Time Slots (GTS) are increasingly being used for reliable contention-free data transmission by nodes within beacon-enabled WPANs. To evaluate the performance of the 802.15.4 GTS management scheme, this paper introduces a new GTS simulation model for OMNeT++ / MiXiM. Our GTS model considers star-topology WPANs within the 2.4 GHz frequency band, and is in full conformance with the IEEE 802.15.4 – 2006 standard. To enable thorough investigation of the behaviors and impacts of different attacks against the 802.15.4 GTS mechanism, a new GTS attacks simulation model for OMNeT++ is also introduced in this paper. Our GTS attacks model is developed for OMNeT++ / NETA, and is integrated with our GTS model to provide a single inclusive OMNeT++ simulation model for both the GTS mechanism and all known-to-date attacks against it.

Keywords: IEEE 802.15.4, simulation model, OMNeT++, GTS management scheme, MiXiM, GTS attacks, NETA

1. Introduction

The IEEE 802.15.4 standard defines the Physical (PHY) and Media Access Control (MAC) layer specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). LR-WPANs target a diverse set of industrial, residential, and medical applications with very low power consumption and cost requirements, short ranges, very small sizes of wireless devices, and relaxed data rate and Quality of Service (QoS) requirements. LR-WPANs conceive a communications range of up to 10 meters in all directions, and a maximum data transfer rate of 250 kbps for operation within the 2.4GHz frequency band (Ergen, 2015).

There exist a number of well-known network simulators in the research community, including ns-2, ns-3, OMNeT++, SWAN, OPNET, Jist, and GloMoSiM, among others (Khan, Bilal & Othman, 2012). (Khan, Bilal & Othman, 2012) provides an extensive performance comparison between these network simulators. For the work in this paper, the OMNeT++ simulator was opted. OMNeT++ is a C++ object-oriented simulation library and framework primarily used for building network simulators. This network simulation platform provides an extensible, modular, and component-based architecture for building modelling frameworks, thus enabling the modelling and simulation of a diverse range of networks, including sensor networks and wireless ad-hoc networks, to name a few (Note 1). OMNeT++ uses discrete event simulation, in which state changes are triggered by events which happen at discrete instants of time, and events take zero time to happen (Note 2).

This paper presents a new IEEE 802.15.4 simulation model in OMNeT++ which integrates two models; a model for the 802.15.4 Guaranteed Time Slots (GTS) management scheme, and a model for the different GTS attack variants known to date. *To the extent of our knowledge*, our GTS model is the first to implement operation of the GTS management scheme in such significant level of detail. *We also believe that* our GTS attacks model is the first simulation model to implement all variants of the GTS attack which are known to date in existing literature.

Our GTS model is built by extending the existing MiXiM framework in OMNeT++, and implements beacon transmission and reception, PAN association, GTS slot allocation, manual GTS slot deallocation, and data transmission during GTS transmit slots. To build our GTS attacks model, OMNeT++'s existing NETA framework is extended. Our GTS attacks model implements five GTS attack variants; *Denial of Service (DoS) against Data Transmissions during Contention Free Period (CFP), False Data Injection, DoS against GTS Requests, Stealing Network Bandwidth, and Interference during CFP* (Amin & Abdel-Hamid, 2015).

The remainder of this paper is organized as follows. Section II begins with an overview of the MiXiM framework, and focuses on the specifications, limitations, architecture, folders structure, parameters, and simulation of our GTS model. This section (Section II) also presents a detailed comparison between our GTS model and other existing IEEE 802.15.4 simulation models in OMNeT++. Section III first gives an overview of the NETA framework, and then follows the same approach as Section II in detailing our GTS attacks model. Integration of both models is also explained in Section III. Finally, Section IV concludes the paper and provides insight into the future work.

2. The IEEE 802.15.4 GTS Simulation Model

Our IEEE 802.15.4 GTS simulation model is in full compliance with the IEEE 802.15.4 – 2006 standard (Note 3), and is developed by extending the existing MiXiM framework in OMNeT++.

2.1 Overview of MiXiM Framework

MiXiM is a merger of several OMNeT++ modeling frameworks, and was created with the specific intent of modeling fixed and mobile wireless networks, including wireless sensor networks, body area networks, ad-hoc networks, and vehicular networks, to name a few. MiXiM includes detailed models of various aspects of such networks, including radio transceiver power consumption and wireless MAC protocols, to name a few (Note 4). Our GTS model is developed using MiXiM 2.3.

2.2 Model Specifications

Our GTS model specifically considers IEEE 802.15.4 star-topology beacon-enabled PANs consisting of one co-ordinator and multiple members. Only operation within the 2.4 GHz frequency band is considered.

The following MAC layer specifications of the IEEE 802.15.4 standard are implemented:

- Beacon transmission, and reception
- PAN association
- GTS allocation
- Manual GTS deallocation
- Data transmission (by members) during GTS transmit slots

2.3 Model Limitations

Our GTS model does not consider peer-to-peer and cluster-tree topologies, nor does it consider beacon-less mode of operation.

The following MAC layer specifications are not included:

- PAN disassociation
- CSMA-CA protocol
- Automatic GTS deallocation
- Data reception (by co-ordinator) during GTS transmit slots
- Data transmission (by members) and reception (by co-ordinator) during assigned GTS receive slots

The reader is referred to the standard (Note 3) for a detailed explanation of all IEEE 802.15.4 MAC layer specifications.

2.4 Model Architecture and Folders Structure

This subsection follows a top-down approach in describing the architectural components of our GTS model within the MiXiM framework.

2.4.1 ***Nodes***

Two types of network devices (nodes) are defined within the model; ***Host802154Coordinator*** and ***Host802154Member***.

The *NED* files of both modules, as well as their parent and reference modules, are located in ***/MiXiM/src/modules/node***.

a) ***Host802154Coordinator***: Represents a PAN co-ordinator. This compound module extends MiXiM's existing ***WirelessNodeBattery*** module, and is composed of the submodules shown in Figure 1. This module's inherited *nicType* parameter is set to ***Nic802154Coordinator***. The existing ***Host802154_2400MHz*** module is used as a reference when setting the values of this module's remaining parameters.

b) ***Host802154Member***: Represents a PAN member. This compound module is similar to the ***Host802154Coordinator*** module, except that its *nicType* parameter is set to ***Nic802154Member***.

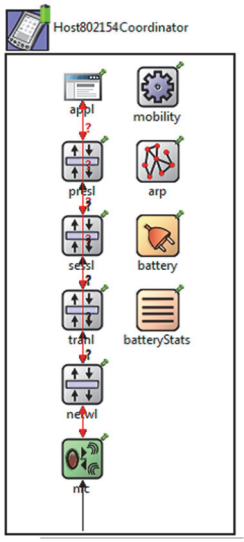

Figure 1. Submodules of ***Host802154Coordinator*** module

2.4.2 ***NICs***

PAN co-ordinators and members require different Network Interface Cards (NICs). As such, two types of NICs are defined within the model; Two types of network devices (nodes) are defined within the model; ***Nic802154Coordinator*** and ***Nic802154Member***.

The *NED* files of both NIC modules, as well as their parent and reference modules, are located in ***/MiXiM/src/modules/nic***.

a) ***Nic802154Coordinator***: Represents the NIC of a PAN co-ordinator. This compound module extends MiXiM's existing ***WirelessNicBattery*** module, and is composed of the submodules illustrated in Figure 2. This module's inherited *macType* parameter is set to ***Mac802154Coordinator***. In setting the values of this module's remaining parameters, the existing ***Nic802154_TI_CC2420*** module is used as a reference.

b) ***Nic802154Member***: Represents the NIC of a PAN member. This compound module is similar to the ***Nic802154Coordinator*** module, except that its *macType* parameter is set to ***Mac802154Member***.

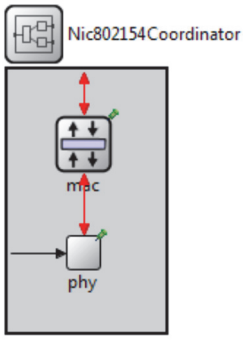

Figure 2. Submodules of ***Nic802154Coordinator*** module

2.4.3 ***MAC***

PAN co-ordinators and members exhibit different behaviors on the MAC level. As such, two types of MAC behaviors are defined within the model; ***Mac802154Coordinator*** and ***Mac802154Member***.

The *NED*, *C++*, and *header* files of both modules are located in ***/MiXiM/src/modules/mac***.

a) ***Mac802154Coordinator***: Represents the behavior of a PAN co-ordinator on the MAC level. This simple module extends the existing ***BaseMacLayer*** module located in ***/MiXiM/src/base/modules***.

b) ***Mac802154Member***: Represents the behavior of a PAN member on the MAC level. Similar to ***Mac802154Coordinator***, this simple module also extends the ***BaseMacLayer*** module.

In addition to these two MAC modules, two additional simple modules are implemented; **ASSOC** and **GTS**. **ASSOC** contains the functions which implement the PAN association specification, whereas **GTS** contains the functions responsible for both GTS allocation and deallocation. These functions are called from within the ***Mac802154Coordinator*** and ***Mac802154Member*** modules. The *C++* and *header* files of **ASSOC** and **GTS** are also located in ***/MiXiM/src/modules/mac***.

2.4.4 Messages

Represent the frames exchanged between PAN co-ordinators and members. Different types of frames have different formats, fields, and default field values, as specified by the standard (Note 3). Our model defines six types of messages; ***Beacon***, ***AssocRequest***, ***AssocResponse***, ***GtsRequest***, ***Data***, and ***Acknowledgment***.

All implemented messages extend the existing ***MacPkt*** message, which is located in ***/MiXiM/src/base/messages***. Fields consisting of several parameters are implemented as C++ data structures (*struct*) within each message. The *message*, *C++*, and *header* files of all messages are located in ***/MiXiM/src/modules/messages***.

a) ***Beacon***: Represents a beacon frame periodically transmitted by a PAN co-ordinator to all members within its PAN.

b) ***AssocRequest***: Represents an association request frame transmitted by a node requesting PAN association to a PAN co-ordinator.

c) ***AssocResponse***: Represents an association response frame transmitted by a PAN co-ordinator to accept or reject a requesting node's PAN association request.

d) ***GtsRequest***: Represents a GTS allocation or deallocation request, depending on the value of the message's *characteristicsType* parameter.

e) ***Data***: Represents a data frame transmitted by a PAN member to a PAN co-ordinator. This frame contains mission-critical information such as temperature or power consumption readings, depending on the type and purpose of the network.

f) ***Acknowledgment***: Represents an acknowledgment frame used to indicate successful frame transmission between two nodes.

While the aforementioned messages represent the packets transmitted between the co-ordinator and member modules, additional messages are implemented to be used as self-messages, which are sent by a module to itself to schedule the occurrence of certain events at specific instants of time. Self-message instances start with a *g* and end with the term *Event*, such as *gAckEvent* and *gGtsTransmitEvent*.

2.5 Model Simulation

To test our GTS model's behavior, a new folder called ***ieee802154*** is created within the ***/MiXiM/examples*** directory. This folder contains the *NED* file of the ***ieee802154*** example network, which extends ***/MiXiM/src/base/modules/BaseNetwork***. The folder also contains the *omnetpp.ini* simulation file, where the values of the parameters mentioned in the following subsection are set. The ***ieee802154*** network is a star-topology network which consists of 4 submodules, as shown in Figure 3; 1 ***Host802154Coordinator*** node called *coordinator*, and 3 ***Host802154Member*** nodes called *member1*, *member2*, and *member3*.

2.6 Model Parameters

A number of parameters can be varied to simulate different aspects of the 802.15.4 GTS management scheme.

Parameters which are not assigned default values in the *NED* files of the ***Mac802154Member*** and ***Mac802154Coordinator*** modules must be set by the user in the *omnetpp.ini* file. These mandatory parameters are listed in Table 1 with respect to each parameter's name and the module in which each parameter appears.

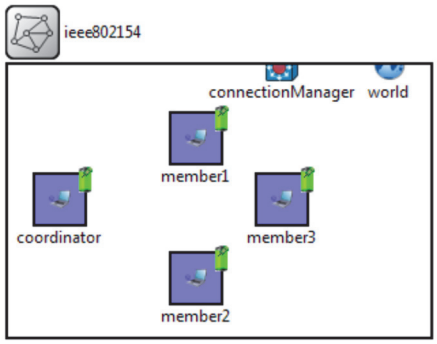

Figure 3. Submodules in ***ieee802154*** network

Table 1. Mandatory parameters in *omnetpp.ini* file

Name	Module
address	***Mac802154Coordinator*** ***Mac802154Member***
modeId *panId* *macBeaconOrder* *macSuperframeOrder* *aMinCapLength*	***Mac802154Coordinator***
assocOrder	***Mac802154Member***

As illustrated in Table 2, the value of the *nodeId* parameter can be manually set in the *omnetpp.ini* file, but only for the co-ordinator, and not for the members. This is because the values of the members' *nodeId* parameters are dynamically assigned by the co-ordinator after successful PAN association. Node ID assignment is implemented in the *C++* files of the ***Mac802154Coordinator*** and ***Mac802154Member*** modules.

All of the parameters shown in Table 1 have self-explanatory names, except for the *address* and *assocOrder* parameters. The *address* parameter is used to represent node MAC address, and the value of the *assocOrder* parameter is set to the index of the Contention Access Period (CAP) slot at which an ***AssocRequest*** message is scheduled for transmission to the targeted PAN co-ordinator. For the latter, CAP slot index is used to calculate the simulation time instant, in seconds, at which association request transmission occurs. To ensure that no collisions occur between the ***AssocRequest*** messages of different requesting nodes, no two nodes should be assigned the same values for their *assocOrder* parameters.

Parameters which are already assigned default values in the *NED* files of the ***Mac802154Member*** and ***Mac802154Coordinator*** modules can optionally be overridden in the *omnetpp.ini* file, but doing so is not mandatory. These optional parameters are listed, along with their default values, in Table 2.

The IEEE 802.15.4 standard specifies that a superframe's active period is subdivided into 16 equally sized superframe slots. For this reason, *aNumSuperframeSlots* is set to 16. While the value given for the *aBaseSuperframeDuration* parameter in Table 2 is not explicitly stated in the standard (Note 3), the calculation of *aBaseSuperframeDuration* is straightforward. The standard (Note 3) specifies the values of the *aBaseSlotDuration* and *aNumSuperframeSlots* parameters as 60 and 16 symbols respectively. The following equation for calculating the value of *aBaseSuperframeDuration* in symbols is also specified in the standard (Note 3):

$$\begin{aligned} aBaseSuperframeDuration &= aBaseSlotDuration \times aNumSuperframeSlots = 60 \text{ symbols} \times 16 \text{ slots} \\ &= 960 \text{ symbols} \end{aligned} \quad (1)$$

According to (Jurcik et al., 2007), 1 symbol corresponds to 4 bits during operation within the 2.4 GHz frequency band. As such, the value of *aBaseSuperframeDuration* in bits is calculated as:

aBaseSuperframeDuration = 960 symbols x 4 bits per symbol = 3840 bits (2)

Since bit rate within the 2.4 GHz frequency band is equal to 250 kilobits per second (which is the value of the *bitrate* parameter shown in Table 2), the default value of *aBaseSuperframeDuration* in seconds is calculated as:

aBaseSuperframeDuration = 3840 bits / 250 kilobits per seconds = 15.36 milliseconds (3)

Table 2. Optional parameters in *omnetpp.ini* file

Name	Module	Default value
aNumSuperframeSlots	***Mac802154Coordinator*** ***Mac802154Member***	16
aBaseSuperframeDuration		15.36 ms
bitrate		250000 bps
txPower		1 mW
gtsTransmitRequestOrder *numGtsTransmitSlots* *gtsReceiveRequestOrder* *numGtsReceiveSlots* *nextReading*	***Mac802154Member***	0
gtsTransmitAllocateSuperframeIndex *gtsTransmitDeallocateSuperframeIndex* *gtsReceiveAllocateSuperframeIndex* *gtsReceiveDeallocateSuperframeIndex*		-1
recurrentDataTransmission		false
dataTransmissionIntervalInSuperframes		1
dataFilename		""

Although possible, overriding the default values of the *aNumSuperframeSlots* and *aBaseSuperframeDuration* parameters in the *omnetpp.ini file* is not recommended, as the default values of these parameters are already set to the IEEE 802.15.4 standard-specific values in the *NED* files of the ***Mac802154Member*** and ***Mac802154Coordinator*** modules. Furthermore, if the value of the *bitrate* parameter is overridden in the *omnetpp.ini* file, the *aBaseSuperframeDuration* parameter should be recomputed using the newly selected bit rate, as demonstrated by equation (3).

Similar to the ***Mac802154Member*** module's *assocOrder* parameter is its *gtsTransmitRequestOrder* parameter, which is optionally set in the *omnetpp.ini* file to the index of the CAP slot at which a ***GtsRequest*** message is scheduled for transmission to the co-ordinator. As is the case with the *assocOrder* parameter, no two member nodes should be assigned the same value for the *gtsTransmitRequestOrder* parameter in the *omnetpp.ini* file. The *numGtsTransmitSlots* parameter is set to the number of superframe slots requested by a node within its assigned GTS transmit slot. Although this value is determined by the node's application layer in reality, we include it in the *omnetpp.ini* file so that it can be determined by the user prior to simulation due to the absence of an application layer implementation. The *gtsReceiveRequestOrder* and *numGtsReceiveSlots* parameters possess the same significance as the *gtsTransmitRequestOrder* and *numGtsTransmitSlots* parameters respectively, but for GTS receive slots instead.

The *gtsTransmitAllocateSuperframeIndex* parameter is used to determine the frequency with which allocation requests for a GTS transmit slot are transmitted by a member node to the co-ordinator, and is also set by the node's application layer in reality. A default value of -1 signifies that a ***GtsRequest*** message is transmitted to the co-ordinator every time that the member node receives a ***Beacon*** message in which no GTS transmit slot is allocated to this member. However, if *gtsTransmitAllocateSuperframeIndex* is set to a specific value, then the member node transmit a ***GtsRequest*** message requesting the allocation of a GTS transmit slot if and only if it receives a ***Beacon*** message with a *sequenceNumber* which is equal to the value of *gtsTransmitAllocateSuperframeIndex*. The *gtsTransmitDeallocateSuperframeIndex* serves the same purpose, but is used for manual deallocation of GTS transmit slots. Similar to *gtsTransmitAllocateSuperframeIndex* and *gtsTransmitDeallocateSuperframeIndex* are the *gtsReceiveAllocateSuperframeIndex* and *gtsReceiveDeallocateSuperframeIndex* parameters respectively, which possess the same significance for GTS receive slots.

To determine how regularly a member node transmits its data during its assigned GTS transmit slot, the

recurrentDataTransmission parameter is used. If *recurrentDataTransmission* is set to true, then a new ***Data*** message is transmitted every number of superframes equal to the value of the *dataTransmissionIntervalInSuperframes* parameter. If, however, *recurrentDataTransmission* is false, then a new ***Data*** message is only transmitted when a ***Beacon*** message with *sequenceNumber* equal to (*gtsTransmitAllocateSuperframeIndex* + 1) is received.

Our implemented model offers two options for setting the *value* fields of transmitted ***Data*** messages. The first option is to read values from a data file by setting the value of the *dataFileName* parameter in the *omnetpp.ini* file to the name of the selected data file. In this case, the format of the chosen data file should be such that each line contains a single reading. The user should also ensure that the used data file contains enough readings for the entire duration of the simulation. The second option offered by our model, in the case that the *dataFileName* parameter contains an empty string, is to allow the user to set the value of the *nextReading* parameter in the *omnetpp.ini* file to a constant value, which is used to set the *value* of all generated and transmitted ***Data*** messages.

Parameters which are not discussed in this subsection are set to the same values as in the *omnetpp.ini* file of the existing reference ***/MiXiM/examples/ieee802154a*** network.

2.7 Existing IEEE 802.15.4 Simulation Models in OMNeT++

According to (Kirsche & Schnurbusch, 2015), the INETMANET, MiXiM, and Castalia modelling frameworks offered by OMNeT++ already include simulation models for the IEEE 802.15.4 standard. The reader is referred to (Note 5) for more information on the INETMANET and Castalia frameworks. The INETMANET framework includes a port of the IEEE 802.15.4 simulation model implemented in (Chen & Dressler, 2007). This model conforms to the IEEE 802.15.4 – 2006 standard (Note 3), and includes a MAC module which covers beacon transmission and reception, simple PAN association, channel access using both slotted and unslotted CSMA-CA protocols, and GTS transfer mode for star and cluster-tree network topologies. The model does not implement PAN disassociation (Chen & Dressler, 2007). MiXiM's 802.15.4 simulation model includes a MAC model which only covers CSMA-CA protocol operation (Kirsche & Schnurbusch, 2015). (Note 6) clarifies that MiXiM 1.2, which is used as the basis for newer versions of MiXiM, covers CSMA-CA for non-beacon enabled mode. Thus, it can be inferred that only unslotted CSMA-CA is implemented, as slotted CSMA-CA is only present within the CAP period of beacon-enabled network mode. The authors of (Note 7) explicitly state that the 802.15.4 simulation model included within Castalia covers beacon-enabled PANs with association, both slotted and unslotted CSMA-CA, and GTS. Although no explicit statements regarding implementation of beacon transmission and reception are given, we infer from the text in (O'Flynn & Message, 2011) that beacon transmission and reception are covered by the implemented model. We also infer that PAN disassociation is not covered by the model, as there is no mention of this specification in (Note 7).

The simulation model implemented in (Kirsche & Schnurbusch, 2015) is based on OMNeT++'s INET framework, and covers beacon and WPAN management, Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA), and GTS, among others. Since the authors do not clarify exactly which beacon and WPAN management functions their implemented model covers, we infer that beacon management encompasses both beacon transmission and reception, and that WPAN management includes both PAN association and disassociation. However, due to the general use of the term CSMA-CA, and due to the fact that the authors of (Kirsche & Schnurbusch, 2015) do not clarify whether the implemented model covers beacon-less network mode of operation, we infer that slotted CSMA-CA is implemented, but no inferences can be made regarding unslotted CSMA-CA.

Table 3 compares between our GTS model and the four aforementioned IEEE 802.15.4 simulation models with respect to the underlying OMNeT++ framework used to build each model, and the relevant MAC layer specifications implemented by each model. For each model, √ is used to indicate specification presence, × indicates specification absence, and √√ indicates that presence of a particular specification is inferred from, but not explicitly stated by, the reference corresponding to the discussed model. Similar to √√, ×× signifies inferred specification absence.

3. The IEEE 802.15.4 GTS Attacks Simulation Model

Our IEEE 802.15.4 GTS attacks simulation model is a result of extending OMNeT++'s existing NETA framework.

3.1 Overview of NETA Framework

NETA, short for NETwork Attacks, is an OMNeT++ framework which was developed to provide researchers in

the field of network security with a means of simulating and evaluating the behaviors and impacts of different attacks in heterogeneous networks. In doing so, researchers are encouraged to develop and test the performance of possible defenses against the studied attacks (Note 8).

While NETA provides existing implementations for three network-layer attacks (IP dropping, IP delay, and sinkhole attacks), the framework's flexible design and modular architecture also facilitate the implementation and simulation of other attacks against different layers of the protocol stack (Note 9).

Our GTS attacks model is built through extension of version 1.0.0 of the NETA framework.

Table 3. Comparison between IEEE 802.15.4 simulation models in OMNeT++

Simulation model	Chen *et al.* [12]	MiXiM	Castalia	Kirsche *et al.* [10]	Our model
Underlying framework	INETMANET	MiXiM	Castalia	INET	MiXiM
Beacon management					
Beacon transmission	√	×	√√	√√	√
Beacon reception	√	×	√√	√√	√
PAN management					
PAN association	√	×	√	√√	√
PAN disassociation	×	×	××	√√	×
CSMA-CA					
Slotted	√	××	√	√√	×
Unslotted	√	√√	√	unknown	×
GTS					
-	√	×	√	√	√

3.2 Model Specifications and Limitations

The reader is referred to the GTS attack classification illustrated and explained in (Amin & Abdel-Hamid, 2015). Our GTS attacks model implements all of the GTS attack variants included in the classification, except for the *DoS against CAP Maintenance* variant.

3.3 Model Integration

The code for our GTS attacks model was initially included within the same project folder, named **netattacks**, as the NETA framework, and the **MiXiM** project folder was added as a project reference to the **netattacks** project. However, doing so resulted in several errors in the **netattacks** project. This is because the **netattacks** project already contains a patch for the **INET** framework on which **MiXiM** is based, which conflicts with the externally referenced **MiXiM** project.

To resolve this issue, the **netattacks** project was copied into the **src** folder of the **MiXiM** project, and the code for our GTS attacks model was included in ***/MiXiM/src/netattacks***. To sum up, the **MiXiM** project is now the unified location for both our GTS mechanism and GTS attacks models.

3.4 Model Architecture and Folders Structure

This subsection uses a top-down approach to describe the architectural components of our GTS attacks model, which are located in ***/MiXiM/src/netattacks***. Following the naming convention set by the NETA framework, the names of components relating to the attacks model are preceded with ***NA_***.

Since all of the attack variants implemented in our attacks model are launched by PAN members (not co-ordinators), only malicious versions of the ***Host802154Member***, ***Nic802154Member***, and ***Mac802154Member*** modules are created for the attacks model.

3.4.1 ***Nodes***

A new type of network node, ***NA_Host802154Member***, is created for our attacks model, and represents an attacker with malicious intent. This compound module represents a malicious version of the ***Host802154Member*** module by setting its *nicType* parameter to ***NA_Nic802154Member***.

The *NED* file of the ***NA_Host802154Member*** module is located in ***/MiXiM/src/netattacks/src/nodes***.

3.4.2 ***NICs***

The ***NA_Nic802154Member*** module is a malicious version of the ***Nic802154Member*** module. The *macType*

parameter of this compound module is set to ***NA_Mac802154Member***.

The *NED* file of the ***NA_Nic802154Member*** module is located in ***/MiXiM/src/netattacks/src/nic***.

3.4.3 ***Hacked Modules***

Within the NETA framework, modified versions of modules which are targeted for attack execution are referred to as hacked modules (Note 9). The behavior of the ***Mac802154Member*** module is modified to create the hacked ***NA_Mac802154Member*** module, which contains the modifications needed for the execution of all implemented GTS attack variants. This hacked module also extends NETA's existing ***NA_HackedModule*** module, which is located in ***/MiXiM/src/netattacks/src/hackedmodules***. The hacked ***NA_IPv4*** module within the original NETA framework is used as a reference during implementation of the hacked ***NA_Mac802154Member*** module.

The *NED*, *C++*, and *header* files of the ***NA_Mac802154Member*** module are located in ***/MiXiM/src/netattacks/src/hackedmodules/maclayer***.

3.4.4 ***Attack Controllers***

Responsible for controlling attack execution by defining the following five attack properties; *attackType*, *active*, *startTime*, *endTime*, and *attack specific parameters* (Note 9).

The *attackType* property is a string which is used to refer to the type of attack being executed. The boolean *active* property is used to determine whether the attack is active in the simulation or not. The time at which an executed attack starts and ends in the simulation is determined by the double values of the *startTime* and *endTime* properties respectively. Depending on the value of the *attackType* property, *attack specific parameters* are additional properties which vary for different types of launched attacks (Note 9).

Since our attacks model implements five variants of the GTS attack, five attack controllers are defined. The naming convention for each of these five attack controllers is ***NA_GtsAttackVx***, where ***x*** is replaced with the variant number, ranging from 1 to 5 inclusive. Each of the ***NA_GtsAttackVx*** attack controllers extends NETA's existing base ***NA_Attack*** attack controller, which defines all of the above listed attack properties. For each of the five attack controllers, the value of *attackType* is defined in the controller's corresponding *NED* file. The remaining attack properties (*active*, *startTime*, *endTime*, and *attack specific parameters*) are initialized within the simulation's *omnetpp.ini* file. In implementing the behavior of all five ***NA_GtsAttackVx*** attack controllers, NETA's existing ***NA_DroppingAttack*** controller module is used as a reference.

The *NED*, *C++*, and *header* files of all GTS attack controllers are located in ***/MiXiM/src/netattacks/src/attacks/controllers/gtsAttack/variantx***, where ***x*** ranges from 1 to 5. The parent ***NA_Attack*** module's *NED*, *C++*, and *header* files are located in ***/MiXiM/src/netattacks/src/attacks/controllers***.

3.4.5 ***Control Messages***

Activation and deactivation of executed attacks are achieved via direct transmission of control messages from attack controllers to hacked modules (Note 9). While the control messages already defined within the original NETA framework pass the values of the *attackType*, *active*, *startTime*, and *endTime* properties from attack controllers to hacked modules, the ***NA_GtsMessage*** control message is defined to pass the values of *additional attack specific parameters* from their corresponding ***NA_GtsAttackVx*** attack controllers to the ***NA_Mac802154Member*** module.

The ***NA_GtsMessage*** message file is located in ***/MiXiM/src/netattacks/src/attacks/controlmessages/gtsAttack***.

3.5 Model Simulation

For the purpose of testing our GTS attacks model's behavior, five folders are created for each of the five GTS attack variants within ***/MiXiM/src/netattacks/simulations/SimpleAttackScenarios/SimpleGtsAttack***. Each of these 5 folders is called ***VxScenario***, where ***x*** ranges from 1 to 5, depending on the variant number. Each ***VxScenario*** folder contains an *NED* file of an example network called ***SimpleGtsAttackVx***, which extends ***/MiXiM/src/base/modules/BaseNetwork*** and constitutes a *star*-topology network.

For the first four attack variants, ***SimpleGtsAttackVx*** (where ***x*** ranges from 1 to 4) consists of 4 submodules as shown in Figure 4; 1 ***Host802154Coordinator*** node called *CO*, 2 ***Host802154Member*** nodes called *LN1* and *LN2*, and 1 ***NA_Host802154Member*** node called *MN3*. The naming convention for each of these submodules is as follows; the first character is an *L* if the node represents a *legitimate* PAN member and an *M* if the node represents a *malicious* attacker, the second character is an *N* for *node*, and the third and final character is a number reflecting the ID assigned to each node during PAN association within the simulation.

For attack variant 5, a second *malicious* attacker, named *MN4*, is added to facilitate simulation of the *Two*

Intelligent Attackers (TIA) and *Two Random Attackers (TRA)* variants. Figure 5 illustrates the submodules forming the ***SimpleGtsAttackV5*** network.

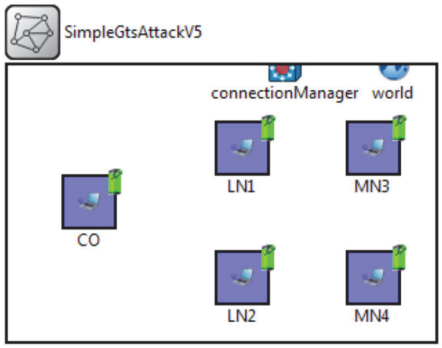

Figure 4. Submodules in ***SimpleGTSAttackVx* (*x* = 1 to 4)** network

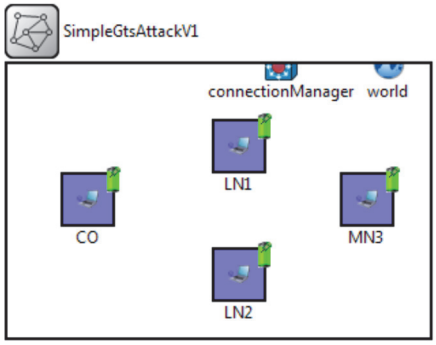

Figure 5. Submodules in ***SimpleGTSAttackV5*** network

3.6 Model Parameters

Table 4 illustrates the parameters set by the user in the *omnetpp.ini* file for each of the 5 implemented GTS attack variants prior to running the simulation. For each parameter, the Table illustrates the attack controller module in which the parameter resides, as well as the default value assigned to the parameter within its containing module's *NED* file.

Table 4. Attack parameters in *omnetpp.ini* file

Name	Module	Default value
active	***NA_GtsAttackVx* (*x* = 1 to 5)**	false
startTime		0 s
endTime		
gtsAttackProbability		0
gtsAttackV5Subtype	***NA_GtsAttackV5***	

In addition to the attack properties inherited by all five attack controllers from the parent ***NA_Attack*** controller, an additional *gtsAttackProbability* parameter is defined for all five controllers, and is set to a double value indicating the probability of attack execution.

For the ***NA_GtsAttackV5*** attack controller, two additional parameters are defined; *gtsAttackV5Subtype* and *gtsAttackV5S1Max*. If *gtsAttackV5Subtype* is set to 1, this means that a *One Intelligent Attacker (OIA)* or *TIA* attack is being executed. A value of 2 for *gtsAttackV5Subtype* indicates the execution of a *One Random Attacker*

(ORA) or *TRA* attack. In case *gtsAttackV5Subtype* is set to 1, it is necessary to determine whether the attack being launched is an *OIA* or *TIA*. In the event of a *TIA*, it is also important to determine whether the current attacker will be targeting the largest or second largest assigned GTS slot (in terms of number of constituent superframe slots). For this purpose, the *gtsAttackV5S1Max* parameter is used, for which a value of 1 indicates that the attacker will target the GTS slot of maximum length, and a value of 2 signifies that the attacker will interfere with the GTS slot of the second largest length.

4. Conclusions and Future Work

In this paper, we present a novel IEEE 802.15.4 simulation model in OMNeT++ which integrates two models; a GTS model and a GTS attacks model. Our GTS model conforms to the IEEE 802.15.4 – 2006 standard, and implements beacon transmission and reception, PAN association, GTS allocation, manual GTS deallocation, and data transmission by members during their assigned GTS transmit slots. In the future, missing MAC layer specifications which are highlighted in the model limitations' subsection will be implemented to provide a more complete simulation model of the IEEE 802.15.4 standard. Our GTS attacks model implements five GTS attack variants, which are *DoS against Data Transmission during CFP*, *False Data Injection*, *DoS against GTS Requests*, *Stealing Network Bandwidth*, and *Interference during CFP*. However, some of these variants' implementations contain slight deviations from their true behaviors in existing literature. The modifications required to eliminate these deviations are left for future work.

5. Acquiring Model Access

Our simulation model is not yet available for online access. Readers who are interested in acquiring model access are encouraged to contact us using the e-mail address listed under the paper title.

References

Amin, Y. M. (2016). Anomaly-Based Intrusion Detection for Home Area Networks in Smart Grids (Unpublished master's thesis). German University in Cairo, Cairo, Egypt.

Khan, A. R., Bilal, S. M., & Othman, M. (2012). A Performance Comparison of Open Source Network Simulators for Wireless Networks, IEEE International Conference on Control System, Computing and Engineering (ICCSCE), pp. 34-38, November 2012.

Amin, Y. M., & Abdel-Hamid, A. T. (2015). Classification and Analysis of IEEE 802.15.4 MAC Layer Attacks, IEEE 11th International Conference on Innovations in Information Technology (IIT'15), pp. 1-6, November 2015.

Jurcik, P., Koubaa, A., Alves, M., Tovar, E., & Hanzalek, Z. (2007). A Simulation Model for the IEEE 802.15.4 protocol: Delay/Throughput Evaluation of the GTS Mechanism, IEEE 15th International Symposium on Modeling Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '07), pp. 109-116, October 2007.

Chen, F., & Dressler, F. (2007). A Simulation Model of IEEE 802.15.4 in OMNeT++, 2007. Retrieved from www.ccs-labs.org/bib/chen2007simulation/chen2007simulation.pdf

Ergen, S. C. (2015). ZigBee/IEEE 802.15.4 Summary, September 2004. Retrieved April, 2015, from http://users.ece.utexas.edu/~valvano/EE345L/Labs/Fall2011/Zigbeeinfo.pdf

Kirsche, M., & Schnurbusch, M. (2015). A New IEEE 802.15.4 Simulation Model for OMNeT++ / INET. Retrieved September 2015, from arxiv.org/pdf/1409.1177.pdf

O'Flynn, C. P., & Message, D. (2011). Alteration on IEEE 802.15.4 Low-Power Radio Networks, 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1-5, February 2011. https://doi.org/10.1109/NTMS.2011.5720580

Notes

Note 1. OMNeT++ Discrete Event Simulator – Introduction, n.d., Retrieved from https://omnetpp.org/intro

Note 2. OMNeT++ User Manual – Version 4.6, n.d., Retrieved from https://omnetpp.org/doc/omnetpp/manual/usman.html#sec138.

Note 3. IEEE, 802.15.4-2006 – IEEE Standard for Information technology – Local and metropolitan area networks—Specific requirements—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs), pp. 1-320, September 2006.

Note 4. MiXiM, n.d., Retrieved from http://mixim.sourceforge.net/

Note 5. OMNeT++ Discrete Event Simulator – Simulation Models, n.d., Retrieved from https://omnetpp.org/models.

Note 6. MiXiM 1.2 Released, n.d., Retrieved from https://omnetpp.org/9-articles/software/3669-mixim-12-released.

Note 7. National ICT Australia (NICTA), Castalia: A Simulator for Wireless Sensor Networks and Body Area Networks – User Manual – Version 3.2, March 2011, Retrieved from https://forge.nicta.com.au/docman/view.php/301/592/Castalia+-+User+Manual.pdf

Note 8. NETwork Attacks (NETA) Framework for OMNeT++ Released, n.d., Retrieved from https://omnetpp.org/component/content/article?id=3712:neta-release

Note 9. Network Engineering & Security Group (NESG), NETA: A NETwork Attacks Framework – Architecture and Usage. (October 8, 2013). Retrieved from http://nesg.ugr.es/NETA/doc/resources/frameworkDescription.pdf