

NetSim[®]

Accelerate Network R & D

Internet of Things (IoT) and Wireless Sensor Networks (WSN)

A Network Simulation & Emulation Software

By



The information contained in this document represents the current view of TETCOS LLP on the issues discussed as of the date of publication. Because TETCOS LLP must respond to changing market conditions, it should not be interpreted to be a commitment on the part of TETCOS LLP, and TETCOS LLP cannot guarantee the accuracy of any information presented after the date of publication.

This manual is for informational purposes only.

The publisher has taken care of the preparation of this document but makes no expressed or implied warranty of any kind and assumes no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained herein.

Warning! DO NOT COPY

Copyright in the whole and every part of this manual belongs to TETCOS LLP and may not be used, sold, transferred, copied or reproduced in whole or in part in any manner or in any media to any person, without the prior written consent of TETCOS LLP. If you use this manual, you do so at your own risk and on the understanding that TETCOS LLP shall not be liable for any loss or damage of any kind.

TETCOS LLP may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from TETCOS LLP, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Rev 14.0 (V), Oct 2023, TETCOS LLP. All rights reserved.

All trademarks are the property of their respective owner.

Contact us at

TETCOS LLP

214, 39th A Cross, 7th Main, 5th Block Jayanagar,
Bangalore - 560 041, Karnataka, INDIA.

Phone: +91 80 26630624

E-Mail: sales@tetcos.com

Visit: www.tetcos.com

Table of Contents

1	Introduction	5
2	Simulation GUI.....	7
2.1	Create Scenario	8
2.1.1	Fast Configuration.....	8
2.1.2	Wireless Sensor Networks	9
2.1.3	Internet of Things	10
2.1.4	Differences between IoT and WSN in NetSim	11
2.1.5	Device Attributes	12
2.2	Set Node, Link and Application Properties.....	19
2.2.1	Setting Static Routes.....	23
2.3	Enable Packet Trace, Event Trace & Plots (Optional).....	24
2.4	Run Simulation	24
3	Model Features	25
3.1	L3 Routing: DSR, OLSR, ZRP and AODV	25
3.2	L3 Routing: RPL Protocol	25
3.2.1	RPL Objective Function.....	26
3.2.2	Topology Construction	27
3.2.3	RPL Log File	29
3.2.4	Viewing RPL control messages in Wireshark	33
3.3	MAC / PHY: 802.15.4 Overview.....	34
3.3.1	CSMA/CA Implementation in NetSim	35
3.3.2	Beacon Order and Super Framer Order	37
3.4	Energy Models: Sources, Consumption and Harvesting	37
3.4.1	Energy Model source code.....	40
3.5	Sensor Application and how to model sensing interval?	40
3.6	WSN/IOT File Based Placement.....	41

3.6.1	Internet of Things	41
3.6.2	Wireless Sensor Networks	43
3.7	Radio measurements log file	45
3.7.1	Implementation details and Assumptions	47
3.8	Model Limitations	47
4	Featured Examples.....	48
4.1	IOT Example Simulations	48
4.1.1	Energy Model.....	48
4.1.2	Working of RPL Protocol in IoT	50
4.1.3	Modes of Operation in IoT RPL	58
4.2	WSN Example Simulation.....	65
4.2.1	CAP Time Analysis	65
4.2.2	Beacon Time Analysis.....	67
4.2.3	Static Routing in WSN.....	69
5	IOT-WSN Experiments in NetSim	72
6	Reference Documents.....	72
7	Latest FAQs	72

1 Introduction

Internet of Things (IoT) is a network of devices that connect to and communicate via the internet or other communication networks. These devices collect, exchange, and act on data, often without human intervention. It enables enhanced interactivity and automation across various domains, from smart homes to industrial applications. A typical IOT deployment consists of:

- Embedded devices / sensors.
- Communication over an IP network (between the devices and to/from cloud servers).
- Cloud services, Big Data, Analytics / Machine learning on the cloud.

In the context of NetSim, the sensors are abstract, which means that they could be any kind of sensor or embedded device. These sensors are assumed to sense some physical property or random field such temperature, pressure etc. After sensing the sensors transmit the sensed data in the form of “IP Packets” of user configurable size and interpacket arrival times. NetSim simulates the transmission of these IP packets over an IoT network and does not focus on either, the actual “application payload (or the sensed data)” being sent, or the data storage and analytics of this payload.

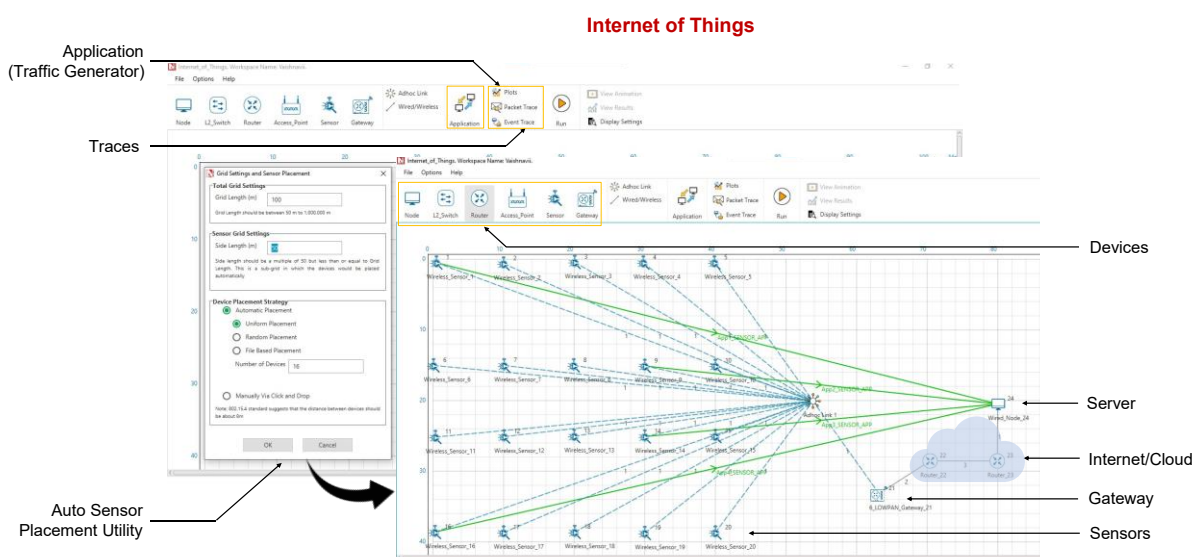


Figure 1-1: A typical IOT scenario in NetSim

NetSim's Internet of Things (IoT) and Wireless Sensor Network (WSN) libraries stack comprises of:

- Application Layer: Sensor App as well as applications such as Voice, Video, CBR etc.
- Transport Layer: UDP
- Network layer: AODV and RPL

- MAC and PHY layers: 802.15.4 Zigbee

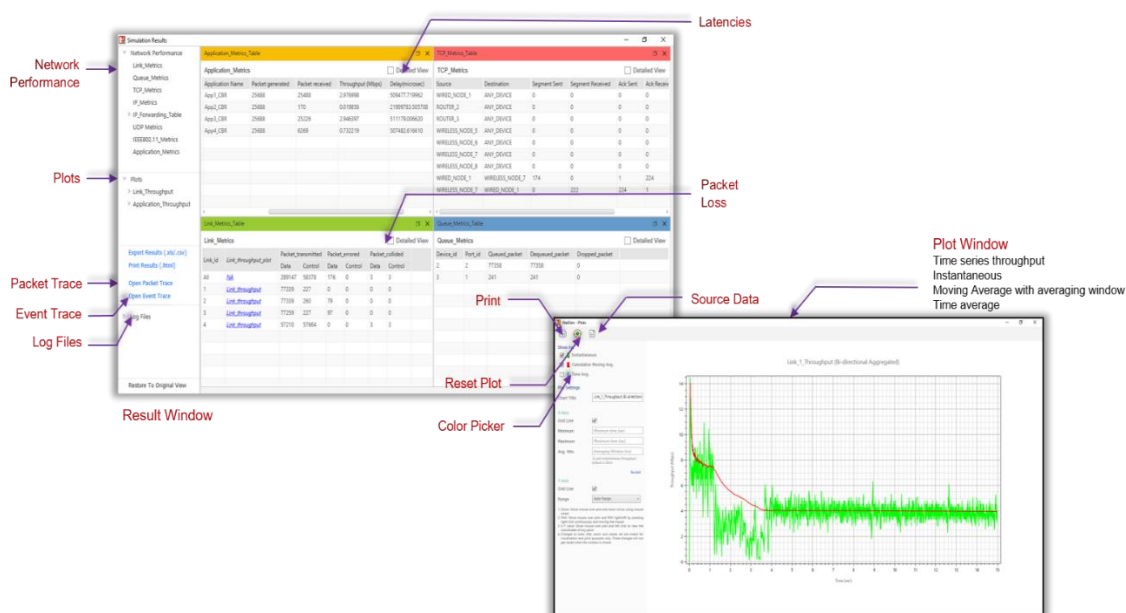


Figure 1-2: The Result dashboard and Plot window shown in NetSim after completion of simulation

NetSim models IoT as a WSN that connects to an Internetwork through a 6LowPAN Gateway. The 6LowPAN Gateway uses two interfaces: a Zigbee (802.15.4) interface and a WAN Interface. The WSN sends data to a 6LowPAN Gateway. The Zigbee interface allows wireless connectivity to the WSN while the WAN interface connects to the external Internetwork.

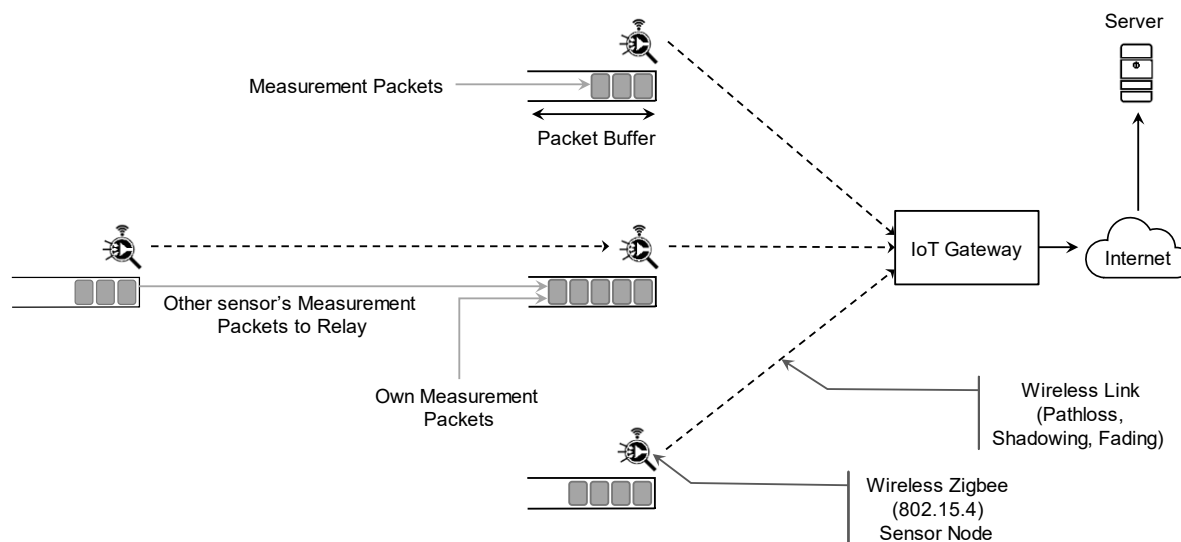


Figure 1-3: A typical application scenario that can be modeled in NetSim. Sensors can generate (measurement) packets that get queued in its packet buffer. These are then transmitted - directly or via hops - over a wireless link to a gateway that then forwards the packet via the internet to a server. Wireless links support various propagation models and ad hoc routing is supported for multi-hop communication. The MAC/PHY layer protocol supported is 802.15.4.

IEEE 802.15.4 uses either Beacon Enabled or Disabled Mode for packet transmission. In Beacon Enabled Mode, nodes use slotted CSMA/CA algorithm for transmitting packets else they use Unslotted CSMA/CA.

2 Simulation GUI

In the Main menu select **New Simulation**→ **Wireless Sensor Networks** as shown Figure 2-1.

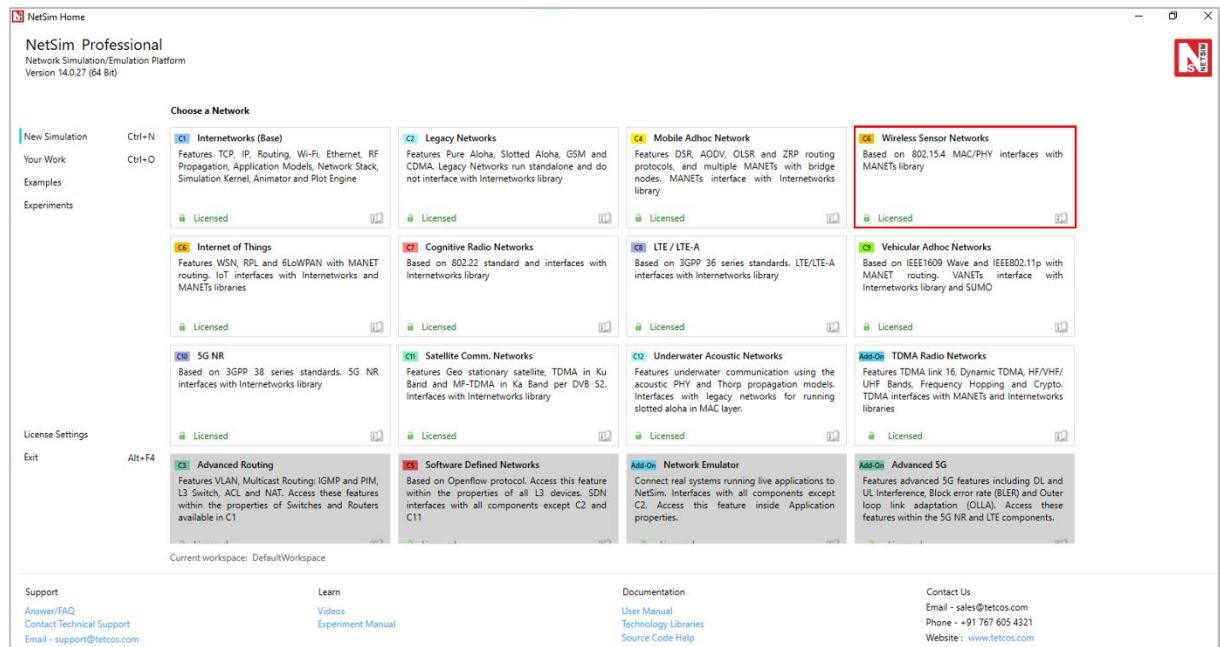


Figure 2-1: NetSim Home Screen

2.1 Create Scenario

2.1.1 Fast Configuration

Device Placement and Grid Settings

Grid Settings
The default grid is a square grid, and you have the option to set the origin and the grid dimensions. Grid Length should be between 50 m to 1,000,000 m.

Origin
X Min (m) Y Min (m)

Dimensions
Width (m) Length (m)

Device Placement Strategy
☐ Manually via Click and Drop
☒ Automatic Placement
☒ Uniform Placement
☐ Random Placement
☐ File Based Placement [View File Format](#)
 Number of Devices

Device Placement Area
The default grid is a square grid, and you have the option to set the grid dimensions.

Dimensions
Width (m) Length (m)

OK

Figure 2-2: Fast Configuration window

Fast Config window allows users to define device placement strategies and conveniently model large network scenarios especially in network such as MANET, WSN and IoT. The parameters associated with the Fast Config Window are explained below:

Grid length: It is the area of simulation environment. Users can change the length of the grid in the range of 50-1000000m.

Side length: It specifies the area in the grid environment within which the devices will be placed. Users can change the side length of the grid in the range of 50 - 1000000m. Side length should be multiple of 50. Side length should always be lesser than or equal to the Grid length.

Device Placement - Automatic Placement:

1. **Uniform Placement:** Devices will be placed uniformly with equal gap between the devices in area as specified inside length. This requires users to specify the number of devices as square number. For E.g., 1, 4, 9, 16 etc.

2. **Random Placement:** Devices will be placed randomly in the grid environment within the area as specified inside length.
3. **File Based Placement:** To place devices in user defined locations file-based placement option can be used. The file has the following general format:

<DEVICE_NAME>,<DEVICE_TYPE>,<X_COORDINATE>,<Y_COORDINATE>

Where, DEVICE_NAME is any name that will be assigned to the device. And DEVICE_TYPE is the unique Device Identifier specific to each type of device in NetSim.

The following table provides a list of all possible devices in MANET, WSN, and IOT Networks that support the Fast Configuration along with their respective device types:

NETWORK	DEVICE_TYPE
Manets	1. WIRELESSNODE 2. BRIDGE_WIREDNODE 3. BRIDGE_WIRELESSNODE 4. WIREDNODE 5. ROUTER 6. L2_SWITCH
WSN	1. Sensors 2. SinkNode
IOT	1. IOT_Sensors 2. LOWPAN_Gateway 3. WIREDNODE 4. WIRELESSNODE 5. ROUTER 6. ACCESSPOINT 7. L2_SWITCH

Table 2-1: Fast Configuration window support different networks

NOTE: For more details about File Based Placement, refer Section 3.6

1. **Number of Devices:** It is the total number of devices that is to be placed in the grid environment. It should be a square number in case of Uniform placement.
2. **Manually Via Click and Drop:** Selecting this option will load a grid environment with an adhoc link where users can add devices manually by clicking and dropping the devices as required.

2.1.2 Wireless Sensor Networks

The devices that are involved in WSN are:

Wireless_Sensor: In general, sensors monitor and records the physical conditions of the environment which is then sent to a central location (Sink node) where the data is collated and analyzed for further action. Sensors in NetSim are abstract in terms of what they sense, and NetSim focuses on the network communication aspects after sensing is performed.

WSN_Sink (in WSN): Sink node is the principal controller in WSN. All sensors send their data to this sink node. In NetSim, users can drop only one sink node in a WSN.

Ad-hoc Link: Ad hoc link depicts a decentralized type of wireless network. The network is ad hoc because it does not rely on any pre-existing infrastructure, such as routers in wired networks or access points in managed wireless networks. In NetSim, Ad hoc link are used to connect the Sensors and the Sink node. Ad hoc links are used here for a visual representation of connection of all the devices in an Ad hoc basis.

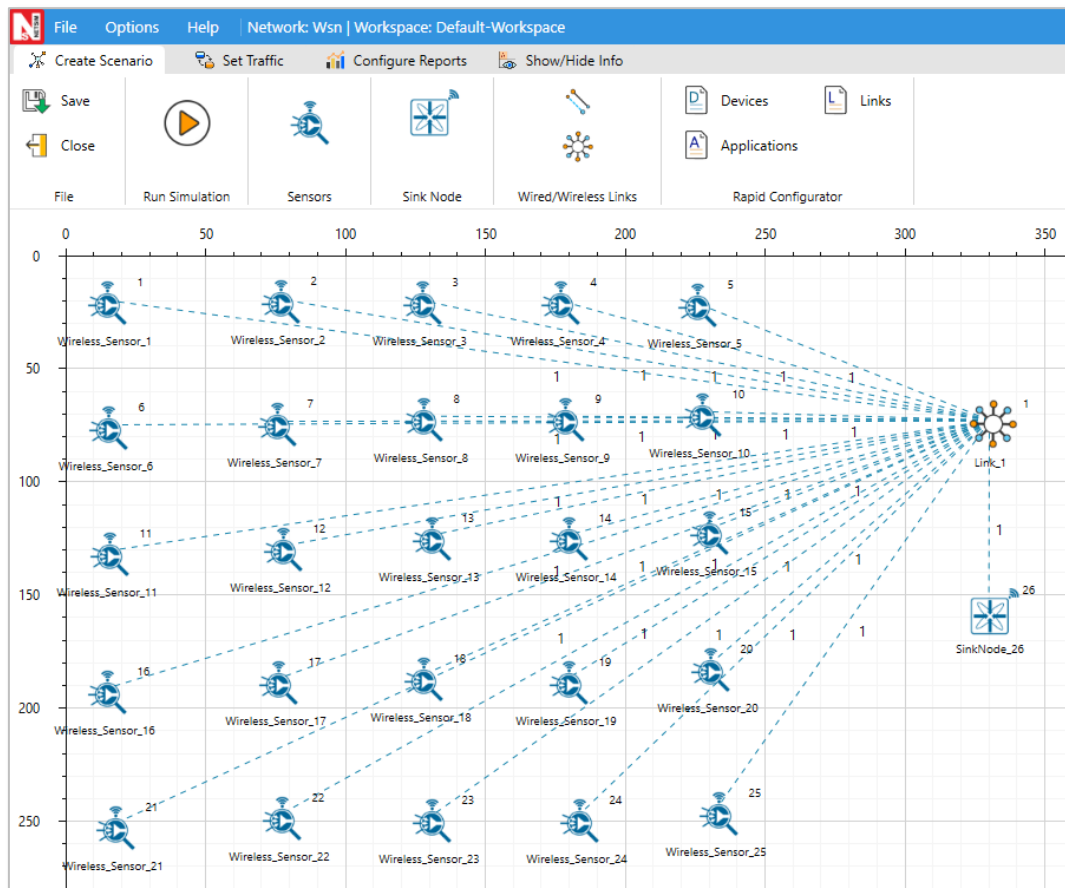


Figure 2-3: Network layout of a WSN. The sensors communication with each other and to the sink via an “Adhoc link”

NOTE: While designing a network, by default an ad hoc link will be present in the scenario. Click sensor nodes and sink nodes present in the ribbon/toolbar and drop them inside the grid. If the auto-connect option in the status bar is turned ON, these devices will be automatically connected to the ad hoc link. Refer section 3.2.3 of User Manual to know more about Auto Connection.

2.1.3 Internet of Things

The devices that are involved in IoT are:

IoT_Sensor - In general, sensors monitor and records the physical conditions of the environment which is then sent to a central location (Lowpan Gateway) where the data is collated and analysed for further action. Sensors in NetSim are abstract in terms of what they sense, and NetSim focuses on the network communication aspects after sensing is performed.

LoWPAN Gateway (in IoT) - LoWPAN is an acronym of Low power Wireless Personal Area Networks. The LoWPAN IoT gateway functions as a border router in a LoWPAN network, connecting a wireless IPv6 network to the Internet. The wired portion of the network in IoT runs IPv4 whereas the wireless portion runs IPv6. The IPv6 routing protocols supported as AODV and RPL.

Ad-hoc Link: Ad hoc link depicts a decentralized type of wireless network. The network is ad hoc because it does not rely on any pre-existing infrastructure, such as routers in wired networks or access points in managed wireless networks. In NetSim IoT, Ad hoc link are used to connect the IoT_Sensors and the 6LoWPAN_Gateway. Ad hoc links are used here for a visual representation of connection of all the devices in an Ad hoc basis.

Users can also add routers and nodes as shown below. Routers can be connected to the 6LoWPAN-Gateway and nodes/switches can be connected to routers using wired/wireless links.

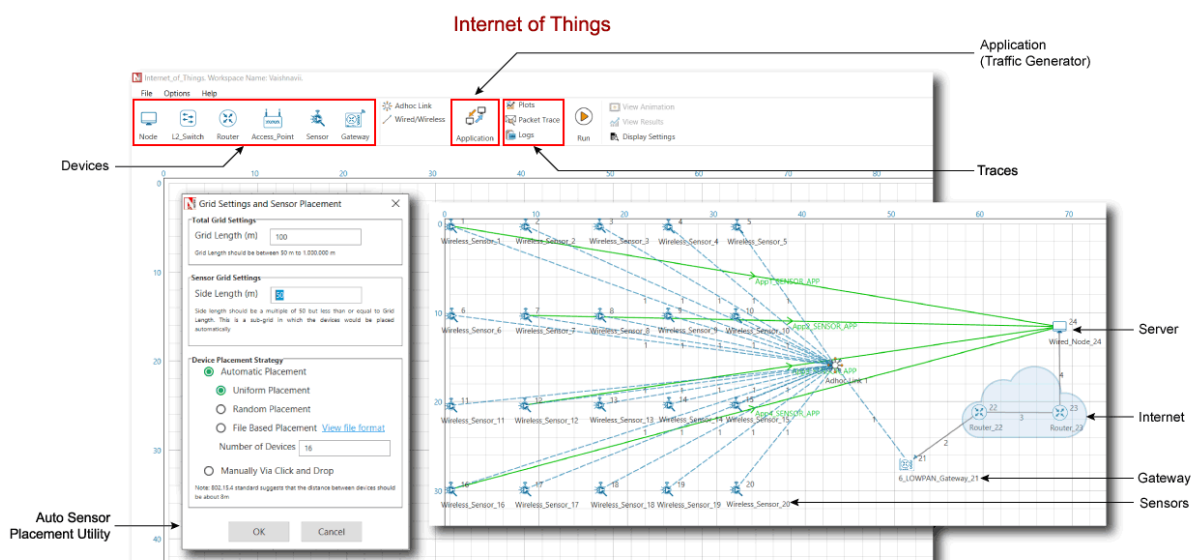


Figure 2-4: Internet of Things

2.1.4 Differences between IoT and WSN in NetSim

WSN	IoT
WSN consists of a network of sink node and sensors.	IOT has a gateway which can be used to connect to internetworks (having routers, switches, APs etc.).
WSN runs IPv4 and features a sink. (not a gateway).	IOT runs IPv6 in the sensor network (802.15.4 MAC/PHY) and IPv4 on the inter-network portion.
Routing protocols in NetSim WSN include, DSR, AODV, OLSR, and ZRP.	Routing protocols in NetSim IoT include, AODV and RPL.

Table 2-2: Differences between IoT and WSN in NetSim

NOTE: Refer MANET Technology library for working of AODV, DSR, OLSR and ZRP.

2.1.5 Device Attributes

GENERAL PROPERTIES

Right click on any sensor and select properties. The general properties of the sensor are:

- **Device name** is the name of sensor which is editable and will reflect in the GUI before and after simulation.
- **X and Y** are the coordinates of a sensor.
- **The Z co-ordinate** by default will be zero (this is reserved for future use).

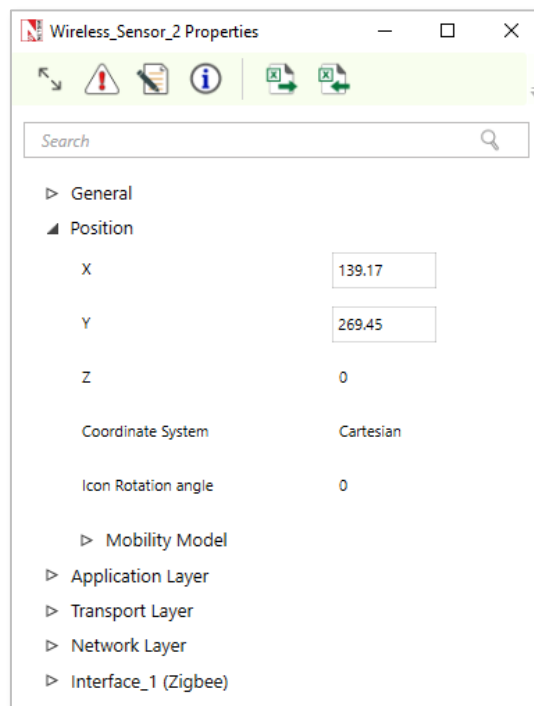


Figure 2-5: General Properties window for Sensor

Interface count is 1 since sensors share the wireless Multipoint-to-Multipoint medium.

Mobility Models: Mobility models can be used to model movement of sensors. The mobility models provided in NetSim are:

- **Random Walk mobility model:** It is a simple mobility model based on random directions and speeds.
- **Random Waypoint Mobility Model:** It includes pause time between changes in direction and/or speed.
- **Group mobility:** It is a model which describes the behavior of sensors as they move together i.e., the sensors having common group id will move together.
- **Pedestrian Mobility Model:** This model is applicable to each node (local parameter), and the configuration parameters are:

- Pedestrian_Max_Speed (m/s) (Range: 0.0 to 10.0. Default: 3.0)
- Pedestrian_Min_Speed (m/s) (Range: 0.0 to 10.0. Default: 1.0)
- Pedestrian_stop_probability (Range: 0 to 1)
- Pedestrian_stop_duration (s). (Range: 1 to 10000)

In this model it is assumed that the pedestrian stops at traffic lights. The stop_probability represents the probability of encountering a traffic light. This is checked for every Calculation_interval. Once stopped, the pedestrian waits for a duration equal to stop_duration for the light to turn green. A new direction is chosen randomly after every stop with θ (angle between new direction and current direction) taking values of 0, 90, 180, 270. These θ values represent the pedestrian continuing in the same direction, taking a left, taking a U turn and taking a right respectively.

A new speed is chosen randomly after every stop. $\text{Min_speed} \leq \text{Speed} \leq \text{Max_Speed}$.

The maximum number of stops and starts is 10.

- **File Based mobility:** In File Based Mobility, users can write their own custom mobility models and define the movement of the mobile users. The name of the trace file generated should be kept as mobility.csv and it should be in the **NetSim Mobility File format**.

APPLICATION PROPERTIES – Transport Protocol, by default set to UDP. To run with TCP, users have to select TCP protocol from the drop down.

NETWORK LAYER- NetSim WSN, supports the following MANET routing protocols.

DSR (Dynamic source routing): Note that in wireless sensor networks, by default Link Layer Ack is enabled. If Network Layer ack is enabled users must set DSR_ACK in addition to Zigbee_ACK in MAC layer.

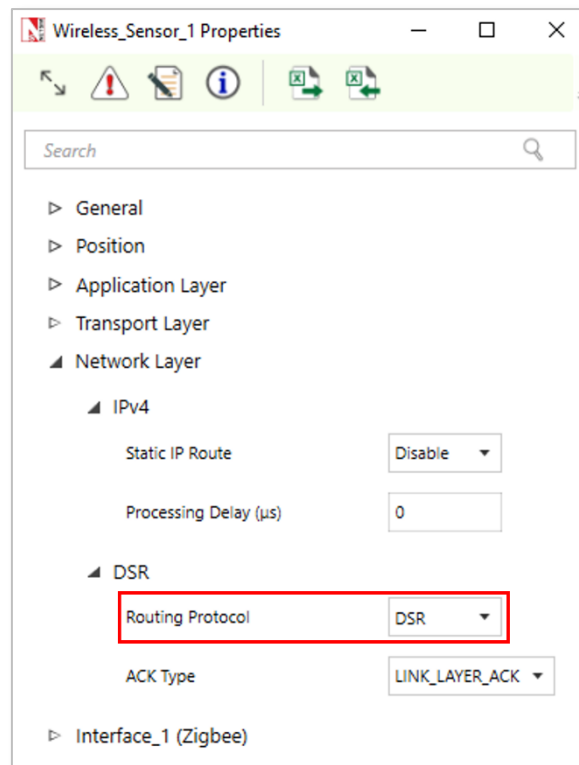


Figure 2-6: Network Layer Properties Window - DSR Protocol

AODV (Ad-hoc on-demand distance vector routing):

AODV (Ad Hoc on Demand Distance Vector) is an on-demand routing protocol for wireless networks that uses traditional routing tables to store routing information. AODV uses timers at each node and expires the routing table entry after the route is not used for a certain time.

Some of the features implemented in NetSim are,

- RREQ, RREP and RERR messages.
- Hello message.
- Interface with other MAC/PHY protocols such as 802.15.4, TDMA / DTDMA.

ZRP (Zone routing protocol): For interior routing mechanism NetSim uses OLSR protocol.

Hello interval describes the interval in which it will discover its neighbor routes.

Refresh interval is the duration after which each active node periodically refresh routes to itself.

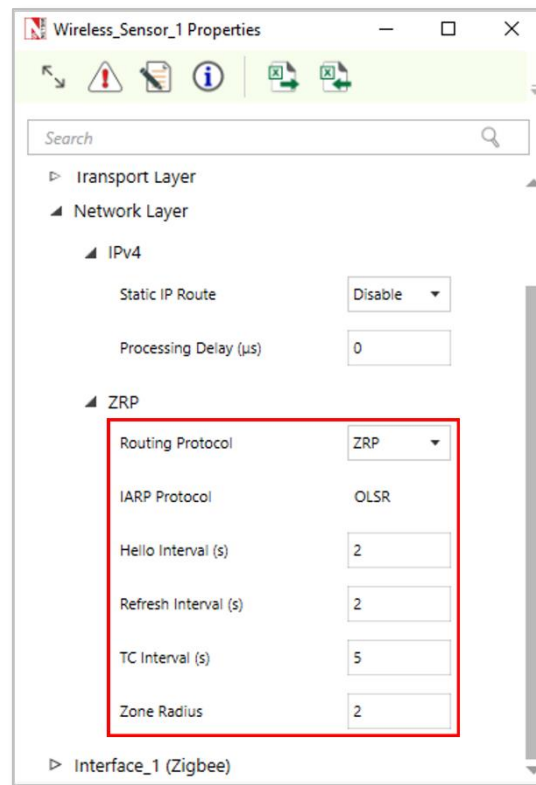


Figure 2-7: Network Layer Properties Window - ZRP Protocol

TC Interval is a Topology control messages are the link state signaling done by OLSR. These messages are sent at TC interval every time.

Zone radius: After dividing the network range of the divided network will be based on zone radius. A node's routing zone is defined as a collection of nodes whose minimum distance in hops from the node in question is no greater than a parameter referred to as the zone radius.

OLSR (Optimized link State Routing): Except zone radius all the parameters are similar to ZRP.

DATALINK LAYER

802.15.4 (Zigbee Protocol) runs in MAC layer. In the sink node or pan coordinator properties users can configure the Beacon frames and the Superframe structure.

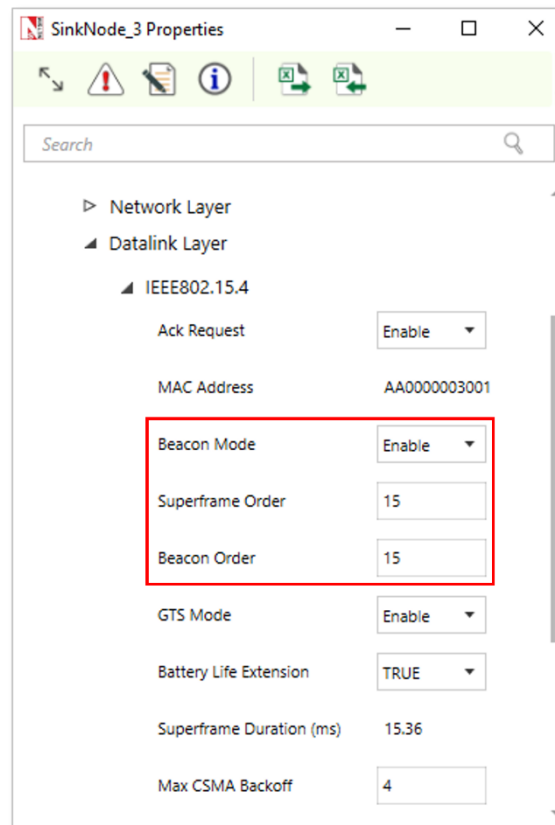


Figure 2-8: Datalink layer properties window for sinknode

Superframe Order – It describes the length of the active portion of the Superframe, which includes the beacon frame. Range is from 0-15.

Beacon Order- Describes the interval at which coordinate shall transmit its beacon frames. Range is from 1-15.

GTS Mode (Guaranteed Time Slot) – If it is enabled it allows a device to operate on the channel within a portion of the super frame that is dedicated (on the PAN) exclusively to the device.

Battery life Extension subfield is 1 bit in length and shall be set to one if frames transmitted to the beaconing device.

Superframe Duration is divided into 16 equally sized time slots, during which data transmission is allowed. The value of super-frame duration by default is 15.36ms.

Max CSMA Backoff is the CSMA-CA algorithms will attempts before declaring a channel access failure. Having range 0-5.

Minimum CAP length is the minimum number of symbols forming the Contention access period. This ensures that MAC commands can still be transferred to devices when GTSS (Guaranteed time slots) are being used.

Max and Min backoff exponent values of CSMA-CA algorithms having range 3-5.

Max frame retries is the total number of retries after failed attempts.

Unit Backoff period is the number of symbols forming the basic time period used by the CSMA-CA algorithms.

PHYSICAL LAYER

The frequency band used in NetSim WSN simulations is 2.4 GHz, and the bandwidth is 5 MHz. NetSim simulates a single channel ZigBee network and does not support multiple channels.

Data rate is the number of bits that are processed per unit of time. The data rate is fixed at 250 kbps per the 802.15.4 standard.

Chip Rate: A chip is a pulse of direct sequence spread spectrum code, so the chip rate is the rate at which the information signal bits are transmitted as pseudo random sequence of chips.

Modulation technique: O-QPSK (Offset quadrature phase shift keying) sometimes called as staggered quadrature phase shift keying is a variant of phase-shift keying modulation using 4 different values of the phase to transmit.

MinLIFSPeriod is minimum long inter frame spacing Period. It's a time difference between short frame and long frame in unacknowledged case and time difference between short frame and acknowledged in acknowledge transmission.

SIFS (Short inter frame Symbol) is generally the time for which receiver wait before sending the CTS (Clear To Send) & acknowledgement package to sender, and sender waits after receiving CTS and before sending data to receiver. Its main purpose is to avoid any type of collision. Min SIFS period is the minimum number of symbols forming a SIFS period.

Phy SHR duration is the duration of the synchronization header (SHR) in symbol for the current PHY.

Phy Symbol per Octet is number of symbols per octet for the current PHY.

Turn Around Time is transmitter to receiver or receiver to transmitter turnaround time is defined as the shortest time possible at the air interface from the trailing edge of the last chip (of the first symbol) of a transmitted PLCP protocol data unit to the leading edge of the first chip (of the first symbol) of the next received PPDU.

CCA (Clear Channel assessment) is carrier sensing mechanisms in Wireless Networks. The different types of CCA modes available are:

- **Carrier Sense Only:** It shall report a busy medium only upon the detection of a signal complaint with this standard with the same modulation and spreading characteristics of

the PHY that is currently in use by the device. This signal may be above or below the ED threshold.

- **Energy Detection:** It shall report a busy medium upon detecting signal strength above the ED threshold.
- **Carrier Sense with Energy Detection:** It shall report a busy medium using a logical combination of detection of a signal with the modulation and spreading characteristics of this standards and Energy above the ED threshold, where the logical operator may be AND or OR.

Receiver sensitivity is the minimum magnitude of input signal required to produce a specified output signal having a specified signal-to-noise ratio, or other specified criteria. It's up to our calculation what we want a receiver sensitivity.

Receiver ED threshold is intended for use by a network layer as part of a channel selection algorithms. It is an estimate of the received signal power within the bandwidth of the channel. No attempt is made to identify or decode signal on the channel. If the received signal power is greater than the ED threshold value, then the channel selection algorithms will return false.

Transmitter Power is the signal intensity of the transmitter. The higher the power radiated by the transmitter's antenna the greater the reliability of the communication system. And connection medium is Wireless.

Reference Distance (d_0) is known as the reference distance and the value of d_0 is usually defined in the pathloss model or in the standard. PL_{d_0} is the pathloss at reference distance. In 805.15.4, the standard defines $d_0 = 8m$ and $PL_{d_0} = 58.5 dB$. Please see Propagation-Model.pdf manual for more information.

POWER MODEL

- **Power source** can be battery or main line. This model in NetSim is used for energy calculations. In case of battery following parameters will be considered: -
- **Recharging current** is the current flow during recharging. Range is from 0-1000mA.
- **Energy Harvesting** is the process by which energy is derived from external source, captured, and stored. NetSim supports an abstract Energy Harvesting model a specified amount of energy (calculated from recharging current and voltage specified) is added to the remaining energy of the node periodically to replenish the battery. It can be turned on or off.
- **Initial Energy** is the battery energy-range is from 0.001 – 3250mAH
- **Transmitting current** for transmitting the power. Range 0-1000mA. Transmit power and transmit current are independent in NetSim. Since the focus of NetSim is packet simulation, the power modeling is abstract. It is left to the user to change the transmit

current accordingly (when increasing/decreasing the transmit power) if the user's goal is to study power consumption.

- **Idle mode** is the current flow during the ideal mode-range is between 0-1000mA.
- **Voltage** is a measure of the energy carried by the charge. Range is from 0-10V.
- **Received current** is the current required to receive the data ranging from 0-1000mA.
- **Sleep mode current** is current flowing in sleep mode of battery-range is from 0-1000mA.

NOTE: The resultant energy metrics and their definitions are provided in the NetSim User Manual in the Outputs section.

The following table shows the properties of sensor in NetSim.

Global properties (and default settings)	
Network layer	
Routing protocol	DSR
ACK_Type	LINK_LAYER_ACK
Data link layer	
ACK request	Enable
Max Csma BO	4
Max Backoff Exponent	5
Min Backoff Exponent	3
Max frame retries	3
Local properties (and Default settings)	
Physical layer	
phySHRduration(symbols)	3
Physymbolperoctet	5.3
CCA mode	CARRIER_SENSE_ONLY
Reciever sensitivity(dbm)	-85
ED_Threshold (dbm)	-95
Transmitter power(mW)	1
Power	
Power source	Battery
Energy harvesting	ON
Recharging current (mA)	0.4
Initial energy (mAH)	0.5
Transmitting current (mA)	8.8
Idle mode current (mA)	3.3
Voltage (v)	3.6
Receiving current (mA)	9.6
Sleep mode current (mA)	0.237

Table 2-3: MAC and PHY layer properties of sensor

2.2 Set Node, Link and Application Properties

- Users need to connect the sensors and **LoWPAN gateway** using adhoc links.
- Interconnection among other devices is same as in Internetworks.
- **LoWPAN gateway** can be connected with router using links.

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements.
- Routing Protocol in Application Layer of routers and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global/Local.
- In Sensor Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer, Physical Layer and Power are Global/Local.

NOTE:

- Global - Changing properties in one node will automatically reflect in any other nodes in that network.
- Local - Changing properties in one node will not reflect in any other nodes in that network.
- The following are the main properties of sensor node in PHY and Datalink layers as shown Figure 2-9/Figure 2-10/Figure 2-11.

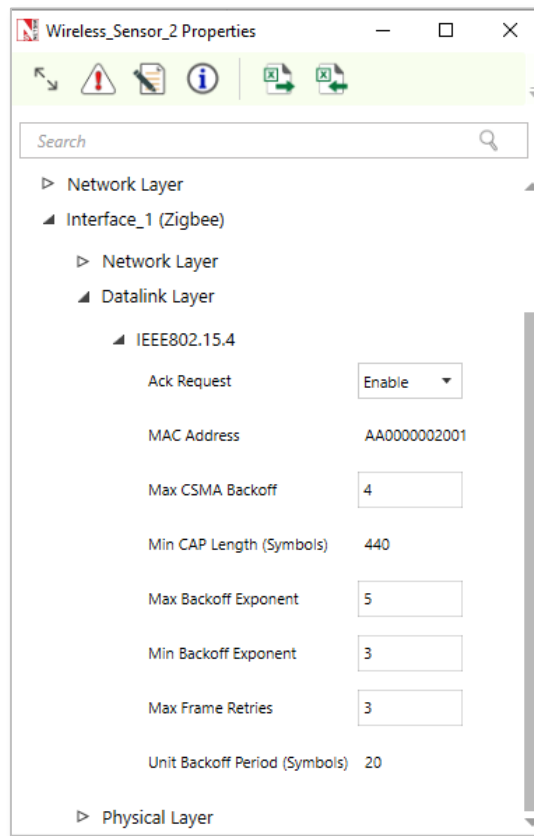
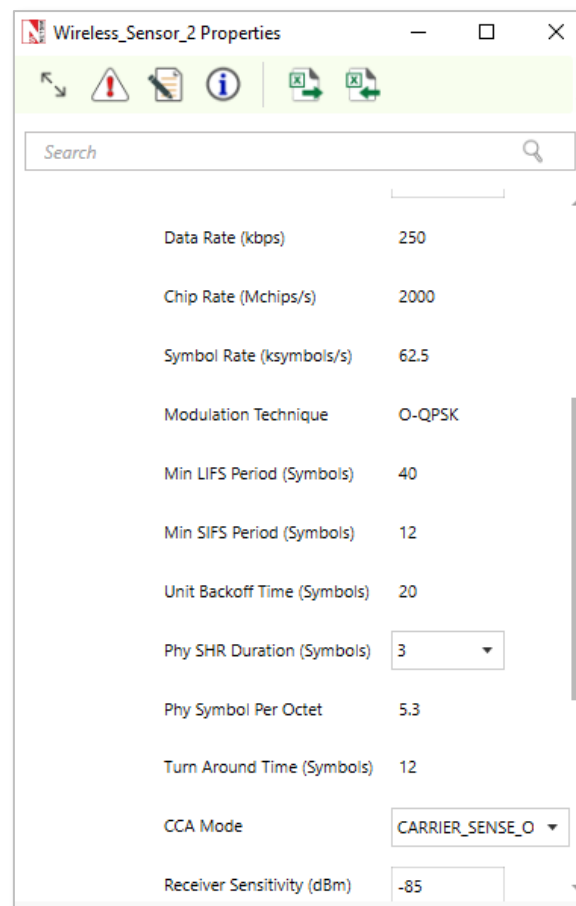


Figure 2-9: Datalink layer properties window for sensor



The image shows a software window titled "Wireless_Sensor_2 Properties". It features a toolbar with icons for navigation, warnings, help, and file operations. Below the toolbar is a search bar. The main area contains a list of PHY layer properties for a sensor, each with a label and a value or a dropdown menu.

Property	Value
Data Rate (kbps)	250
Chip Rate (Mchips/s)	2000
Symbol Rate (ksymbols/s)	62.5
Modulation Technique	O-QPSK
Min LIFS Period (Symbols)	40
Min SIFS Period (Symbols)	12
Unit Backoff Time (Symbols)	20
Phy SHR Duration (Symbols)	3
Phy Symbol Per Octet	5.3
Turn Around Time (Symbols)	12
CCA Mode	CARRIER_SENSE_O
Receiver Sensitivity (dBm)	-85

Figure 2-10: PHY layer properties window for sensor

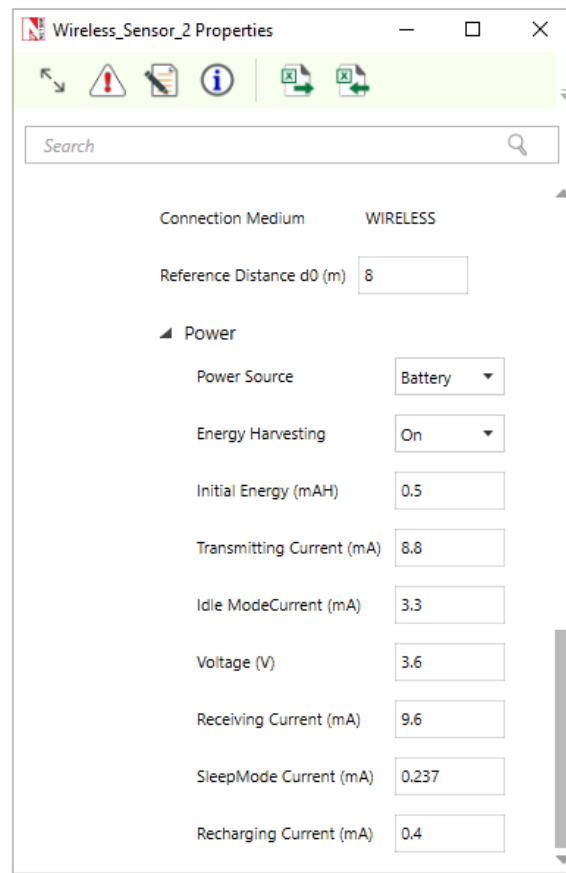


Figure 2-11: Battery Model for Sensor

- Set the values according to requirement and click OK.
- Click on the Set Traffic tab present on the top ribbon and select an application.

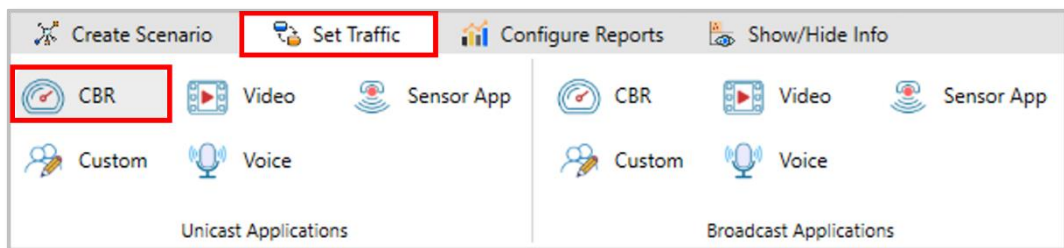


Figure 2-12: Application icon present on top ribbon.

- Set the application properties as per the requirements and proceed.

Property	Value
Application ID	1
Application Name	App1_SENSOR_APP
Source Count	1
Source ID	1
Destination Count	1
Destination ID	3
Start Time (s)	0
End Time (s)	100000
Encryption	NONE
Random Startup	FALSE
Session Protocol	NONE
Transport Protocol	UDP
QoS	BE

Figure 2-13: Application Configuration window

Detailed information on Application properties is available in section 6 of NetSim User Manual.

2.2.1 Setting Static Routes

In Device Properties > Network layer > Static IP Route, users can set static routes. When static routes are set the dynamic routing protocol entries are overwritten by the static routing entries. Static route configured explained in Internetworks technology library document, Section Configuring Static Routing in NetSim

Static route option is available for all sensors in WSN.

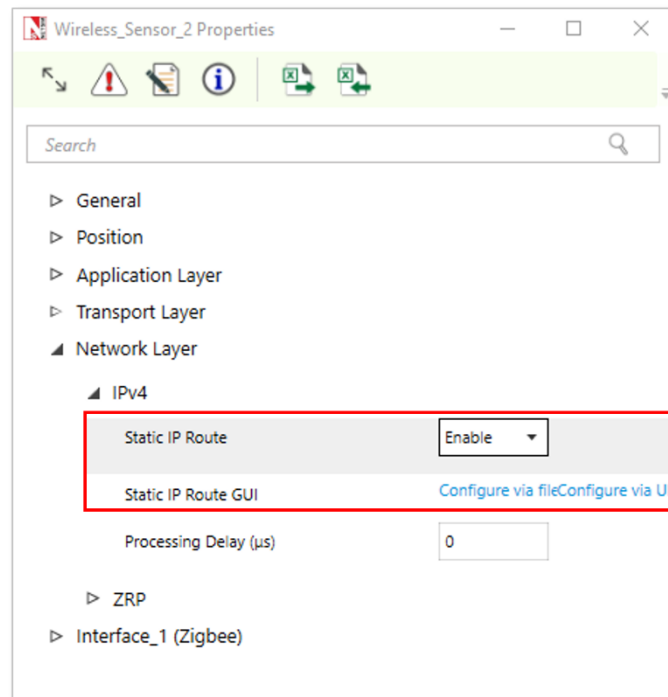


Figure 2-14: Static Route configuration window

The Static routes option is not available in the wireless portion of the IoT network as IoT devices work with IPv6 network addressing. The devices present in the wired portion of network say have IPv4 addressing. Hence static routes can be configured in the wired section (till the gateway) of an IoT network.

2.3 Enable Packet Trace, Event Trace & Plots (Optional)

Check Enable Packet Trace / Event Trace check box from Configure Reports tab and click OK. To get detailed help, please refer sections 8.4 and 8.5 in User Manual. Select Plots icon for enabling Plots and click OK.

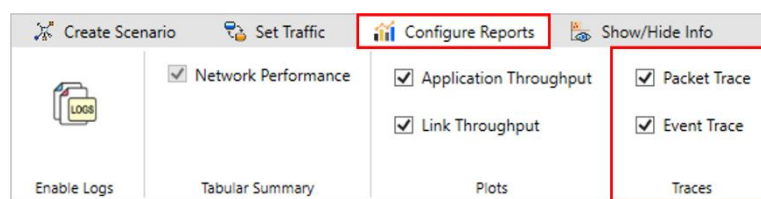


Figure 2-15: Enable Packet Trace, Event Trace & Plots options on top ribbon

2.4 Run Simulation

Click on **Run Simulation** icon on the top toolbar.

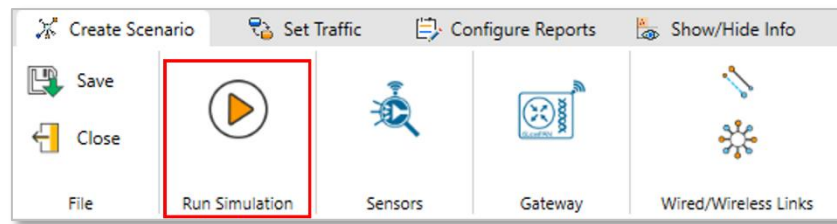


Figure 2-16: Run Simulation options on top ribbon

Set the Simulation Time and click on OK.

3 Model Features

3.1 L3 Routing: DSR, OLSR, ZRP and AODV

Refer to the MANETs technology library (PDF) document for detailed information. Note that:

- WSN supports DSR, OLSR, ZRP and AODV protocols
- IOT supports AODV and RPL protocol, since only AODV and RPL have IPv6 support in NetSim.

3.2 L3 Routing: RPL Protocol

Routing Protocol for Low power and Lossy Networks (RPL) Overview

Low-power and Lossy Networks consist largely of constrained nodes (with limited processing power, memory, and sometimes energy when they are battery operated). These routers are interconnected by lossy links, typically supporting only low data rates that are usually unstable with relatively low packet delivery rates. Another characteristic of such networks is that the traffic patterns are not simply point-to-point, but in many cases point-to-multipoint or multipoint-to-point.

RPL Routing Protocol works in the Network Layer and uses IPv6 addressing. It runs a distance vector routing protocol based on Destination Oriented Directed Acyclic Graph (DODAGs).

Terminology of RPL routing protocol:

- **DAG (Directed Acyclic Graph):** A directed graph having the property that all edges are oriented in such a way that no cycles exist. All edges are contained in paths oriented toward and terminating at one or more root nodes.
- **DAG root:** A DAG root is a node within the DAG that has no outgoing edge. Because the graph is acyclic, by definition, all DAGs must have at least one DAG root and all paths terminate at a DAG root. In NetSim, only single root is possible. i.e. the 6LowPAN Gateway

- **Destination-Oriented DAG (DODAG):** A DAG rooted at a single destination, i.e., at a single DAG root (the DODAG root) with no outgoing edges.
- **Up:** Up refers to the direction from leaf nodes towards DODAG roots, following DODAG edges. This follows the common terminology used in graphs and depth-first search, where vertices further from the root are "deeper" or "down" and vertices closer to the root are "shallower" or "up"
- **Down:** Down refers to the direction from DODAG roots towards leaf nodes, in the reverse direction of DODAG edges. This follows the common terminology used in graphs and depth-first search, where vertices further from the root are "deeper" or "down" and vertices closer to the root are "shallower" or "up"
- **Rank:** A node's Rank defines the node's individual position relative to other nodes with respect to a DODAG root. Rank strictly increases in the Down direction and strictly decreases in the Up direction.
- **RPLInstanceID:** An RPL Instance ID is a unique identifier within a network. DODAGs with the same RPLInstanceID share the same Objective Function.
- **RPL instance:** When we have one or more DODAG, then each DODAG is an instance. An RPL Node may belong to multiple RPL Instances, and it may act as router in some and as a leaf in others. Any sensor can be configured as a Router or Leaf. Leaf nodes does not take part in RPL routing.
- **DODAG ID:** Each DODAG has an IPV6 ID. This ID is given to its root only. And as the root doesn't change ID also don't change
- **Objective Function (OF):** An OF defines how routing metrics, optimization objectives, and related functions are used to compute Rank. Furthermore, the OF dictates how parents in the DODAG are selected and, thus, the DODAG formation.

3.2.1 RPL Objective Function

The objective function in NetSim RPL seeks to find the route with the best link quality. The objective function:

static UINT16 compute_candidate_rank(NETSIM_ID d, PRPL_NEIGHBOR neighbour) can be found in Neighbor.c under the RPL project.

Link quality calculations, available in Zigbee Project 802.15.4 c file in function get_link_quality().

$$L_q = \left(1 - \left(\frac{p}{r_s}\right)\right)$$

where p = Received power (dBm) and rs = Receiver sensitivity (dBm)

And Final

$$\text{Link Quality} = \frac{\text{Sending Link Quality} + \text{Receiving Link Quality}}{2}$$

The rank calculations are done in Neighbour.c in RPL project and is.

$$\text{Rank} = (\text{Max}_{\text{increment}} - \text{Min}_{\text{increment}}) \times (1 - L_q)^2 + \text{Min}_{\text{increment}}$$

The link quality in this case is based on received power and can be modified by the user to factor in distance, delay etc, Link quality is calculated by making calls to the functions in the following order:

1. **compute_candidate_rank()** – RPL \Neighbor.c
2. **fn_NetSim_stack_get_link_quality()** - NetSim network stack which in turn calls
3. **zigbee_get_link_quality()** – ZigBee \802_15_4.c

The function **fn_NetSim_stack_get_link_quality()** is part of NetSim's network stack which is closed to user. However the function **zigbee_get_link_quality()** is open to the users and can be modified if required.

3.2.2 Topology Construction

NetSim IOT WSNs do not typically have predefined topologies, for example, those imposed by point-to-point wires, so RPL has to discover links and then select peers sparingly. RPL routes are optimized for traffic to or from one or more roots that act as sinks for the topology. As a result, RPL organizes a topology as a Directed Acyclic Graph (DAG) that is partitioned into one or more Destination Oriented DAGs (DODAGs), one DODAG per sink.

RPL identifiers: RPL uses four values to identify and maintain a topology:

- The first is an RPLInstanceID. An RPLInstanceID identifies a set of one or more Destination Oriented DAGs (DODAGs). A network may have multiple RPLInstanceIDs, each of which defines an independent set of DODAGs, which may be optimized for different Objective Functions (OFs) and/or applications. The set of DODAGs identified by an RPLInstanceID is called a RPL Instance. All DODAGs in the same RPL Instance use the same OF
- The second is a DODAGID. The scope of a DODAGID is a RPL Instance. The combination of RPLInstanceID and DODAGID uniquely identifies a single DODAG in the network. A RPL Instance may have multiple DODAGs, each of which has a unique DODAGID.
- The third is a DODAGVersionNumber. The scope of a DODAGVersionNumber is a DODAG. A DODAG is sometimes reconstructed from the DODAG root, by incrementing the DODAGVersionNumber. The combination of RPLInstanceID, DODAGID, and DODAGVersionNumber uniquely identifies a DODAG Version.

- The fourth is Rank. The scope of Rank is a DODAG Version. Rank establishes a partial order over a DODAG Version, defining individual node positions with respect to the DODAG root.

DIS (DODAG Information Solicitation) transmission:

Root sends DIS message to the Router/Leaf which are in range. The Router in turn broadcasts its further and so on.

DIO (DODAG Information Object) transmission:

RPL nodes transmit DIOs using a Trickle Timer. This message is multicasted downwards in a DODAG. With DIO child parent relationship and sibling relationship is established.

DODAG Construction

- Nodes periodically send link-local multicast DIO messages.
- Stability or detection of routing inconsistencies influence the rate of DIO messages.
- Nodes listen for DIOs and use their information to join a new DODAG, or to maintain an existing DODAG.
- Nodes may use a DIS message to solicit a DIO as shown below.

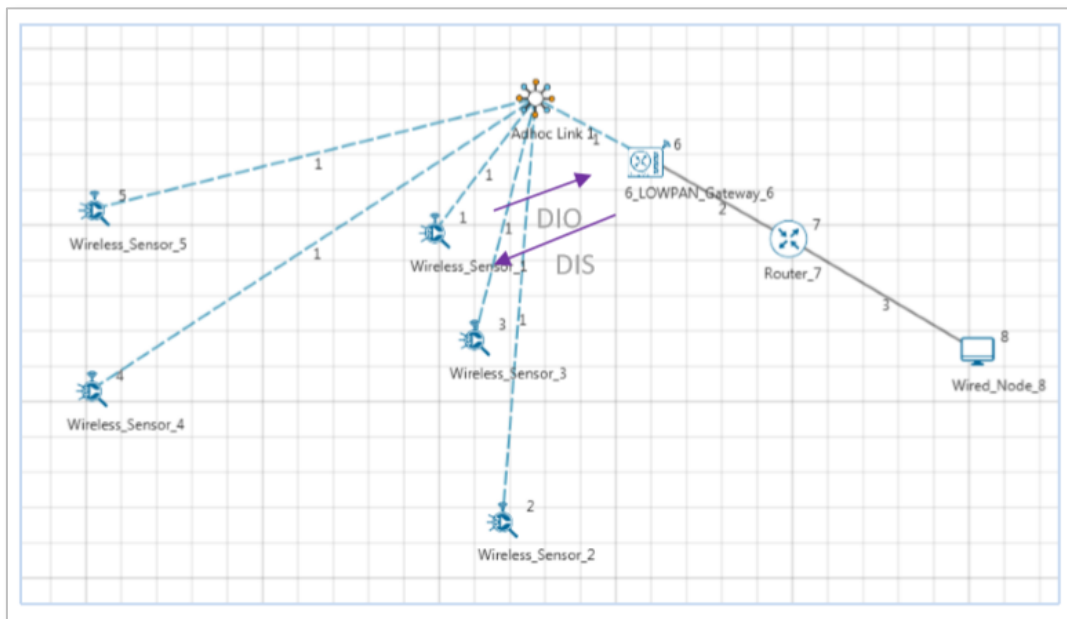


Figure 3-1: Exchange DIO/DIS Control message for Sensor and Lowpan gateway

- Rank is then established. Rank is decided based on the DIS message received from the Root and the link quality.
- Based on information in the DIOs the node chooses parents to the DODAG root
- As a Result, the nodes follow the upward routes towards the DODAG root.
- If the destination is unreachable, then the root will drop the packet.

- Note that DIS messages are sent by sensors which are not part of the DODAG. The sensors which are part of the DODAG, and which received the DIO message will send the DAO message in return, whereas the sensors which did not receive the DIO messages will send the DIS message.

3.2.3 RPL Log File

Once simulation is completed, users can access the `rpl_log` file from the results dashboard as shown below Figure 3-2.

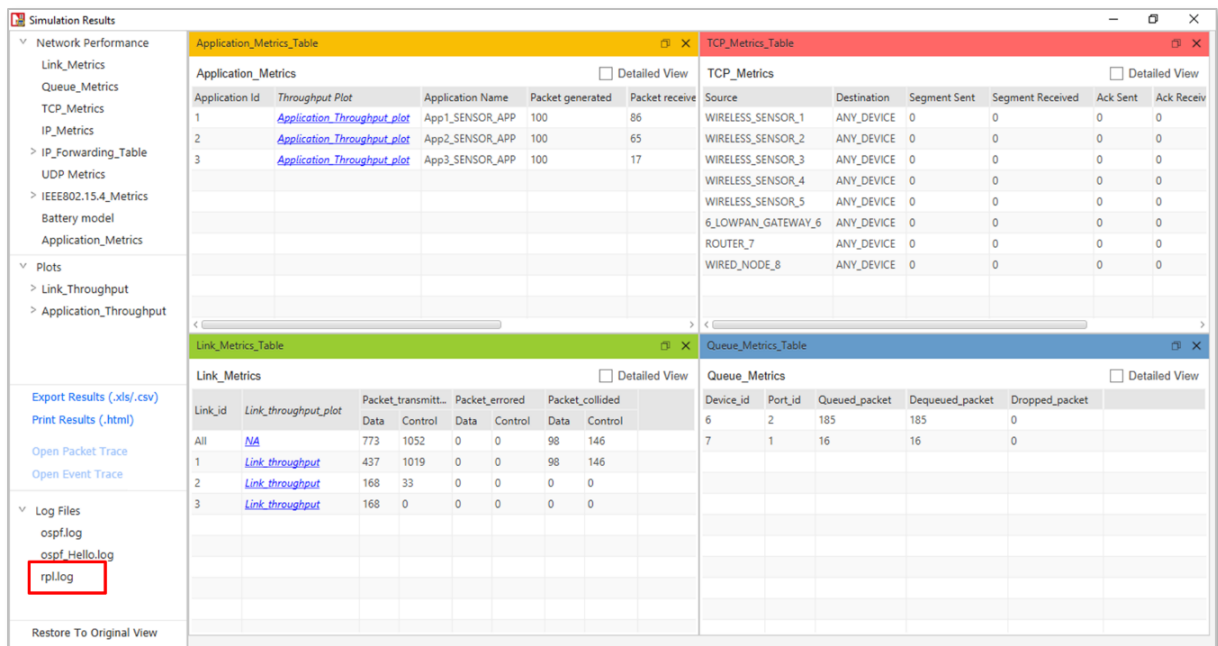


Figure 3-2: Results dashboard window

However, to get detailed information related to Rank Calculations the `DEBUG_RPL` preprocessor directive needs to be uncommented in the code.

Procedure to get the RPL log file:

- Go to NetSim Home page and click on Your work.
- Click on Workspace Options and then click on Open Code and open the codes in Visual Studio. Set x86 or x64 according to the NetSim build which you are using.
- Go to the RPL Project in the Solution Explorer. Open RPL.h file and change `//#define DEBUG_RPL` to `#define DEBUG_RPL` as shown below:

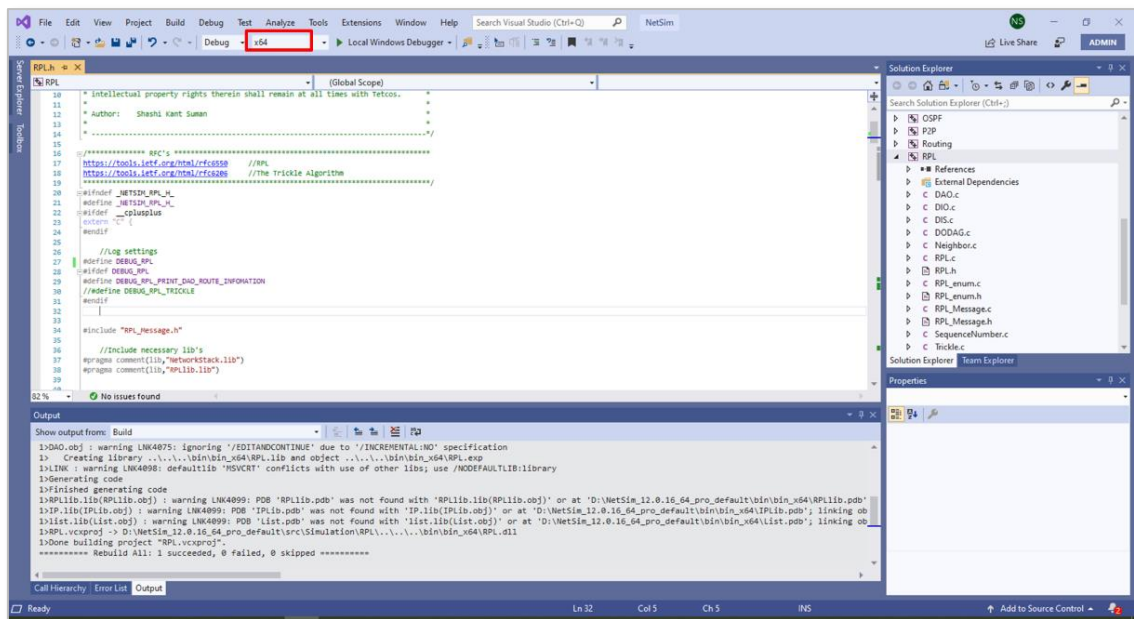


Figure 3-3: Visual Studio

- Right click on the **RPL** project in the solution explorer and click on rebuild.
- After the RPL project is rebuild successful, go back to the network scenario.

Now after any IoT-RPL simulation, an RPL log file is generated with detailed information about the DODAG formation. An example rpl_log file is shown below Figure 3-4.

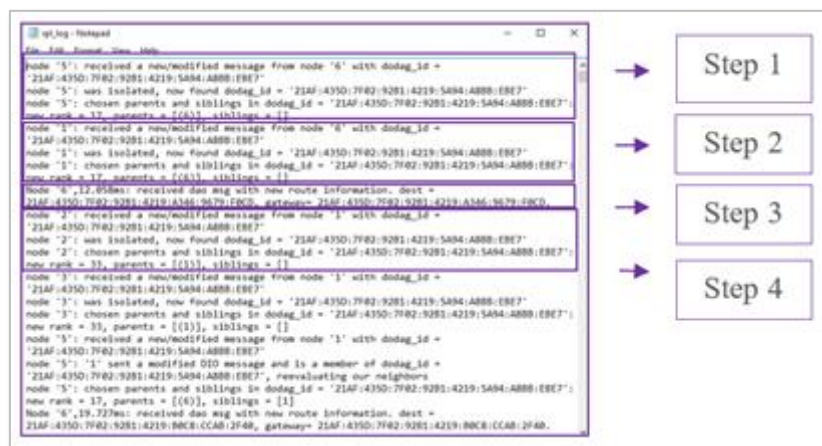


Figure 3-4: RPL Log file

Explanation of the log file:

Step 1:

- Node 5 receives a DIO msg from Node 6 (i.e., root).
- Node 5 finds the DODAG id.
- Based on the DIO message received from Node 6, Node 5 choses its “Parent as Node 6” and establishes its “New Rank = 17”. It does not have any siblings.

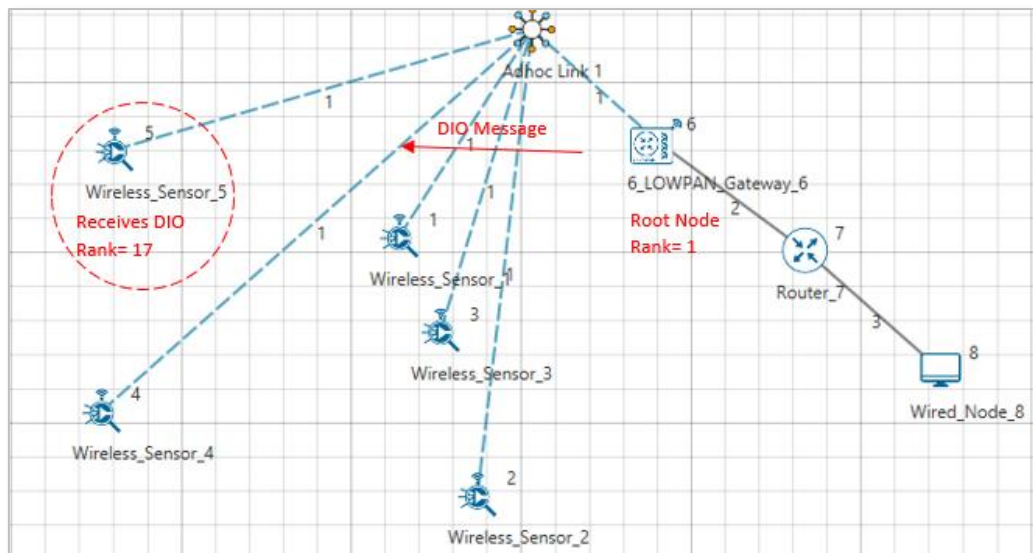


Figure 3-5: Node 5 choses its “Parent as Node 6 - New Rank = 17

Step 2:

- Node 1 receives a DIO msg from Node 6 (i.e. root).
- Node 1 finds the DODAG id.
- Based on the DIO message received from Node 6, Node 1 choses its “Parent as Node 6” and establishes its “New Rank = 17”. It does not have any siblings.

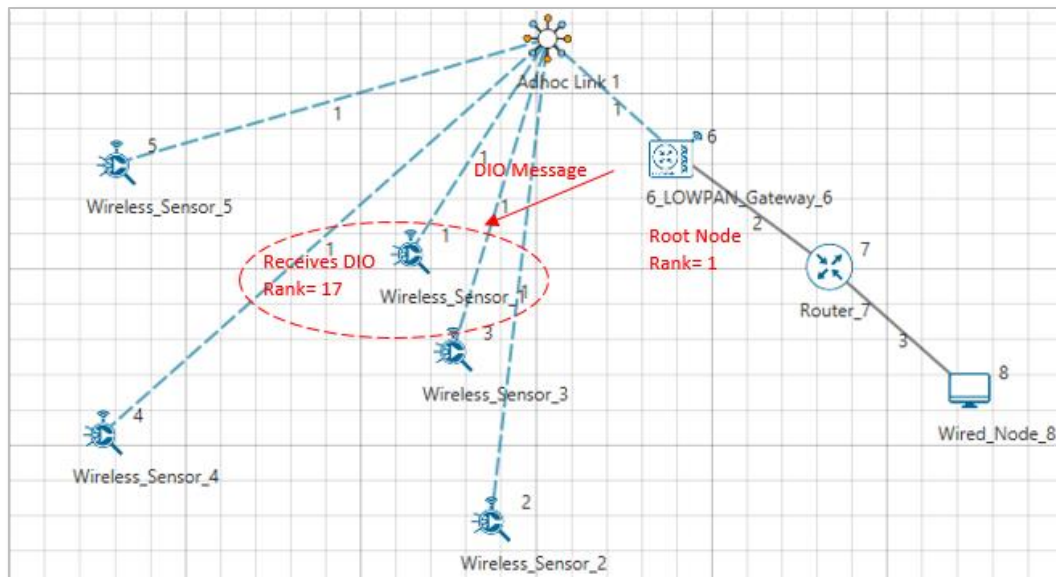


Figure 3-6: Node 1 choses its “Parent as Node 6 - New Rank = 17

Step 3:

- Node 6 receives as DAO message from Node 1 with the new route information about the destination and the Gateway.

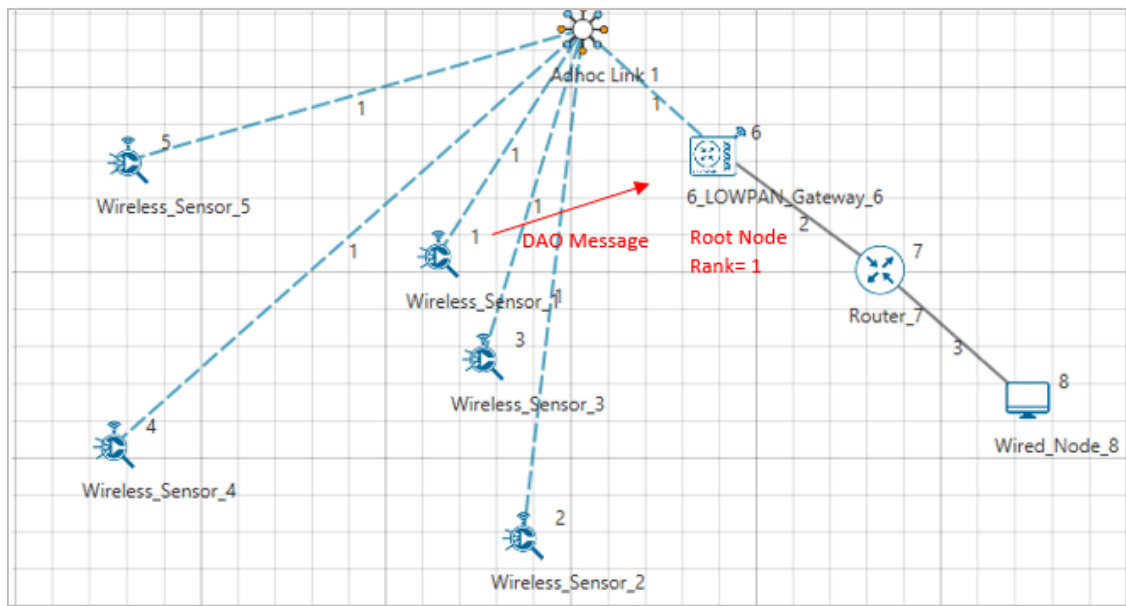


Figure 3-7: Node 6 receives as DAO message from Node 1

Step 4:

- Node 2 receives a DIO msg from Node 1 (i.e. Sensor which is configured as Router).
- Node 2 finds the DODAG id.
- Based on the DIO message received from Node 1, Node 2 choses its “Parent as Node 1” and establishes its “New Rank = 33” since it is in the next Rank level. It does not have any siblings.

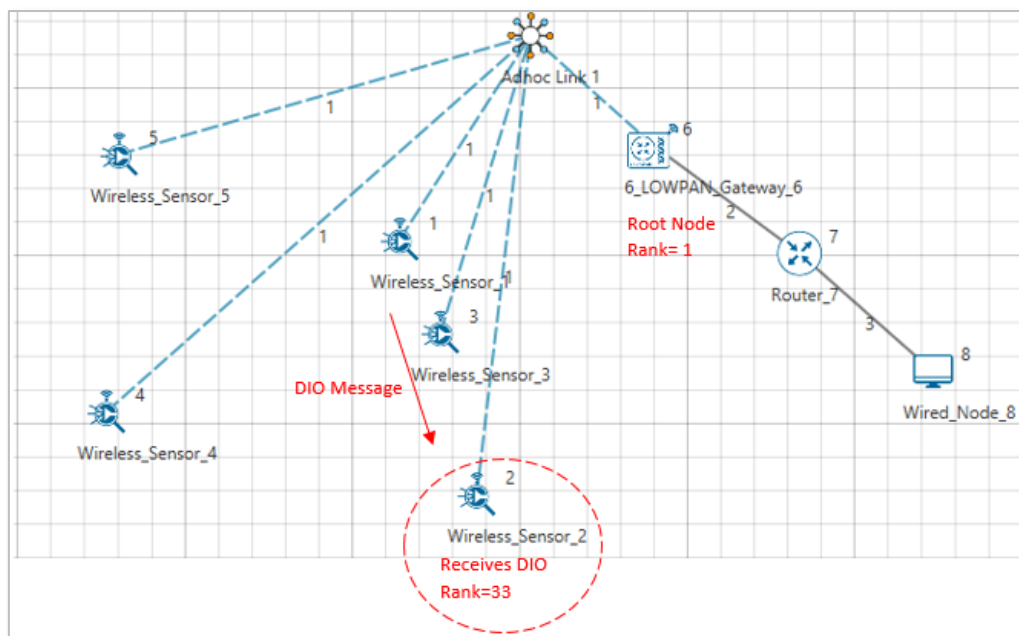


Figure 3-8: Node 2 choses its “Parent as Node 6 - New Rank = 33

Likewise, DODAG formation throughout the simulation is logged inside the rpl_log file

3.2.4 Viewing RPL control messages in Wireshark

Wireshark option can be enabled in the ZigBee devices to capture network traffic during the simulation. RPL control messages such as DAO, DIO etc. can be seen in Wireshark as shown below Figure 3-9.

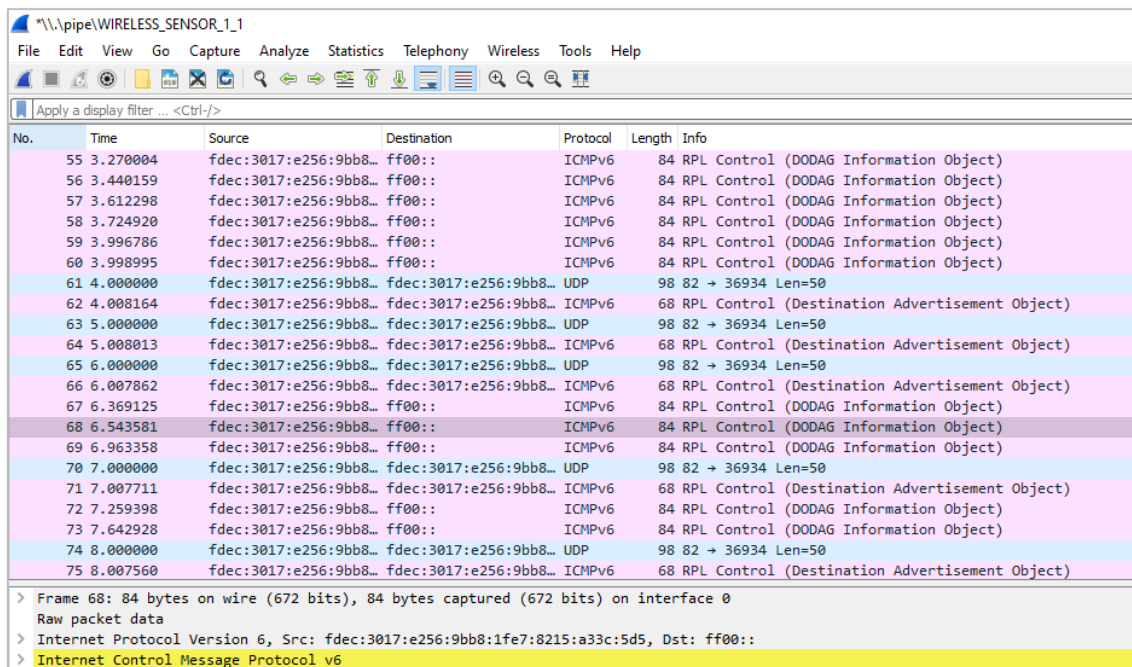


Figure 3-9: Wireshark captures RPL control messages such as DAO, DIO etc.

Following is a screenshot of a DIO message where the Rank information is highlighted as shown Figure 3-10.

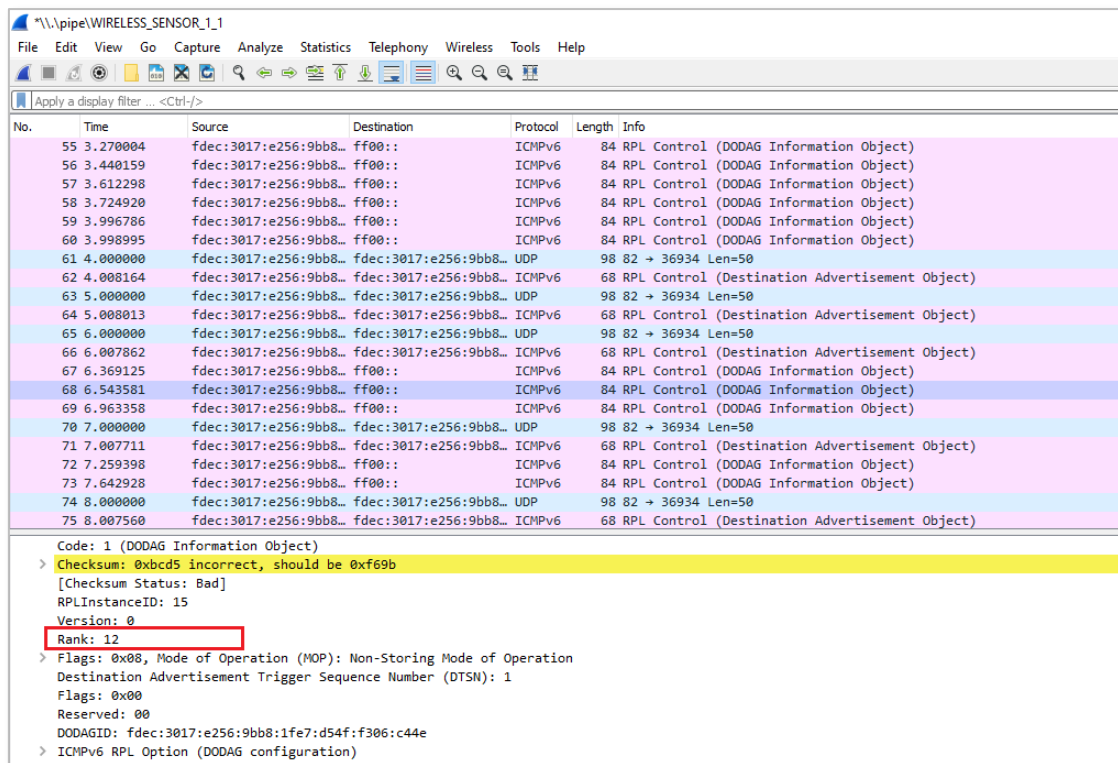


Figure 3-10: DIO message with Rank information in Wireshark

3.3 MAC / PHY: 802.15.4 Overview

IEEE802.15/TG4 formulated the IEEE802.15.4 for low-rate wireless personal area network, i.e., LR-WPAN. The standard gives priority to low-power, low-rate and low-cost.

In NetSim, the WSN part of the IOT Network runs 802.15.4 in MAC and PHY. The features implemented are:

- PHY rate is fixed at 250 Kbps; it does not vary per the received SNR.
- Receive sensitivity is a GUI parameter modifiable by users; the default value is -85 dBm
- Superframe
 - Beacon enabled and beacon disabled mode.
 - In beacon enabled mode NetSim supports slotted CSMA/CA with Active & Inactive Period (controlled by Beacon order and super-frame order parameters)
 - GTS is not implemented.
- Data Transfer Model
 - Device to coordinator, coordinator to device and device to device (peer to peer topology)
 - AckRequestFlag: If set the device acknowledges successful reception of the data frame by transmitting an ack frame.

- Frames
 - Beacon
 - Data
 - Acknowledgement
- CSMA / CA Mechanism
 - Non-beacon mode uses unslotted CSMA/CA.
 - Beacon mode uses slotted CSMA/CA.
- Energy Model
 - Energy sources: Main Line and Battery
 - Energy Harvesting which uses recharging current to replenish battery energy.
 - Consumption Modes: Transmit, Receive, Idle and Sleep

3.3.1 CSMA/CA Implementation in NetSim

- In both Slotted and Unslotted CSMA/CA cases, the CSMA/CA algorithm is based on backoff periods, where one backoff period is equal to `aUnitBackoffPeriod` which is 20 symbols long.
- This is the basic time unit of the MAC protocol and the access to the channel can only occur at the boundary of the backoff periods. In slotted CSMA/CA the backoff period boundaries must be aligned with the super-frame slot boundaries whereas in unslotted CSMA/CA the backoff periods of one device are completely independent of the backoff periods of any other device in a PAN.
- The CSMA/CA mechanism uses three variables to schedule the access to the medium:
 - NB is the number of times the CSMA/CA algorithm was required to backoff while attempting access to the current channel. This value is initialized to zero before each new transmission attempt.
 - CW is the contention windows length, which defines the number of backoff periods that need to be clear of channel activity before starting transmission. CW is only used with the slotted CSMA/CA. This value is initialized to 2 before each transmission attempt and reset to 2 each time the channel is assessed to be busy.
 - BE is the backoff exponent, which is related to how many backoff periods a device must wait before attempting to assess the channel activity.
- In beacon-enabled mode, each node employs two system parameters: *Beacon order (BO)* and *Superframe Order (SO)*.

- The parameter BO decides the length of beacon interval (BI), where $BI = aBaseSuperframeDuration \times 2^{BO}$ symbols and $0 \leq BO \leq 14$; while the parameter SO decides the length of superframe duration (SD),
- where $SD = aBaseSuperframeDuration \times 2^{SO}$ symbols and $0 \leq SO \leq BO \leq 14$.
- The value of $aBaseSuperframeDuration$ is fixed to 960 symbols. The format of the superframe is defined as shown in the following Figure 3-11.

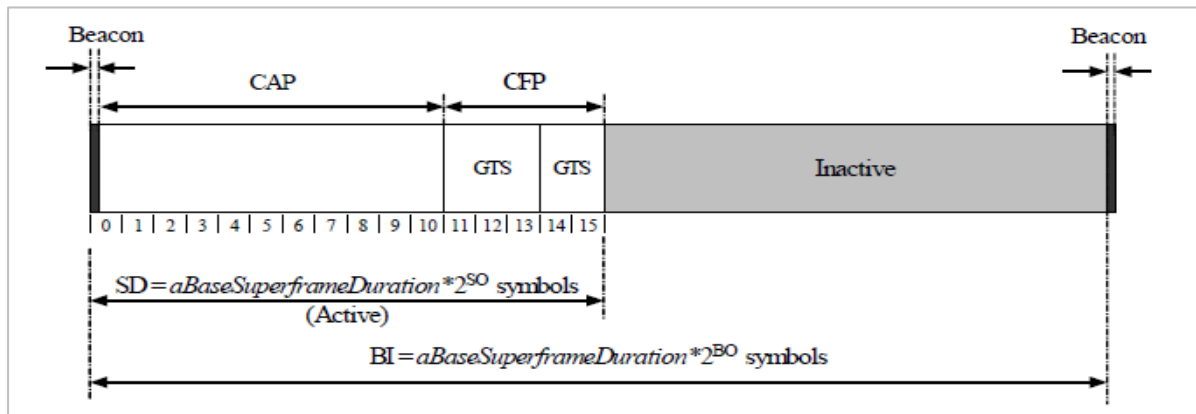


Figure 3-11: The format of the Superframe structure

- Furthermore, the active portion of each Superframe consists of three parts: beacon, CAP, and CFP, which is divided into 16 equal length slots. The length of one slot is equal to $aBaseSlotDuration \times 2^{SO}$ symbols, where $aBaseSlotDuration$ is equal to 60 symbols.
- In CAP, each node performs the CSMA/CA algorithm before transmitting data packet or control frame. Each node maintains three parameters: the number of backoffs (NB), contention window (CW), and backoff exponent (BE).
- The initial values of NB, CW, and BE are equal to 0, 2, and *Min Backoff Expo*, respectively, where *Min Backoff Expo* is by default 3 and it can be set up to 8.
- For every backoff period, node takes a delay for random backoff between 0 and $2^{BE} - 1$ Unit backoff Time (UBT), where UBT is equal to 20 symbols (or 80 bits).
- A node performs clear channel assessment (CCA) to make sure whether the channel is idle or busy, when the number of random backoff periods is decreased to 0.
- The value of CW will be decreased by one if the channel is idle; and the second CCA will be performed if the value of CW is not equal to 0. If the value of CW is equal to 0, it means that the channel is idle; then the node starts data transmission.
- However, if the CCA is busy, the value of CW will be reset to 2; the value of NB is increased by 1; and the value of BE is increased by 1 up to the maximum BE (*Max Backoff Expo*), where the value *Max Backoff Expo* is by default 5 and can be up to 8.

- The node will repeatedly take random delay if the value of NB is less than the value of Max CSMA BO (*macMaxCSMABackoff*), where the value of Max CSMA BO is equal to 4; and the transmission attempt fails if the value of NB is greater than the value of Max CSMA BO.

3.3.2 Beacon Order and Super Framer Order

Beacon frame is one of the management frames in IEEE 802.15.4 based WSNs and contains all the information about the network. A coordinator in a PAN can optionally bound its channel time using a Superframe structure which is bound by beacon frames and can have an active portion and an inactive portion. The coordinator enters a low-power (sleep) mode during the inactive portion.

The structure of this Superframe is described by the values of *macBeaconOrder* and *macSuperframeOrder*. The MAC PIB attribute *macBeaconOrder*, describes the interval at which the coordinator shall transmit its beacon frames. The value of *macBeaconOrder*, BO, and the beacon interval, BI, are related as follows:

For $0 \leq BO \leq 14$, $BI = aBaseSuperframeDuration \times 2^{BO}$ symbols

If BO = 15, the coordinator shall not transmit beacon frames except when requested to do so, such as on receipt of a beacon request command. The value of *macSuperframeOrder*, SO shall be ignored if BO = 15.

If SuperFrame Order (SO) is same as Beacon Order (BO) then there will be no inactive period and the entire SuperFrame can be used for packet transmissions. If BO=10, SO=9 half of the Superframe is inactive and so only half of Superframe duration is available for packet transmission. If BO=10, SO=8 then $\left(\frac{3}{4}\right)^{th}$ of the Superframe is inactive and so nodes have only $\left(\frac{1}{4}\right)^{th}$ of the Superframe time for transmitting packets.

3.4 Energy Models: Sources, Consumption and Harvesting

Wireless nodes, especially sensors, possess limited processing capability, storage and energy resources. The life of the sensor nodes i.e., the energy consumption during its operation, is critical to the network performance. Therefore, researchers often need to study energy consumption at the devices and in the network.

NetSim has a dedicated energy models at sensor nodes (WSN/IoT networks) for modelling energy sources, energy consumption and energy harvesting.

The power model is user configurable and can be found in the ZigBee Interface properties of the Sensor nodes as shown below Figure 3-12. The default settings are as per Reference document [1].

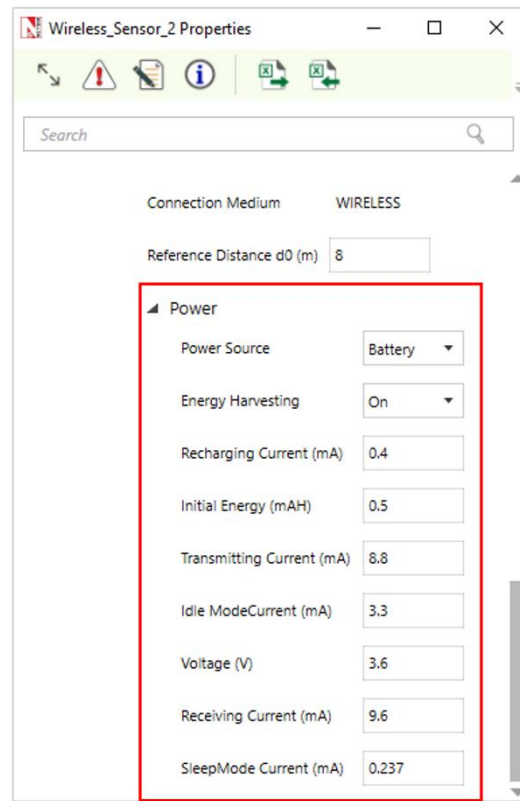


Figure 3-12: Power model properties window

The power source represents the source of energy. Each node has its own single source of power. Main line power source is assumed to have infinite energy while batteries have limited initial energy. When energy harvesting is turned on, it replenishes the battery energy. If the power of a node is completely depleted the node can no longer operate.

The different currents used in the Sensor Battery model calculations are:

- TransmitCurrent
- ReceiveCurrent
- IdleModeCurrent
- SleepModeCurrent

The energy consumed in each of these activities would be.

$$\text{TransmitEnergy} = \text{TransmitCurrent} \times \text{Voltage} \times \text{Time (for which node transmits packets)}$$

$$\text{ReceiveEnergy} = \text{ReceiveCurrent} \times \text{Voltage} \times \text{Time (for which node receives packets)}$$

$$\text{IdleModeEnergy} = \text{IdleModeCurrent} \times \text{Voltage} \times \text{Time (in Idle mode)}$$

$$\text{SleepModeEnergy} = \text{SleepModeCurrent} \times \text{Voltage} \times \text{Time (in sleep mode)}$$

$$TotalEnergy = TransmitEnergy + ReceiveEnergy + IdleModeEnergy + SleepModeEnergy$$

NetSim also has an Energy-Harvesting Model which is modelled as

$$EnergyHarvested = RechargingCurrent \times Voltage \times Time$$

Hence,

$$BatteryEnergy (at any time) = InitialEnergy - TotalEnergyConsumed + EnergyHarvested$$

Energy consumption is calculated individually for each sensor node that is part of the network scenario. The sensors have various Radio States such as SLEEP, TRX_ON_BUSY, RX_ON_IDLE, RX_ON_BUSY, RX_OFF. As explained in the formulas above the energy consumed is proportional to the time for which the node is a particular state. For example, the time for which a node transmits a packet is equal to the time for which the node is in TRX_ON_BUSY state. This duration in turn depends on the protocol operation. Thus, the correlation between protocol operation and energy consumption.

The units in NetSim for current is mA, for Voltage is V and for Total-Energy-Consumed is mJ. The Unit for Initial-Energy is mAH and this is converted to mJ for calculations since the output metrics are in mJ. The Initial energy in mAH is converted to mJ using the formula:

$$Initial\ Energy\ (mJ) = Initial\ Energy\ (mAH) \times Voltage\ (V) \times 3600$$

For example, if we set *Initial energy* = 0.5mAH, and if the voltage is 1V then *Initial Energy (mJ)* = 1 × (0.5 × 3600) = 1800mJ

Post simulation NetSim outputs an Energy Metrics table which provides energy consumption of each device with respect to Transmission, Reception, Idle Mode, and Sleep Mode as shown below Figure 3-13.

Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_1	1800.000000	30.295429	1773.310601	0.392163	0.458035	29.445230	0.000000
WIRELESS_SENSOR_2	1800.000000	30.116616	1773.489413	0.114127	0.450355	29.552134	0.000000
WIRELESS_SENSOR_3	1800.000000	30.116616	1773.489413	0.114127	0.450355	29.552134	0.000000
WIRELESS_SENSOR_4	1800.000000	30.116616	1773.489413	0.114127	0.450355	29.552134	0.000000
WIRELESS_SENSOR_5	1800.000000	30.296314	1773.309715	0.120050	0.718541	29.457724	0.000000

Figure 3-13: Battery model Table in result window showing various categories of energy consumption for each device. The categories are Initial energy, consumed energy, remaining energy, transmitting energy, receiving energy, idle energy and sleep energy.

3.4.1 Energy Model source code

The Energy consumed by the sensor devices are computed in the function `battery_set_mode()` present in the `BatteryModel.c` file which belongs to the `BatteryModel` project. This is called in the function `fn_NetSim_Zigbee_ChangeRadioState()` present in the `ChangeRadioState.c` file which belongs to `ZigBee` project. The protocol operations decide the time for which the radio is in a particular state. The energy calculation function then multiplies this time by the current drawn and the voltage.

Users can implement energy aware protocols by accessing the information such as remaining energy of each node, when modifying the source code.

3.5 Sensor Application and how to model sensing interval?

Agents and sensing range were used in earlier versions (before v11) of NetSim as an abstraction of physical phenomenon to trigger packet generation in the sensor nodes respectively. Sensor nodes generate packets whenever they sense an agent within the sensor range. From NetSim v11 onwards users have the facility to configure traffic in the sensor network using the application as shown Figure 3-14.

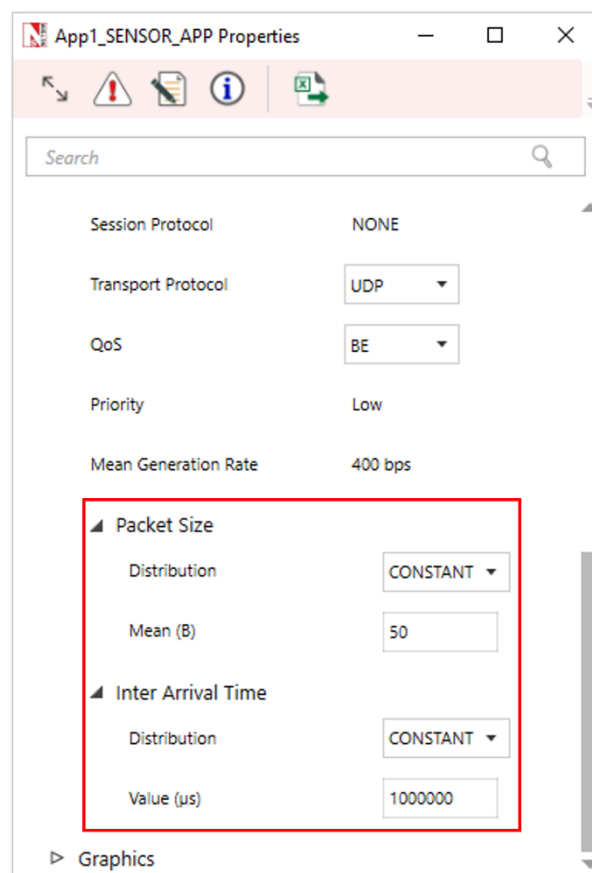


Figure 3-14: Application window

In the application properties, the size of the packet can be set under packet size and the Inter-arrival time can be thought of as sensor interval.

Users can now configure traffic between sensor and sink node as well as between the sensor nodes.

Note that Agents, sensor interval, sensor range are deprecated in NetSim v11.

3.6 WSN/IOT File Based Placement

File based placement, as the name suggests is an option that can be used to place devices in user defined locations based on the text file which is provided as the input.

Why do we need File Based Placement?

- File Based Placement gives completely a user-defined approach for device placements during the process of Network design.
- This feature allows the user to design a large network scenario comprising of various devices with ease.
- It allows device placements with precision and so on.

Create a text file as per the following or use the file present in the Docs folder of NetSim Install Directory < C:\Program Files\NetSim Standard\Docs\Sample_Configuration\IOT>

3.6.1 Internet of Things

The text file that we give as an input can be saved as follows: **IOT_File_Based_Placement.txt**

The general format to be followed while creating an IOT_File_Based_Placement.txt for all the devices used in it is given below:

<DEVICE_NAME>,<DEVICE_TYPE>,<X>,<Y>

where,

DEVICE_NAME represents the name of the device and can be user defined.

DEVICE_TYPE represents the type of device, and this info can be obtained from the "General Properties" of that particular device.

X represents the X_Coordinate position of the device upon the grid.

Y represents the Y_Coordinate position of the device upon the grid.

NOTE: Once we give a file-based input for device placement, an ad-hoc link will automatically be established connecting all the devices pertaining to it. And users need to manually connect the remaining devices using the Wired/Wireless links.

Must the IOT text file contain only IOT devices?

The IOT txt file can include all the devices that are present in the top ribbon/toolbar when we select Internet of Things from the home screen. This varies based on the network type. For e.g. WSN and MANETs network types support different devices comparatively.

IOT_File_based_placement.txt

Wireless_Sensor,IOT_Sensors,0,0

Wireless_Sensor,IOT_Sensors,10,10

Wireless_Sensor,IOT_Sensors,20,20

Wireless_Sensor,IOT_Sensors,30,30

Wireless_Sensor,IOT_Sensors,40,20

Low_Pan_Gateway,LOWPAN_Gateway,50,10

Router,IOT_ROUTER,60,20

Wired_Node,WIREDNODE,70,20

Open NetSim and click **New Simulation → Internet Of Things**. In the Fast Config window, Choose the **File Based Placement** option under Automatic Placement and give the path of the text file as shown below Figure 3-15.

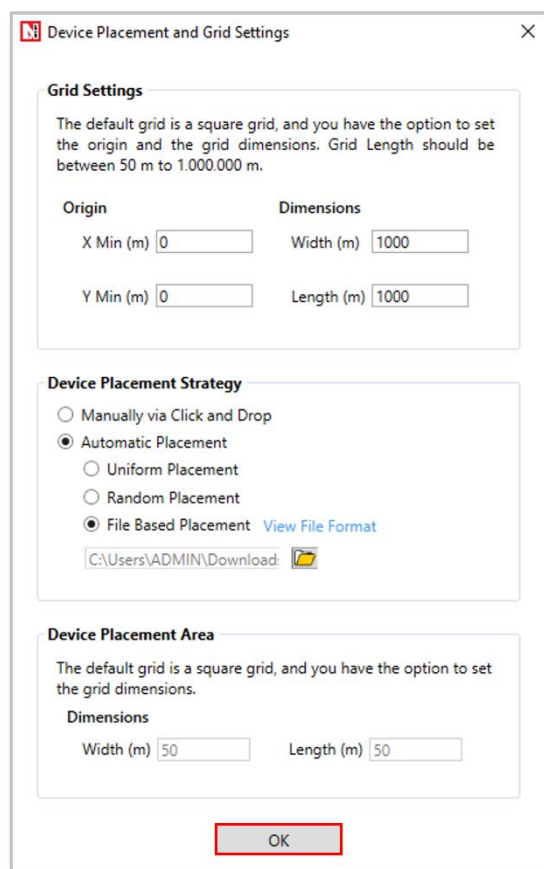


Figure 3-15: Device placement Strategy to File based Placement in IOT

After giving the path, Click on OK. It will display the IOT network as shown below, where all devices are placed as per the positions given in the text file.

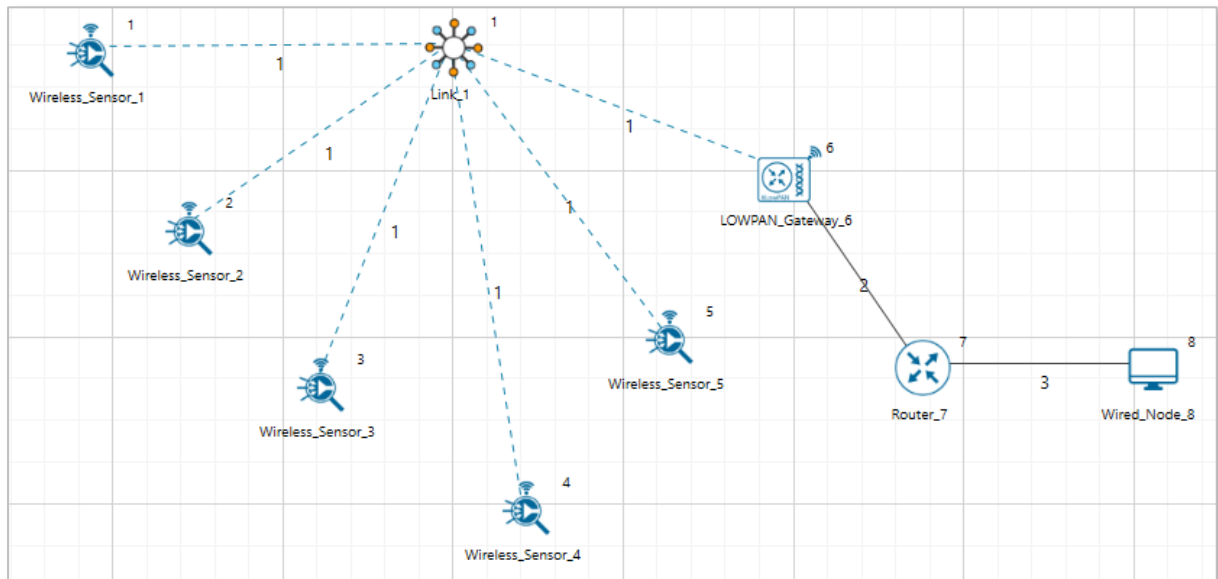


Figure 3-16: Network Topology in IOT

Connect Low_Pan_Gateway to Router and Router to Wired node. Configure application and run simulation.

3.6.2 Wireless Sensor Networks

Create a text file as per the following or use the file present in the Docs folder of NetSim Install Directory < C:\Program Files\NetSim Standard\Docs\Sample_Configuration\WSN>

WSN_File_based_placement.txt

Wireless_Sensor,Sensors,5,5

Wireless_Sensor,Sensors,10,10

Wireless_Sensor,Sensors,20,10

Wireless_Sensor,Sensors,5,10

WSN_Sink,SinkNode,10,15

Open NetSim and click **New Simulation** → **Wireless Sensor Networks**. Select **File Based Placement** option under Automatic Placement and give the path of the text file as shown below Figure 3-17.

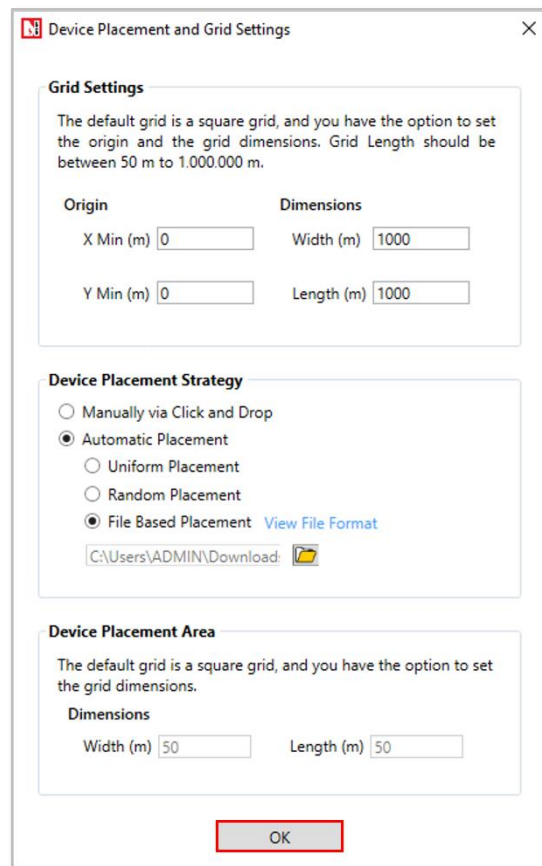


Figure 3-17: Device placement Strategy to File based Placement in WSN

After giving the path, Click on OK. It will display the WSN network as shown below, where all devices are placed as per the positions given in the text file.

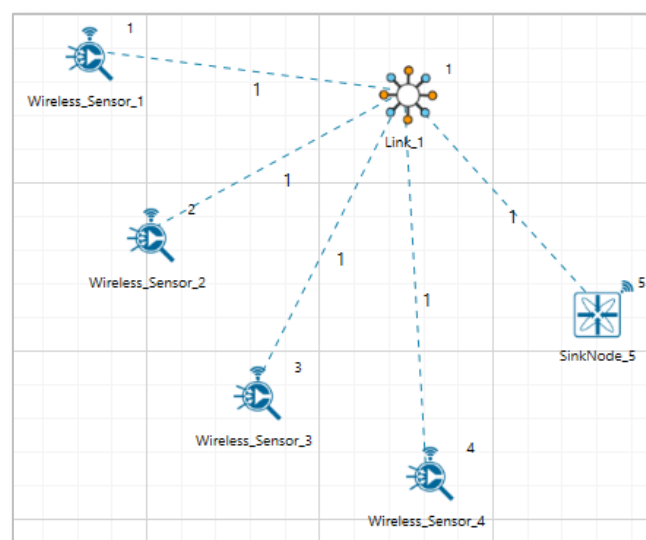


Figure 3-18: Network Topology in WSN

NOTE: Please refer to section “2.1 Fast configuration” for more information.

3.7 Radio measurements log file

NetSim ZigBee 802_15_4 Radio measurements csv log file records pathloss, shadowing loss, fading loss, received power, transmitted power, SNR, BER. This log file can be enabled from the Options->Enable Logs as explained in section 8.7 of NetSim User Manual.

By default, the log is written for all transmitters and receivers in the network. To get the log written only for specific receivers in the network, the `DEVICE_ID_LIST` defined in the `802_15_4_RadioMeasurements.c` file can be modified suitably.

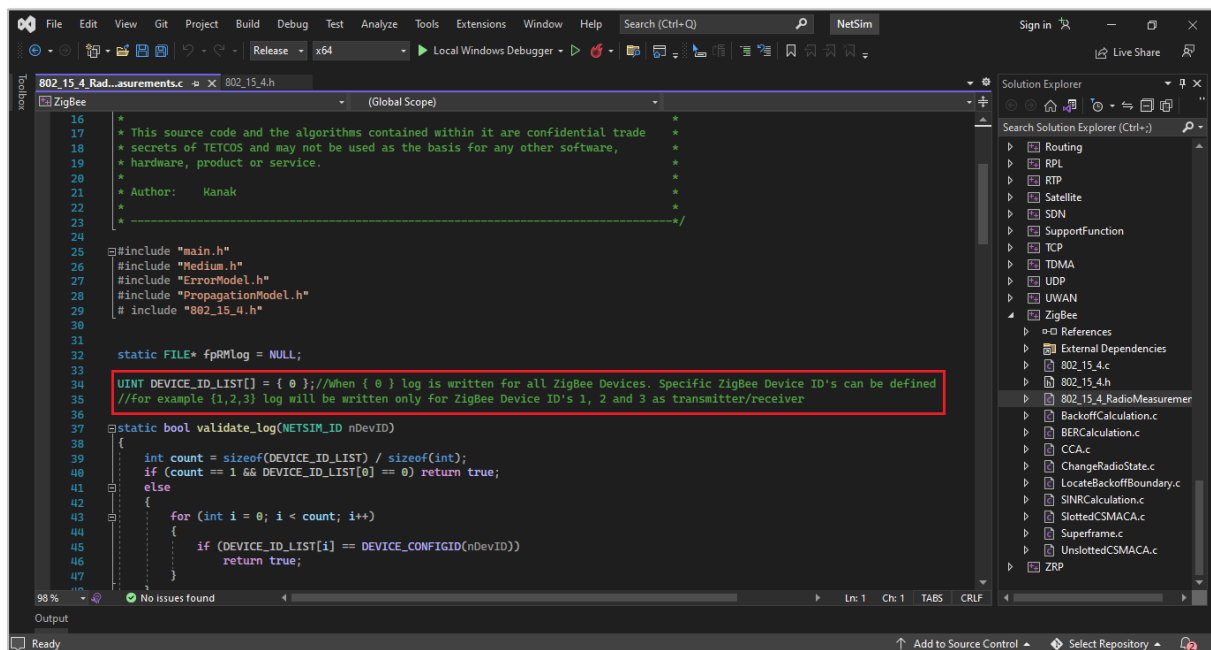


Figure 3-19: Highlight showing how to write the log for specific transmitters/receivers in the network

For example, if the log is to be written only for DEVICE ID's 10, 11 and 12 then the list can be modified as shown below:

```
UINT DEVICE_ID_LIST[] = { 10, 11, 12 };
```

The `IEEE802_15_4_RADIO_MEASUREMENTS_LOG.csv` file will contain the following information:

- Time in Milliseconds
- Transmitter Name
- Receiver Name
- Distance between the Transmitter and the Receiver in meters
- Packet ID
- Packet Type
- Control Packet Type

- Transmitter Power in dBm
- Total Loss in dB
- Pathloss in dB
- Shadowing Loss in dB
- Fading Loss in dB
- Received Power in dBm
- SNR in dB
- BER

The log file can be accessed from the Simulations Results Window under the log file drop down in the left pane.

The screenshot displays the 'Simulation Results' window with a sidebar on the left containing a tree view of simulation metrics. The 'Log Files' section is highlighted with a red box, showing a list of log files, including '802_15_4_RADIO_MEASUREMENTS_LOG.csv'. The main area of the window is divided into four panes, each displaying a table of simulation results:

- Application_Metrics_Table**: Shows application-level metrics.

Application ID	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (ms)
1	App1_SENSOR_APP	100	100	0.000400	4662
- TCP_Metrics_Table**: Shows TCP-level metrics.

Source	Destination	Segment Sent	Segment Received	Ack Sent	Ack Received
WIRELESS_SENSOR_1	ANY_DEVICE	0	0	0	0
WSN_SINK_2	ANY_DEVICE	0	0	0	0
- Link_Metrics_Table**: Shows link-level metrics.

Link ID	Link Throughput Plot	Packets Transmitted	Packets Received	Packets Collided
All	NA	100	103	0
1	NA	100	103	0
- Queue_Metrics_Table**: Shows queue-level metrics. The table is currently empty, displaying 'No content in table'.

Figure 3-20: IEEE802_15_4_RADIO_MEASUREMENTS_LOG.csv file highlighted in the Results window.

Time(us)	Transmitter Name	Receiver Name	Distance(m)	Packet ID	Packet Type	Control Packet Type	Tx_Power(dBm)	Path Loss(dB)	Shadowing Loss(dB)	Fading Loss(dB)
512	WSN_SINK_3	WIRELESS_SENSOR	150.163828	0	Control_Packet Zigbee_BEACON_FRAM		0	83.5776	0	0
512	WSN_SINK_3	WIRELESS_SENSOR	89.366432	0	Control_Packet Zigbee_BEACON_FRAM		0	79.069782	0	0
985504	WIRELESS_SENSOR_1	WIRELESS_SENSOR	145.92	0	Control_Packet DSR_RREQ		0	83.32859	0	0
985504	WIRELESS_SENSOR_1	WSN_SINK_3	150.163828	0	Control_Packet DSR_RREQ		0	83.5776	0	0
989344	WIRELESS_SENSOR_1	WIRELESS_SENSOR	145.92	0	Control_Packet DSR_RREQ		0	83.32859	0	0
992192	WSN_SINK_3	WIRELESS_SENSOR	150.163828	0	Control_Packet DSR_RREP		0	83.5776	0	0
992576	WIRELESS_SENSOR_1	WSN_SINK_3	150.163828	0	Control_Packet Zigbee_ACK		0	83.5776	0	0
995552	WIRELESS_SENSOR_2	WIRELESS_SENSOR	145.92	0	Control_Packet DSR_RREQ		0	83.32859	0	0
995552	WIRELESS_SENSOR_2	WSN_SINK_3	89.366432	0	Control_Packet DSR_RREQ		0	79.069782	0	0
1000992	WIRELESS_SENSOR_1	WSN_SINK_3	150.08331	1	Sensing	App1_SENSOR_APP	0	83.572942	0	0
1001376	WSN_SINK_3	WIRELESS_SENSOR	150.08331	0	Control_Packet Zigbee_ACK		0	83.572942	0	0
1005152	WIRELESS_SENSOR_1	WSN_SINK_3	150.08331	2	Sensing	App1_SENSOR_APP	0	83.572942	0	0
1005536	WSN_SINK_3	WIRELESS_SENSOR	150.08331	0	Control_Packet Zigbee_ACK		0	83.572942	0	0
1008640	WSN_SINK_3	WIRELESS_SENSOR	89.022469	0	Control_Packet DSR_RREP		0	79.036287	0	0
1009024	WIRELESS_SENSOR_2	WSN_SINK_3	89.022469	0	Control_Packet Zigbee_ACK		0	79.036287	0	0
1012800	WIRELESS_SENSOR_2	WIRELESS_SENSOR	146	0	Control_Packet DSR_RREP		0	83.333351	0	0
1013184	WIRELESS_SENSOR_1	WIRELESS_SENSOR	146	0	Control_Packet Zigbee_ACK		0	83.333351	0	0
2003872	WIRELESS_SENSOR_1	WSN_SINK_3	150.08331	3	Sensing	App1_SENSOR_APP	0	83.572942	0	0
2004256	WSN_SINK_3	WIRELESS_SENSOR	150.08331	0	Control_Packet Zigbee_ACK		0	83.572942	0	0
3003872	WIRELESS_SENSOR_1	WSN_SINK_3	150.08331	4	Sensing	App1_SENSOR_APP	0	83.572942	0	0

Figure 3-21: IEEE802_15_4_RADIO_MEASUREMENTS_LOG.csv file

3.7.1 Implementation details and Assumptions

- The log is written during each packet received at the physical layer (PHY_IN). Hence the number of entries will be based on the number of packets received, by all nodes or specific nodes based on values present in the DEVICE_ID_LIST array.

3.8 Model Limitations

- NetSim currently supports only a single root in RPL.
- NetSim GUI supports only one RPL instance. Multiple RPL instances can be created by manually editing the config file.
- DODAG Repair is not supported
- Nodes (sensors) in NetSim retrieve time from the same (single) virtual clock that ticks virtual time within NetSim. As such, they can be considered as perfectly time synchronized. Real world clocks drift from a reference clock due to various reasons (heat, deficient oscillator, etc.), resulting in an offset of a few milliseconds per day. In the COTS version of NetSim, clock drift is not available. This means that it is not possible to model clocks to run with different rates and/or offsets, in different nodes/sensors.
- Security in 802.15.4 is not implemented.

4 Featured Examples

Sample configuration files for all networks are available in the Examples Menu in NetSim Home Screen. These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

4.1 IOT Example Simulations

4.1.1 Energy Model

Open NetSim, Select **Examples->IOT-WSN->Internet of Things->Energy Model** then click on the tile in the middle panel to load the example as shown in Figure 4-1.

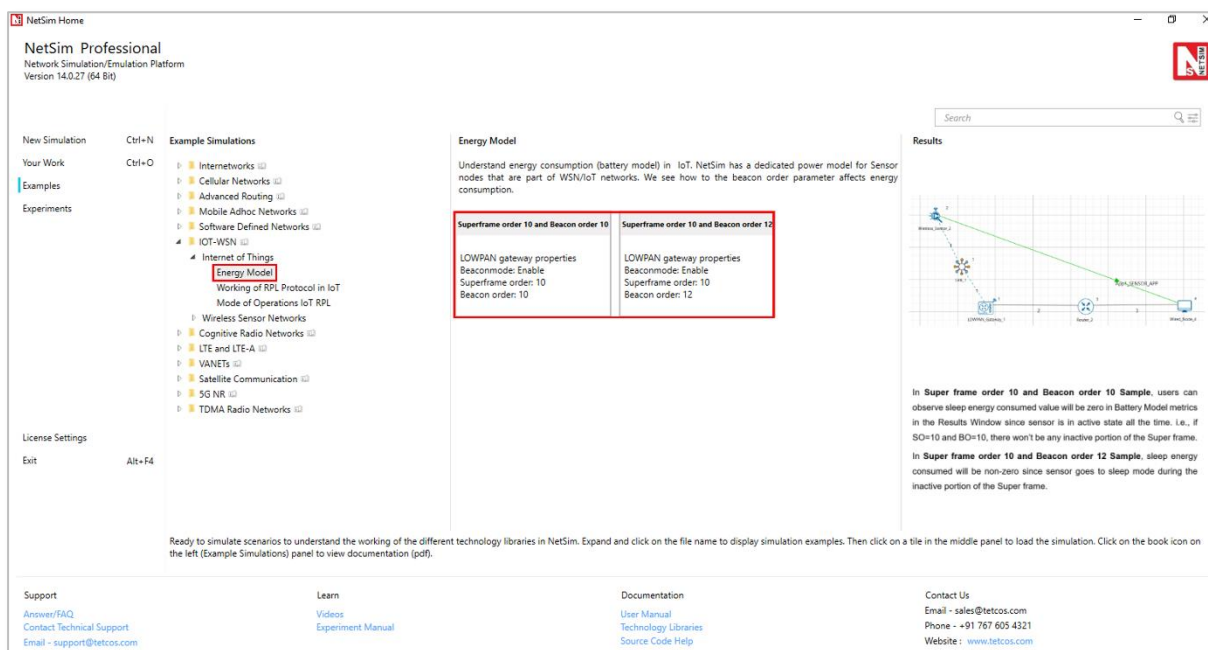


Figure 4-1: List of scenarios for the example of Energy Model

The following network diagram illustrates what the NetSim UI displays when you open the example configuration file.

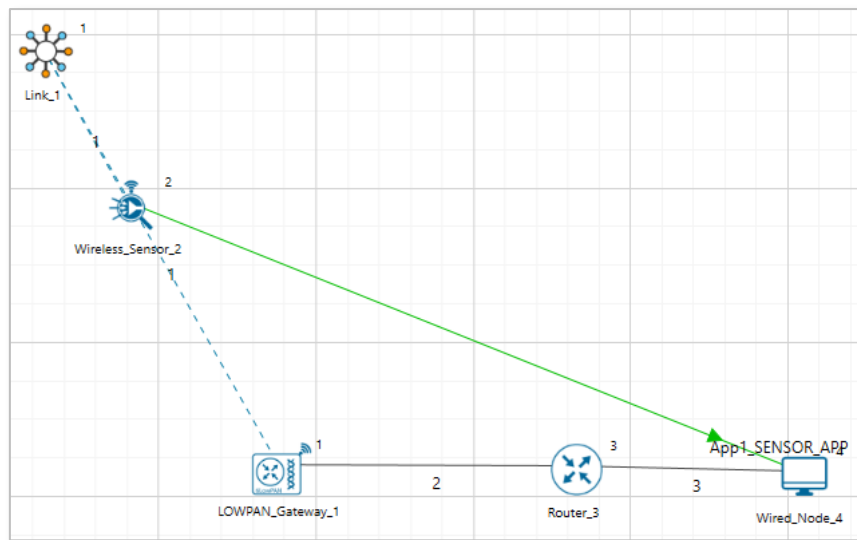


Figure 4-2: Network set up for studying the Energy Model

Settings done in sample network

1. Grid length(m) → 1000, Side Length(m) → 1000, Manually via Click and Drop.
2. In LOWPAN_Gateway change BeaconMode → Enable, Superframe Order → 10, Beacon Order → 10.
3. Channel Characteristic → NO PATHLOSS in Ad hoc link Properties.

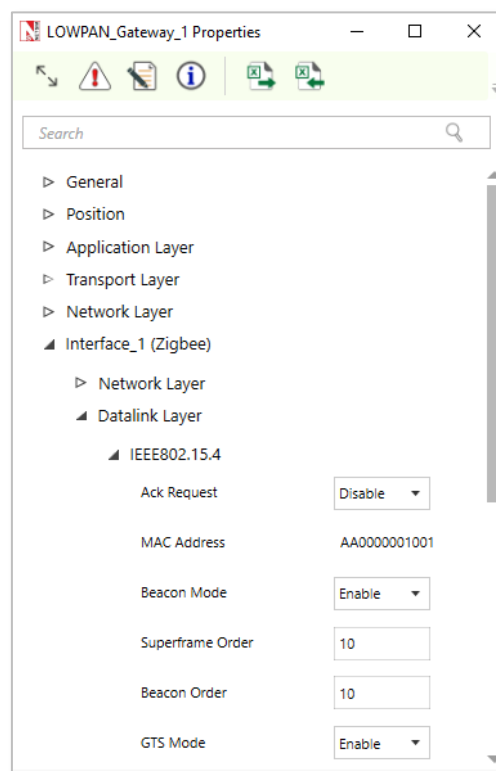


Figure 4-3: Datalink layer Properties for Lowpan Gateway

4. Click on the Application icon present in the top ribbon/toolbar

- Created Sensor Application from Sensor_1 to Wired_Node_4 with default Properties.
5. In NetSim GUI Plots are Enabled.
 6. Run the simulate for 100 sec.
 7. Check the Battery model in simulation results window, users should get zero value for Sleep energy (mJ) consumed.
 8. Go back to Simulation window change following properties in LOWPAN_Gateway for another sample.
 9. Set Superframe Order (SO) and Beacon Order (BO) as 10 and 12 respectively ($0 \leq SO \leq BO \leq 14$)
 10. Re-run the Simulate for 100 sec.
 11. Check the Battery Model metrics in metrics window, users should get non-zero value for Sleep energy consumed.

Results: In **Superframe order 10 and Beacon order 10 Sample**, users can observe sleep energy consumed value will be zero in Battery Model metrics in the Results Window since sensor is in active state all the time. i.e., if $SO=10$ and $BO=10$, there won't be any inactive portion of the Superframe.

Battery model_Table							
Battery model							
Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_2	6480.000000	1186.359073	5436.210778	13.308705	2.846638	1170.201545	0.002185

Figure 4-4: Battery Model Table for $SO=10$ & $BO=10$

In **Superframe order 10 and Beacon order 12 Sample**, sleep energy consumed will be non-zero since sensor goes to sleep mode during the inactive portion of the Superframe.

Battery model_Table							
Battery model							
Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_2	6480.000000	438.610027	6183.949973	10.551056	1.029612	369.401912	57.627449

Figure 4-5: Battery Model Table for $SO=10$ & $BO=12$

4.1.2 Working of RPL Protocol in IoT

Open NetSim, Select **Examples->IOT-WSN->Internet of Things->Working of RPL Protocol in IoT** then click on the tile in the middle panel to load the example as shown in Figure 4-6.

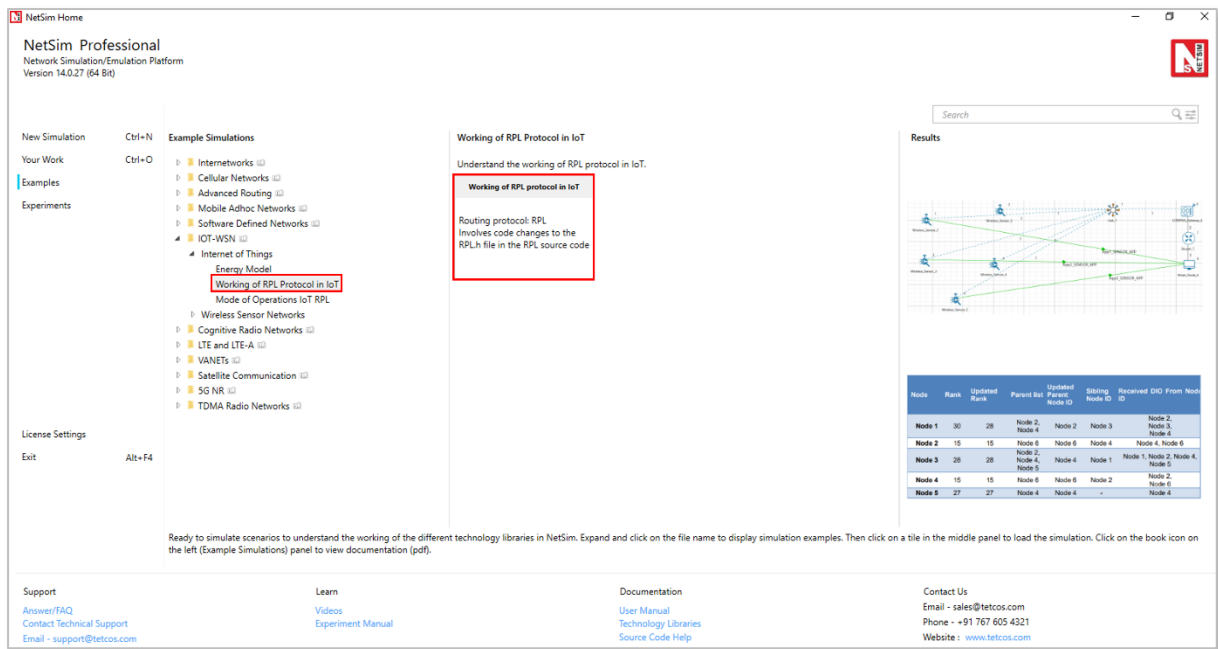


Figure 4-6: List of scenarios for the example of Working of RPL Protocol in IoT

The following network diagram illustrates, what the NetSim UI displays when you open the example configuration file.

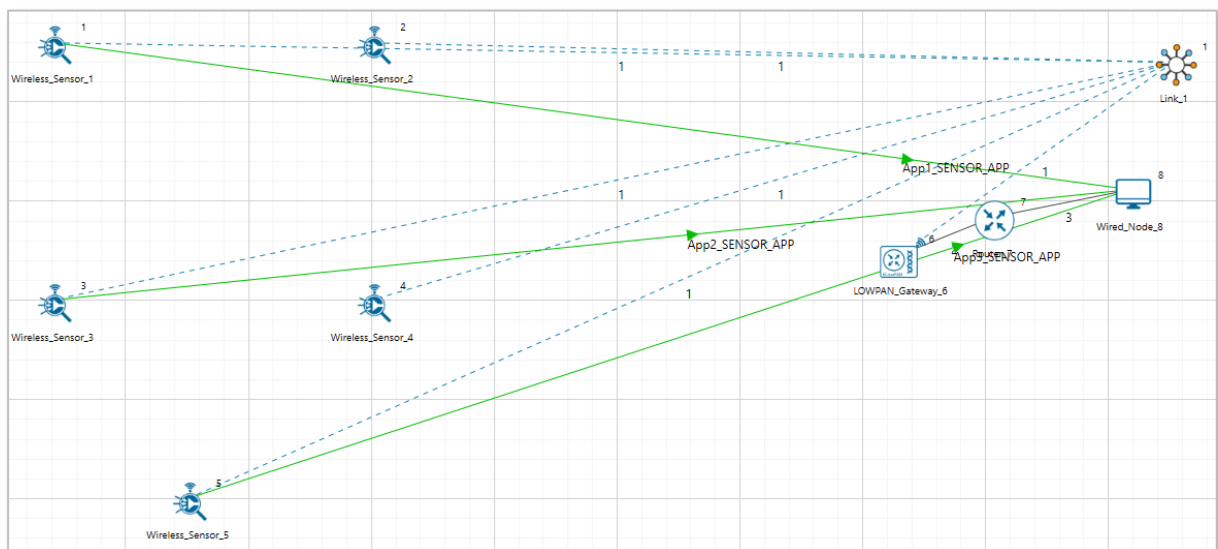


Figure 4-7: Network set up for studying the Working of RPL Protocol in IoT

Settings done in sample network:

- Grid length(m) → 100m, Side Length(m) → 50, Manually via Click and Drop.
- Routing protocol has set as RPL for 6LOWPAN Gateway and Sensor. Go to properties → Network Layer → Routing Protocol as shown Figure 4-8.

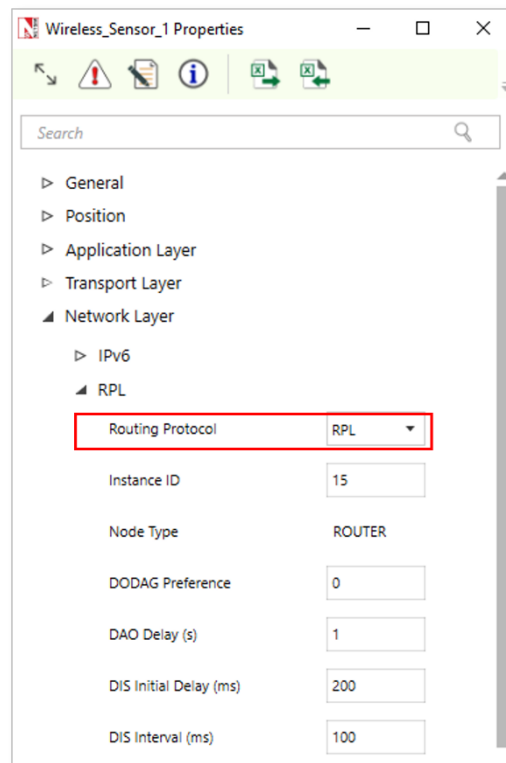


Figure 4-8: Routing Protocol to RPL in Network layer

3. In Adhoc Link Properties change Channel characteristics → Path Loss only, Path Loss Model → Log Distance and path loss exponent → 3.5
4. Application properties has set as shown in below Table 4-1.

Application Properties			
Application ID	Application Type	Source Id	Destination Id
1	SENSOR_APP	1	8
2	SENSOR_APP	3	8
3	SENSOR_APP	5	8

Table 4-1: Application properties

Procedure to get detailed RPL log file:

- Go to NetSim Home page and click on **Your work**.
- Click on **Source code** and then click on **Open Code** and open the codes in Visual Studio. Set **x64** according to the NetSim build which you are using.
- Go to the RPL Project in the Solution Explorer. Open RPL.h file and change **//#define DEBUG_RPL** to **#define DEBUG_RPL** as shown below Figure 4-9.

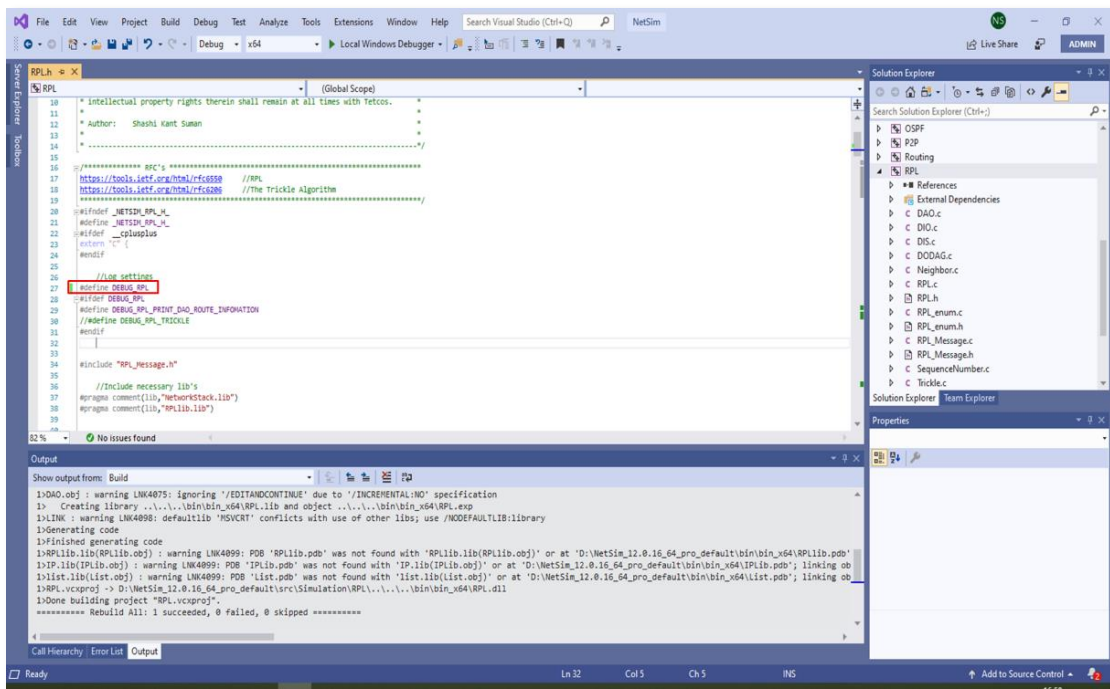


Figure 4-9: Visual Studio

- Right click on the **RPL** project in the solution explorer and click on rebuild.
- After the RPL project is rebuilt successful, go back to the network scenario.

5. In NetSim GUI Plots are Enabled. Run simulation for 100 sec and go to Result window → Log Files, open rpl log file.

Simulation Results

Network Performance

Link_Metrics

Queue_Metrics

TCP_Metrics

IP_Metrics

IP_Forwarding_Table

UDP_Metrics

IEEE802.15.4_Metrics

Battery model

Application_Metrics

Plots

Link_Throughput

Application_Throughput

Export Results (.xls/.csv)

Print Results (.html)

Open Packet Trace

Open Event Trace

Log Files

ospflog

ospf Hello.log

rpllog

Restore To Original View

Application_Metrics_Table

Application_Metrics

Detailed View

Application Id	Throughput Plot	Application Name	Packet generated	Packet receive
1	Application_Throughput_plot	App1_SENSOR_APP	100	86
2	Application_Throughput_plot	App2_SENSOR_APP	100	65
3	Application_Throughput_plot	App3_SENSOR_APP	100	17

TCP_Metrics_Table

TCP_Metrics

Detailed View

Source	Destination	Segment Sent	Segment Received	Ack Sent	Ack Receive
WIRELESS_SENSOR_1	ANY_DEVICE	0	0	0	0
WIRELESS_SENSOR_2	ANY_DEVICE	0	0	0	0
WIRELESS_SENSOR_3	ANY_DEVICE	0	0	0	0
WIRELESS_SENSOR_4	ANY_DEVICE	0	0	0	0
WIRELESS_SENSOR_5	ANY_DEVICE	0	0	0	0
6_LOWPAN_GATEWAY_6	ANY_DEVICE	0	0	0	0
ROUTER_7	ANY_DEVICE	0	0	0	0
WIRED_NODE_8	ANY_DEVICE	0	0	0	0

Link_Metrics_Table

Link_Metrics

Detailed View

Link_id	Link_throughput_plot	Packet_transmitt...	Packet_errored	Packet_collided			
		Data	Control	Data	Control	Data	Control
All	NA	773	1052	0	0	98	146
1	Link_throughput	437	1019	0	0	98	146
2	Link_throughput	168	33	0	0	0	0
3	Link_throughput	168	0	0	0	0	0

Queue_Metrics_Table

Queue_Metrics

Detailed View

Device_id	Port_id	Queued_packet	Dequeued_packet	Dropped_packet
6	2	185	185	0
7	1	16	16	0

Figure 4-10: Simulation Result window Results

```

File Edit Format View Help
rpllog - Notepad
Node '2': received a new/modified message from node '6' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '2': was isolated, now found dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '2': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 15, parents = [(6)], siblings = []
Node '4': received a new/modified message from node '6' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '4': was isolated, now found dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '4': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 15, parents = [(6)], siblings = []
Node '6',12.129ms: received dao msg with new route information. dest = FDEC:3017:E256:9888:1FE7:F726:8AAA:B60B, gateway= FDEC:3017:E256:9888:1FE7:F726:8AAA:B60B.
Node '6',16.737ms: received dao msg with new route information. dest = FDEC:3017:E256:9888:1FE7:6393:D036:18E8, gateway= FDEC:3017:E256:9888:1FE7:6393:D036:18E8.
Node '1': received a new/modified message from node '4' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '1': was isolated, now found dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '1': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 30, parents = [(4)], siblings = []
Node '2': received a new/modified message from node '4' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '2': '4' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '2': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 15, parents = [(6)], siblings = [4]
Node '3': received a new/modified message from node '4' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '3': was isolated, now found dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '3': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 28, parents = [(4)], siblings = []
Node '5': received a new/modified message from node '4' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '5': was isolated, now found dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '5': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 27, parents = [(4)], siblings = []
Node '4',27.140ms: received dao msg with new route information. dest = FDEC:3017:E256:9888:1FE7:3530:E8EA:F5E7, gateway= FDEC:3017:E256:9888:1FE7:3530:E8EA:F5E7.
Node '1': received a new/modified message from node '2' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '1': '2' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '1': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 28, parents = [4, (2)], siblings = []
Node '1': in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', updated dodag config (l_min = 3, i_doublings = 20, c_threshold = 10, max_rank_inc = 0, min_hop_rank_inc = 0)
Node '3': received a new/modified message from node '2' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '3': '2' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '3': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 28, parents = [(4), 2], siblings = []
Node '3': in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', updated dodag config (l_min = 3, i_doublings = 20, c_threshold = 10, max_rank_inc = 0, min_hop_rank_inc = 0)
Node '4': received a new/modified message from node '2' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '4': '2' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '4': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 15, parents = [(6)], siblings = [2]
Node '2',40.617ms: received dao msg with new route information. dest = FDEC:3017:E256:9888:1FE7:3530:E8EA:F5E7, gateway= FDEC:3017:E256:9888:1FE7:3530:E8EA:F5E7.
Node '1': received a new/modified message from node '3' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '1': '3' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '1': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 28, parents = [4, (2)], siblings = [3]
Node '3': received a new/modified message from node '5' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '3': '5' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '3': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 28, parents = [(4), 2, 5], siblings = []
Node '3': in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', updated dodag config (l_min = 3, i_doublings = 20, c_threshold = 10, max_rank_inc = 0, min_hop_rank_inc = 0)
Node '3': received a new/modified message from node '1' with dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482'
Node '3': '1' sent a modified DIO message and is a member of dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482', reevaluating our neighbors
Node '3': chosen parents and siblings in dodag_id = 'FDEC:3017:E256:9888:1FE7:CA28:7F00:E482': new rank = 28, parents = [(4), 2, 5], siblings = [1]

```

Figure 4-11: RPL log file

The observations of rpl log file which is generated in NetSim is given in the below Table 4-2.

Node	Rank	Updated Rank	Parent list	Updated Parent Node ID	Sibling Node ID	Received DIO From Node ID
Node 1	30	28	Node 2, Node 4	Node 2	Node 3	Node 2, Node 3, Node 4
Node 2	15	15	Node 6	Node 6	Node 4	Node 4, Node 6
Node 3	28	28	Node 2, Node 4, Node 5	Node 4	Node 1	Node 1, Node 2, Node 4, Node 5
Node 4	15	15	Node 6	Node 6	Node 2	Node 2, Node 6
Node 5	27	27	Node 4	Node 4	-	Node 4

Table 4-2: RPL Log file contains Rank, Updated Rank, parent list, Updated Parent Node ID, Sibling Node ID and Received DIO From Node ID etc.

The above table can be summarized as follows:

DIO message is sent by the root node, i.e. LowPAN Gateway, with Rank 1 to Sensor 2 and Sensor 4 as shown Figure 4-12.

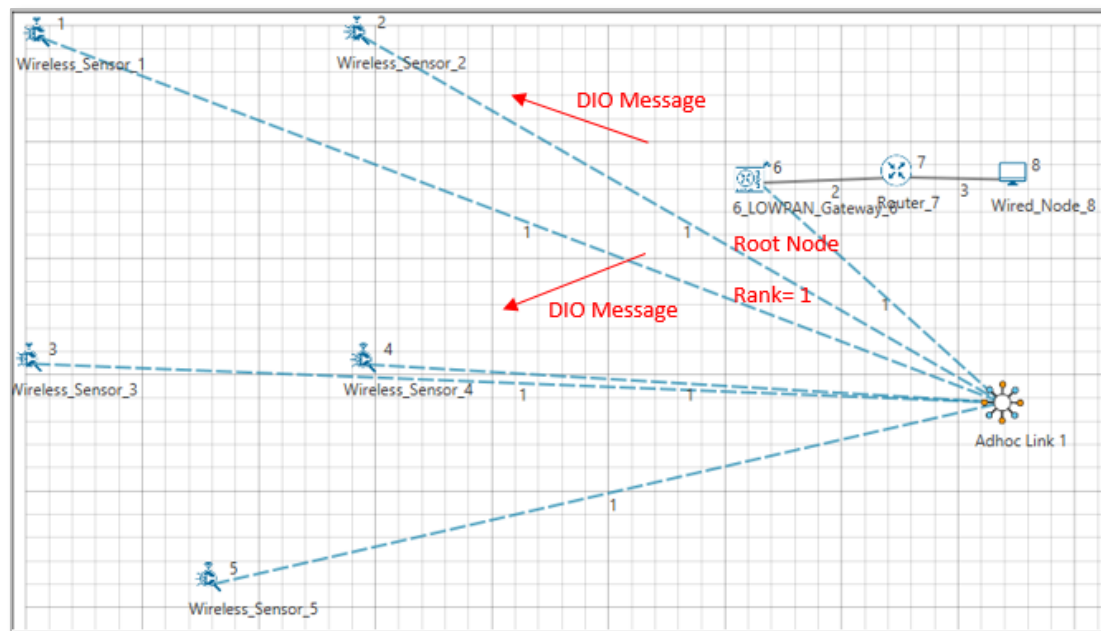


Figure 4-12: DIO messages sent by the LOWPAN Gateway to Sensors

On receiving the DIO message, Node 4 will recognize the DODAG Id of Root Node and identifies it as Parent node. Rank of Node 4 will get updated to 15. Also, Node 2 is recognized as sibling of Node 4.

Now, node 4 will broadcast DIO message to other nodes as shown Figure 4-13.

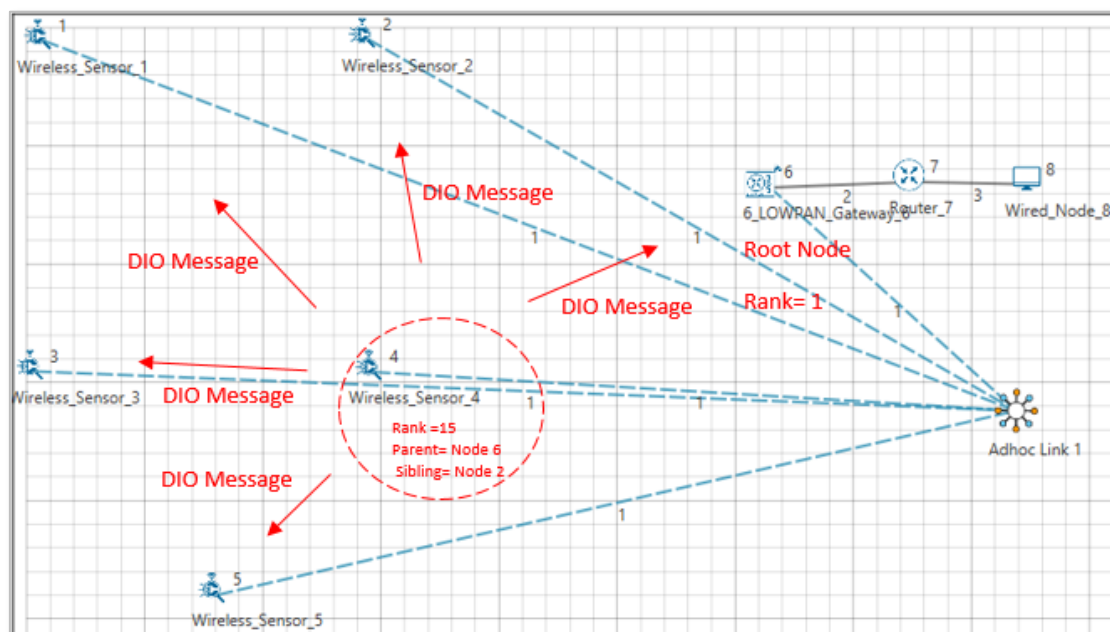


Figure 4-13: Wireless Sensor 4 broadcasting DIO message to other Sensors

Node 1 receives DIO message from Node 4 and it identifies the DODAG Id of Node 4. Hence, Node 1 recognizes Node 4 as the Parent Node. Rank of Node 1 will get updated to 30. As Node 3 is within the range of Node 1, Node 3 is identified as a sibling of Node 1.

Node 1 will then broadcast DIO messages as shown Figure 4-14.

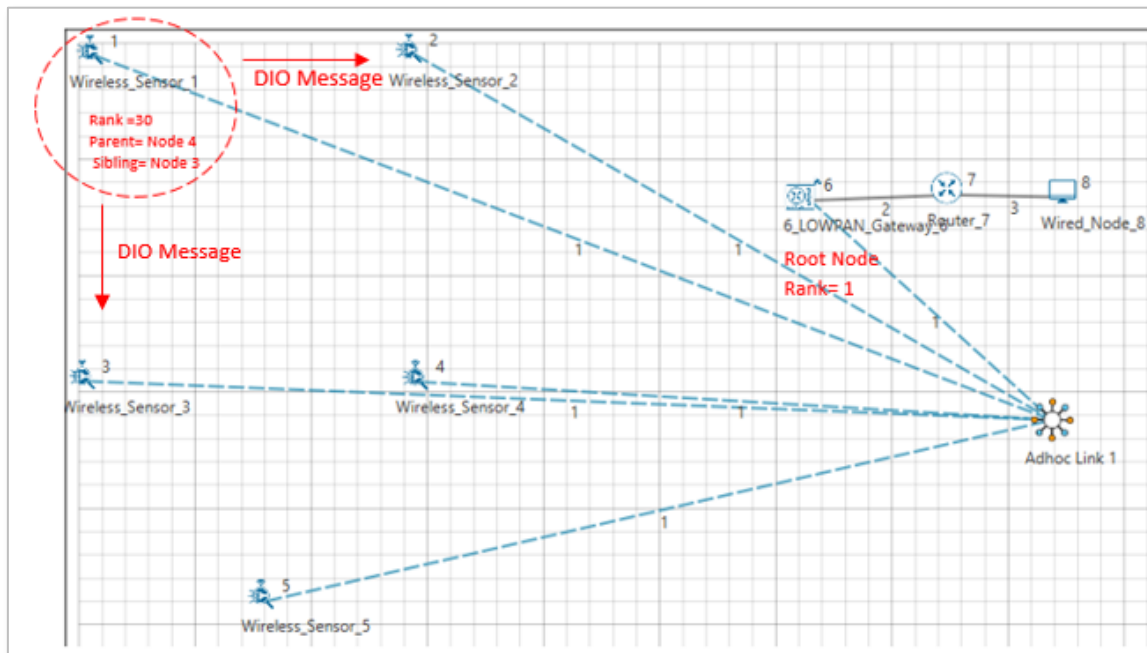


Figure 4-14: Wireless Sensor 1 broadcasting DIO message to other Sensors

Node 2 receives the DIO message from Node 6 and identifies it as Parent node. The Rank of Node 2 gets updated to 15. Node 2 now broadcasts the DIO message to other nodes as shown in Figure 4-15.

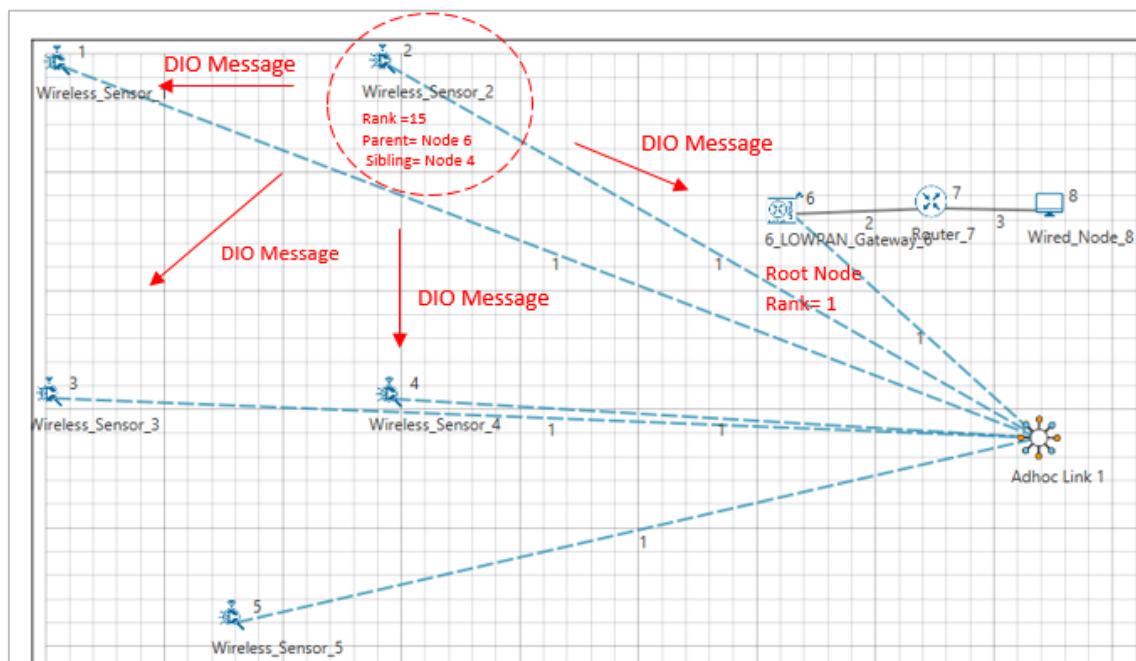


Figure 4-15: Wireless Sensor 2 broadcasting DIO message to other Sensors

Node 3 receives DIO message from Node 2 and the rank of Node 3 gets updated to 28. Node 2 is identified as the parent node of Node 3.

Node 3 then broadcasts DIO message to other nodes as shown in Figure 4-16.

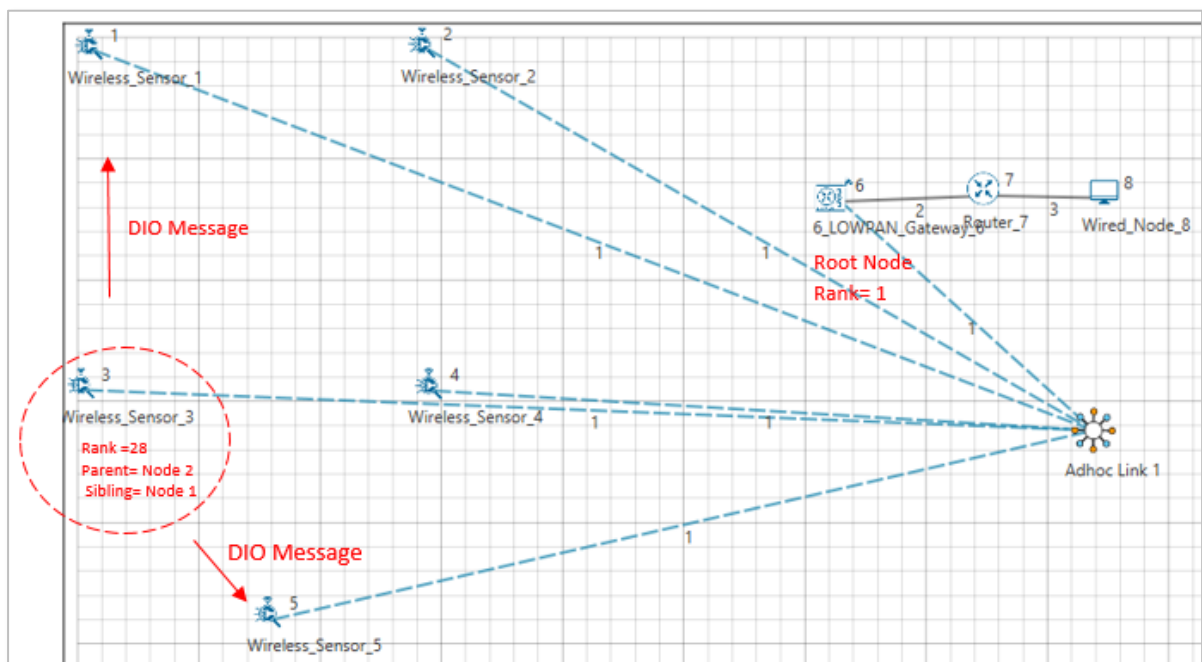


Figure 4-16: Wireless Sensor 3 broadcasting DIO message to other Sensors

Node 5 receives DIO message from Node 3, hence, it identifies Node 4 as the parent node. The rank of Node 5 gets updated to 27.

Node 5 then broadcasts DIO message as shown in Figure 4-17

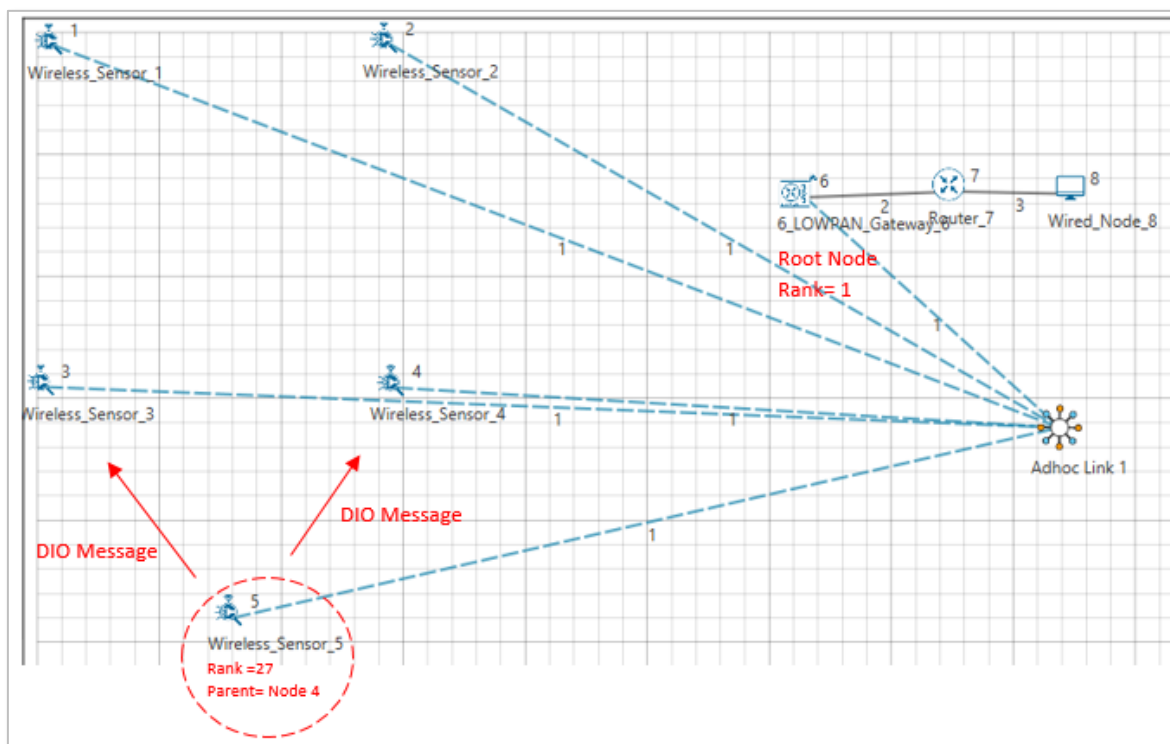


Figure 4-17: Wireless Sensor 5 broadcasting DIO message to other Sensors

Further, Node 1 receives DIO message from Node 2 and the parent list of Node 1 gets updated to Node 4 and Node 2. Also, the rank of Node 1 gets updated to 28.

According to the link quality, DODAG is formed.

- Node 2 and Node 4 are siblings and their parent is Node 6 (Root Node). Rank is 15.
- Node 1 and Node 3 are siblings.
 - Node 1 established its parent as Node 2. Rank is 28.
 - Node 3 establishes its parent as Node 4. Rank is 28.
- Node 5 doesn't have any siblings and establishes its parent as Node 4. Its rank is 27.

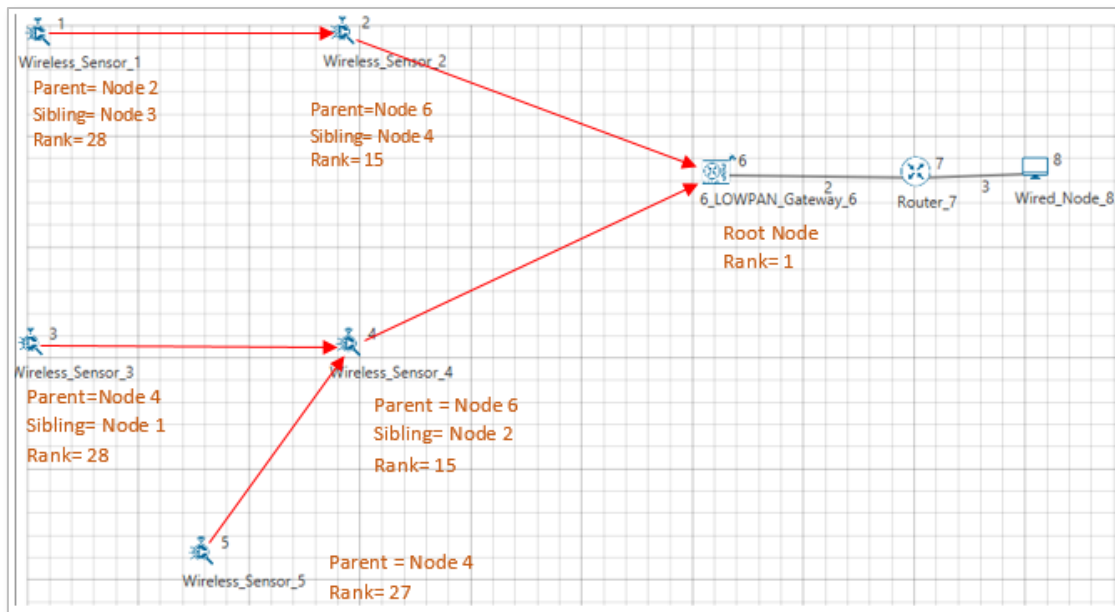


Figure 4-18: Based on link quality, DODAG is formed and Rank assigned to sensors and lowpan gateway

4.1.3 Modes of Operation in IoT RPL

The DODAG nodes process the DAO messages according to the RPL Mode of Operations (MOP), which are presented below. Independent of the MOP used, DAO messages may require reception confirmation, which should be done using DAO-ACK messages.

Although it is designed for the Multipoint-to-Point (MP2P) traffic pattern, RPL also admits the data forwarding using Point-to-Multipoint (P2MP) and Point-to-Point (P2P). In MP2P, the nodes send data messages to the root, creating an upward flow as shown in **Error! Reference source not found..** In P2MP, sometimes termed as multicast, the root sends data messages to the other nodes, producing a downward flow depicted in **Error! Reference source not found..**

In P2P, a node sends messages to the other node (non-root) of DODAG; thus, both upward and downward forwarding may be required as illustrated in **Error! Reference source not found**. RPL defines four MOPs that should be used considering the traffic pattern required by the application and the computational capacity of the nodes. In the first, MOP 0 (Point to multipoint), RPL does not maintain downward routes; thus, consequently, only MP2P traffic is enabled. In storing without multicast MOP (MOP 2 (Point to point)), downward routes are also supported, but are different from MOP 1; the nodes maintain, individually, a routing table constructed using DAO messages to provide downward traffic. Hence, downward forwarding occurs without the use of the root node, as illustrated in **Error! Reference source not found**. Storing with multicast has a functioning similar to MOP 2 (Point to point) plus the possibility of multicast data sending. This type of transmission permits the non-root node to send messages to a group of nodes formed using multicast DAOs.

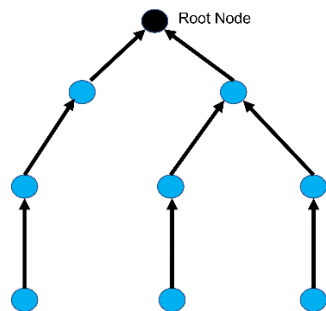


Figure 4-19: Multipoint to Point

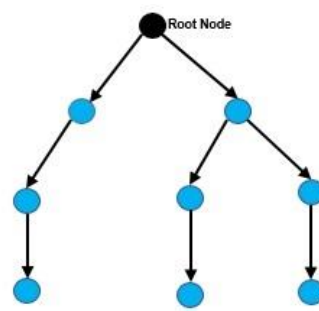


Figure 4-21: Point to Multipoint

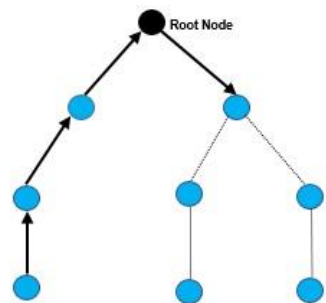


Figure 4-20: Point to Point

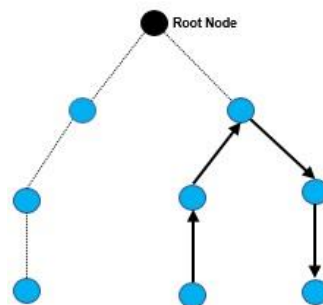


Figure 4-22: RPL Storing Mode

Open NetSim, Select **Examples->IOT-WSN->Internet of Things->Mode of Operations IoT RPL** then click on the tile in the middle panel to load the example as shown in Figure 4-23.

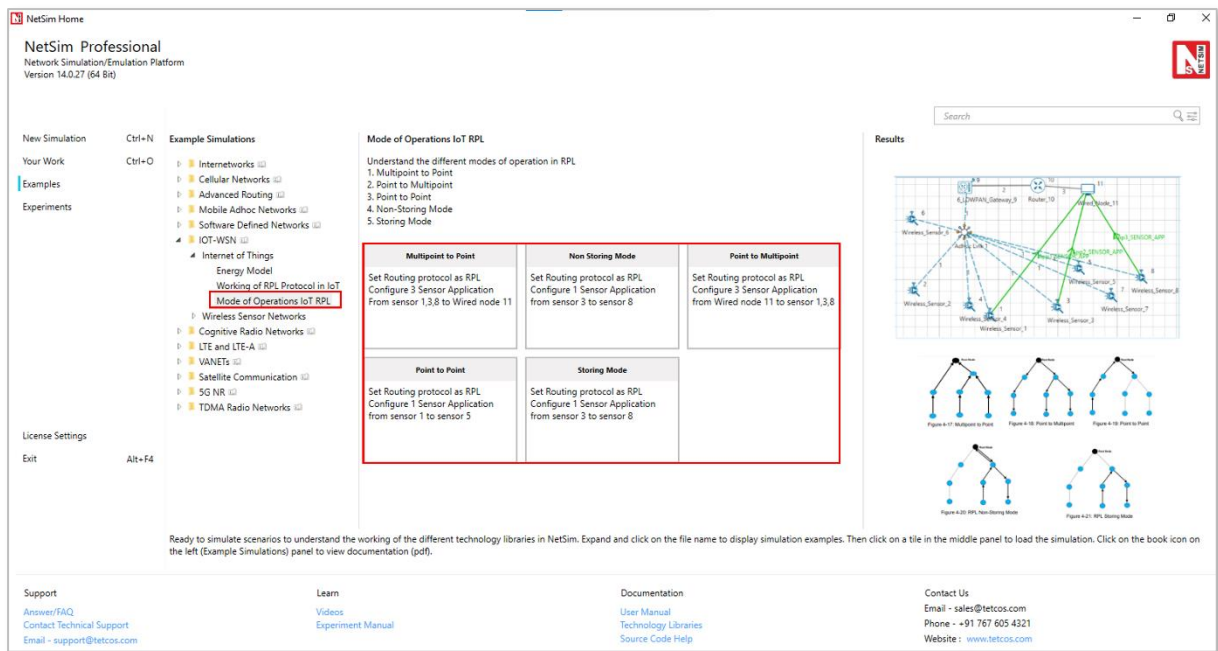


Figure 4-23: List of scenarios for the example of Mode of Operations IoT RPL

The following network diagram illustrates, what the NetSim UI displays when you open the example configuration file.

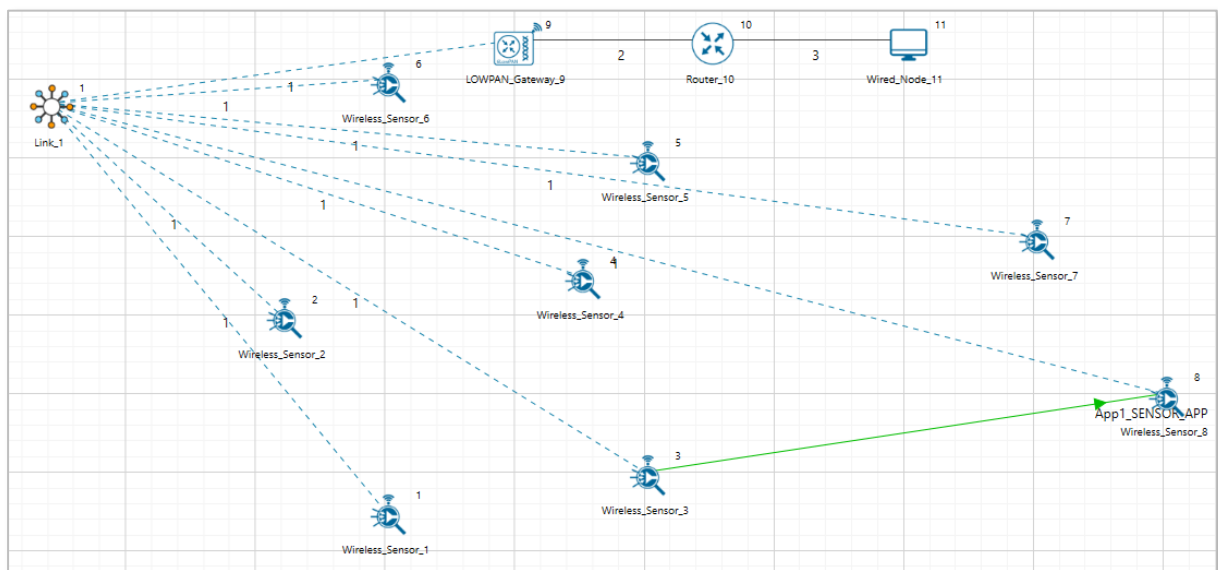


Figure 4-24: Network set up for studying the Mode of Operations IoT RPL

4.1.3.1 Multipoint to Point

Settings done in sample network

1. Grid length(m) → 200m, manually via Click and Drop.
2. Routing protocol has set as RPL for 6LOWPAN Gateway and Sensor. Go to properties → Network Layer → Routing Protocol as shown in Figure 4-25.

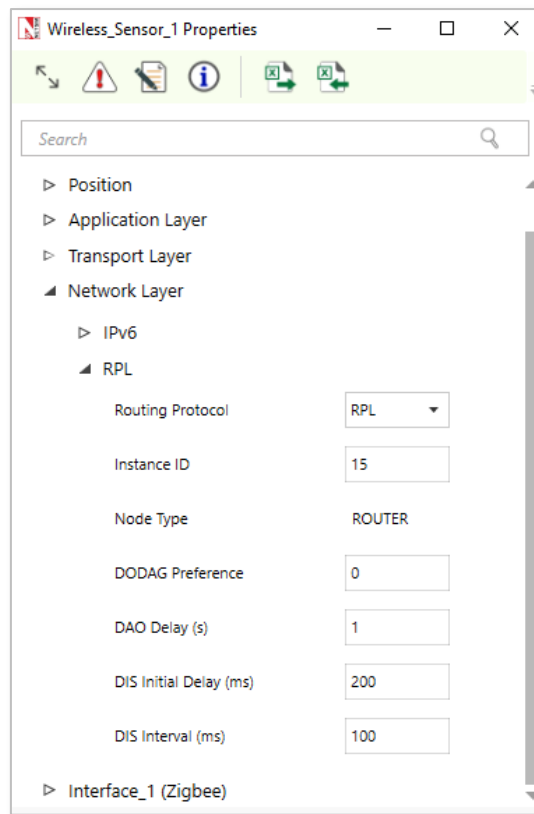


Figure 4-25: Routing Protocol to RPL in Network layer

3. In Adhoc Link Properties change Channel characteristics → Path Loss only, Path Loss Model → Log Distance and path loss exponent → 4.2.
4. Application properties has set as shown in below Table 4-3.

Application Properties			
Application ID	Application Type	Source Id	Destination Id
1	SENSOR_APP	1	11
2	SENSOR_APP	3	11
3	SENSOR_APP	8	11

Table 4-3: Application properties

5. In NetSim GUI Plots and Packet Traces are Enabled. Run simulation 100s and observe that all the sensors are sending data to route node in packet trace.

Result

Packet trace: The nodes send data messages to the root, creating an upward flow which can be observed in the packet trace. Once the simulation is completed, to view the packet trace file, click on “Open Packet Trace” option present in the left-hand-side of the Results Dashboard and filter the PACKET_TYPE to Sensing. you can observe the flow of Sensor Application packets as shown in Figure 4-26.

	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
131	2	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-8	SENSOR-7
132	2	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-1	SENSOR-2
133	2	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-7	SENSOR-5
134	2	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-2	SENSOR-6
135	2	0	Sensing	App2_SENSOR_APP	SENSOR-3	NODE-11	SENSOR-3	SENSOR-4
138	2	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-5	SINKNODE-9
139	2	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SINKNODE-9	ROUTER-10
140	2	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	ROUTER-10	NODE-11
164	3	0	Sensing	App2_SENSOR_APP	SENSOR-3	NODE-11	SENSOR-3	SENSOR-4
165	3	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-1	SENSOR-2
166	3	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-8	SENSOR-7
167	3	0	Sensing	App2_SENSOR_APP	SENSOR-3	NODE-11	SENSOR-4	SENSOR-5
168	3	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-2	SENSOR-6
179	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-1	SENSOR-2
180	4	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-8	SENSOR-7
181	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-2	SENSOR-6
182	4	0	Sensing	App2_SENSOR_APP	SENSOR-3	NODE-11	SENSOR-3	SENSOR-4
183	4	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-7	SENSOR-5
184	4	0	Sensing	App2_SENSOR_APP	SENSOR-3	NODE-11	SENSOR-4	SENSOR-5
186	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-6	SINKNODE-9
211	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-1	SENSOR-2
212	5	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-8	SENSOR-7
213	5	0	Sensing	App2_SENSOR_APP	SENSOR-3	NODE-11	SENSOR-3	SENSOR-4
214	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-2	SENSOR-6
215	5	0	Sensing	App3_SENSOR_APP	SENSOR-8	NODE-11	SENSOR-7	SENSOR-5
216	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SENSOR-6	SINKNODE-9
217	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	SINKNODE-9	ROUTER-10
218	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	NODE-11	ROUTER-10	NODE-11

Figure 4-26: Packet Trace for Multipoint to Point

4.1.3.2 Point to Multipoint

Settings done in sample network

1. Set the all the properties same as Multipoint to Point scenario
2. Application properties has set as shown in below Table 4-4.

Application Properties			
Application ID	Application Type	Source Id	Destination Id
1	SENSOR_APP	11	1
2	SENSOR_APP	11	3
3	SENSOR_APP	11	8

Table 4-4: Application properties

3. In NetSim GUI Plots are Enabled. Run simulation 100s and observe that all the sensors are sending data to route node in packet trace.

Result

Packet trace: Root sends data messages to the other nodes, producing a downward flow which can be observed in the packet trace. Once the simulation is completed, to view the packet trace file, click on “Open Packet Trace” option present in the left-hand-side of the Results Dashboard and filter the PACKET_TYPE to Sensing, you can observe the flow of Sensor Application packets as shown below:

	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
4		1	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	NODE-11	ROUTER-10
5		1	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	NODE-11	ROUTER-10
6		1	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	ROUTER-10	SINKNODE-9
7		1	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	NODE-11	ROUTER-10
8		1	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	ROUTER-10	SINKNODE-9
9		1	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	ROUTER-10	SINKNODE-9
192		2	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	NODE-11	ROUTER-10
193		2	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	NODE-11	ROUTER-10
194		2	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	ROUTER-10	SINKNODE-9
195		2	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	NODE-11	ROUTER-10
196		2	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	ROUTER-10	SINKNODE-9
197		2	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	ROUTER-10	SINKNODE-9
230		3	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	NODE-11	ROUTER-10
231		3	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	NODE-11	ROUTER-10
232		3	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	ROUTER-10	SINKNODE-9
233		3	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	NODE-11	ROUTER-10
234		3	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	ROUTER-10	SINKNODE-9
235		3	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	ROUTER-10	SINKNODE-9
252		4	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	NODE-11	ROUTER-10
253		4	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	NODE-11	ROUTER-10
254		4	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	ROUTER-10	SINKNODE-9
255		4	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	NODE-11	ROUTER-10
256		4	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	ROUTER-10	SINKNODE-9
257		4	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	ROUTER-10	SINKNODE-9
287		5	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	NODE-11	ROUTER-10
288		5	0 Sensing	App2_SENSOR_APP	NODE-11	SENSOR-3	NODE-11	ROUTER-10
289		5	0 Sensing	App1_SENSOR_APP	NODE-11	SENSOR-1	ROUTER-10	SINKNODE-9
290		5	0 Sensing	App3_SENSOR_APP	NODE-11	SENSOR-8	NODE-11	ROUTER-10

Figure 4-27: Packet trace for Point to Multipoint

4.1.3.3 Point to Point

Settings done in sample network

1. Set the all the properties same as Multipoint to Point scenario.
2. Application properties has set as shown in below Table 4-5.

Application Properties			
Application ID	Application Type	Source Id	Destination Id
1	SENSOR_APP	1	5

Table 4-5: Application properties

3. In NetSim GUI Plots are Enabled. Run simulation 100s and observe that all the sensors are sending data to route node in packet trace.

Result

Packet trace: Root sends data messages to the other nodes, producing a downward flow which can be observed in the packet trace. Once the simulation is completed, to view the packet trace file, click on “Open Packet Trace” option present in the left-hand-side of the Results Dashboard and filter the PACKET_TYPE to Sensing, you can observe the flow of Sensor Application packets as shown below:

	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
131	2	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
132	2	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
135	2	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
137	2	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SINKNODE-9	SENSOR-5
159	3	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
160	3	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
163	3	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
166	3	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SINKNODE-9	SENSOR-5
174	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
175	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
177	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
178	4	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SINKNODE-9	SENSOR-5
202	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
203	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
205	5	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
214	6	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
215	6	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
217	6	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
225	7	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
226	7	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
228	7	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
246	8	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
247	8	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
250	8	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9
251	8	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SINKNODE-9	SENSOR-5
265	9	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-1	SENSOR-2
266	9	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-2	SENSOR-6
268	9	0	Sensing	App1_SENSOR_APP	SENSOR-1	SENSOR-5	SENSOR-6	SINKNODE-9

Figure 4-28: Packet trace for Point to Point

4.1.3.4 Storing Mode

Settings done in sample network

1. Set the all the properties same as Multipoint to Point scenario.
2. Application properties has set as shown in below Table 4-7.

Application Properties			
Application ID	Application Type	Source Id	Destination Id
1	SENSOR_APP	3	8

Table 4-6: Application properties

3. In NetSim GUI Plots are Enabled. Run simulation 100s and observe that all the sensors are sending data to root node in packet trace.

Result

In storing without multicast MOP (MOP 2 (Point to point)), downward routes are also supported, but are different from MOP 1 (Point to multipoint); the nodes maintain, individually, a routing table constructed using DAO messages to provide downward traffic. Hence, downward forwarding occurs without the use of the root node. The results can be analyzed from the packet trace.

Packet trace: Once the simulation is completed, to view the packet trace file, click on “Open Packet Trace” option present in the left-hand-side of the Results Dashboard and filter the PACKET_TYPE to Sensing, you can observe the flow of Sensor Application packets as shown below.

1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
131	2	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
132	2	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
134	2	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SINKNODE-9
158	3	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
159	3	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
161	3	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SENSOR-7
163	3	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-7	SENSOR-8
172	4	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
173	4	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
175	4	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SENSOR-7
178	4	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-7	SENSOR-8
201	5	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
202	5	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
204	5	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SENSOR-7
209	5	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-7	SENSOR-8
214	6	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
215	6	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
217	6	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SENSOR-7
219	6	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-7	SENSOR-8
226	7	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
227	7	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
229	7	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SENSOR-7
232	7	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-7	SENSOR-8
249	8	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4
250	8	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-4	SENSOR-5
252	8	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-5	SENSOR-7
253	8	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-7	SENSOR-8
269	9	0	Sensing	App1_SENSOR_APP	SENSOR-3	SENSOR-8	SENSOR-3	SENSOR-4

Figure 4-29: Packet trace for RPL Storing Mode

4.2 WSN Example Simulation

4.2.1 CAP Time Analysis

Open NetSim, Select **Examples->IOT-WSN->Wireless Sensor Networks->CAP Time Analysis** then click on the tile in the middle panel to load the example as shown in Figure 4-30.

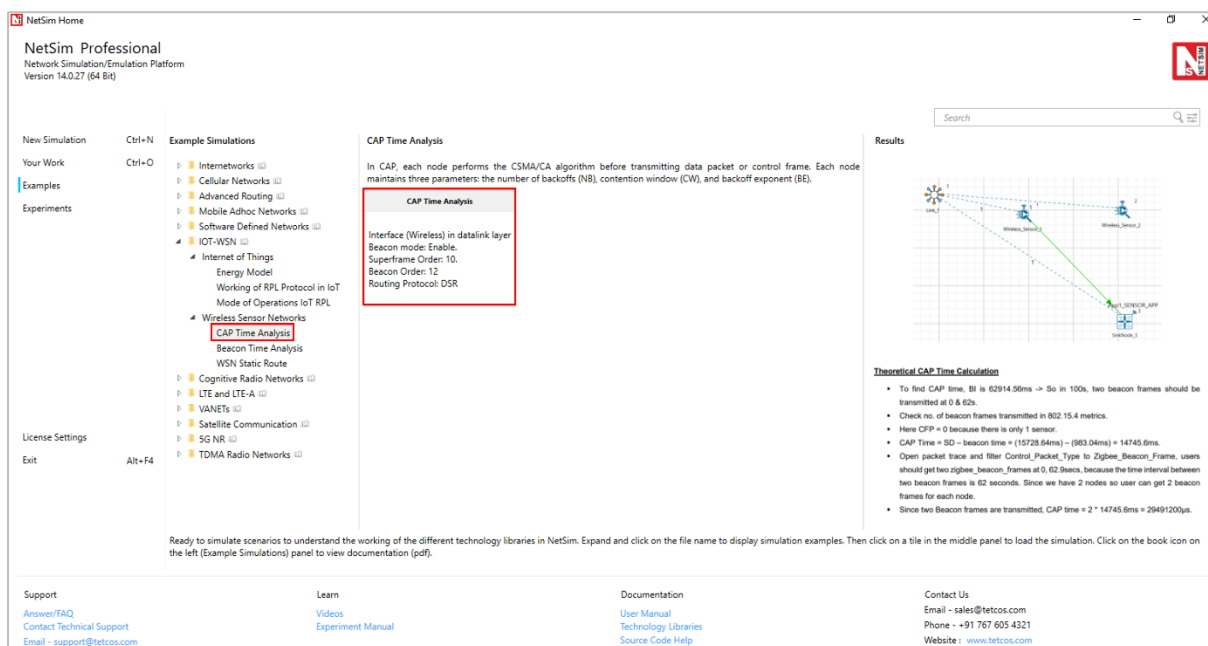


Figure 4-30: List of scenarios for the example of CAP Time Analysis

The following network diagram illustrates, what the NetSim UI displays when you open the example configuration file.

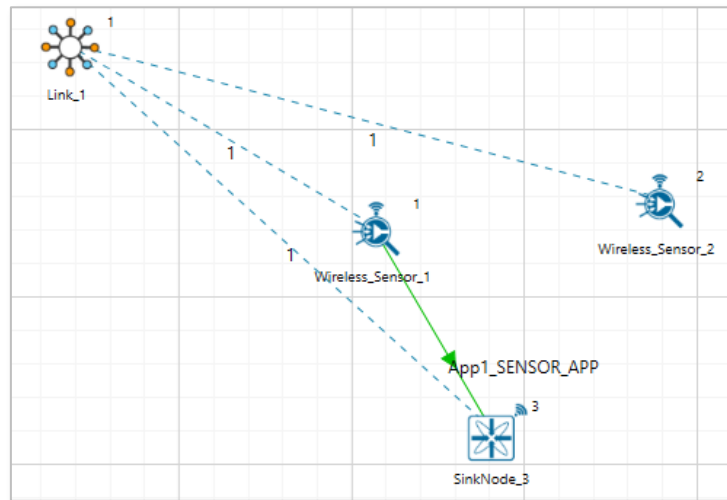


Figure 4-31: Network set up for studying the CAP Time Analysis

Settings done in example config file

1. The following Environment properties are already set, as Manually Via Click and Drop.
2. In SinkNode->INTERFACE_1(Zigbee)> Datalink Layer enable Beacon mode set Superframe Order (SO) -> 10 and Beacon Order (BO) -> 12

BeaconMode	Enable
SuperframeOrder	10
BeaconOrder	12

Figure 4-32: Datalink layer Properties window for Sink node

3. In Adhoc Link Properties change Channel characteristics -> Path Loss only, Path Loss Model -> Log Distance and path loss exponent -> 2.
4. Generate SENSOR_APP Traffic Between Wireless_Sensor_1 to WSN_Sink_3 and set the transport layer protocol as UDP other properties are default.
5. Enable Packet Trace and Plots.
6. Set Simulation time as 100 sec.

Theoretical CAP Time Calculation

- To find CAP time, BI is 62914.56ms -> So in 100s, two beacon frames should be transmitted at 0 & 62s.
- Check no. of beacon frames transmitted in 802.15.4 metrics.
- Here CFP = 0 because there is only 1 sensor.
- $CAP\ Time = SD - beacon\ time = (15728.64ms) - (983.04ms) = 14745.6ms.$

- Open packet trace and filter Control_Packet_Type to Zigbee_Beacon_Frame, users should get two zigbee_beacon_frames at 0, 62.9secs, because the time interval between two beacon frames is 62 seconds. Since we have 2 nodes so user can get 2 beacon frames for each node.
- Since two Beacon frames are transmitted,

$$CAP\ time = 2 \times 14745.6ms = 29491200\mu s.$$

NetSim Results:

- Check and compare the theoretical CAP time with NetSim simulation results in IEEE 802.15.4 metrics in Results Windows.
- CAP time = 29491200.0000Microsec.

CAPTime(Micro seconds)
0.0000
0.0000
29491200.0000

Figure 4-33: IEEE802.15.4_Metrics_Table in Results Windows

4.2.2 Beacon Time Analysis

Open NetSim, Select **Examples->IOT-WSN->Wireless Sensor Networks->Beacon Time Analysis** then click on the tile in the middle panel to load the example as shown in Figure 4-34.

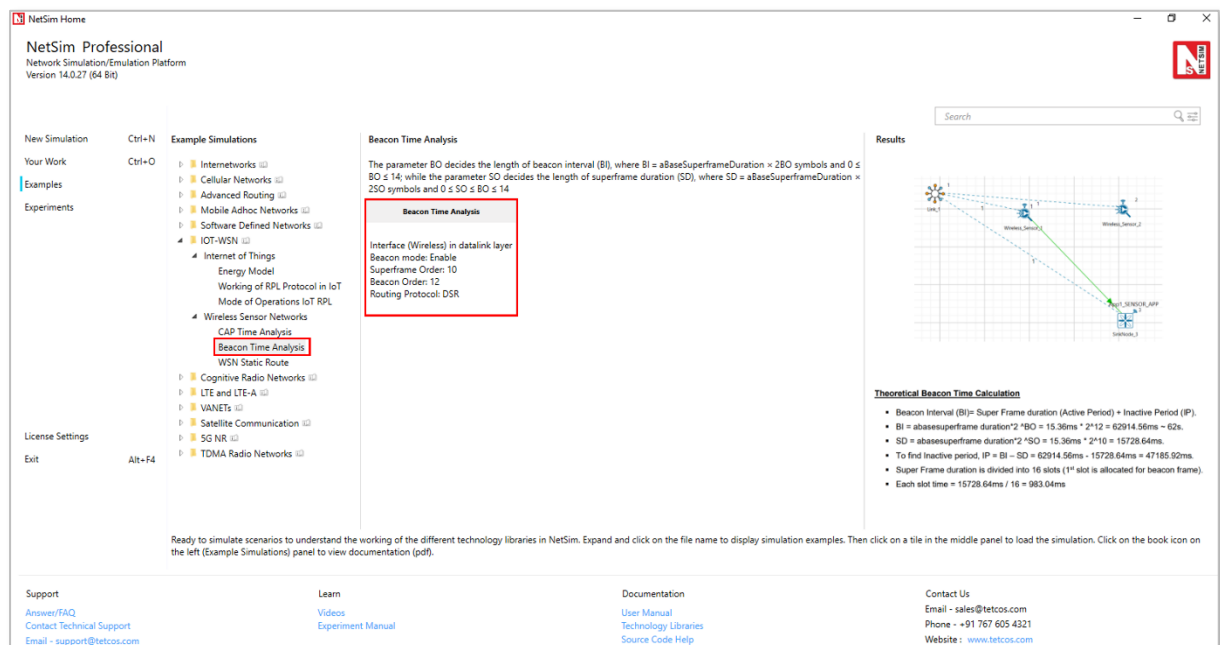


Figure 4-34: List of scenarios for the example of Beacon Time Analysis

The following network diagram illustrates, what the NetSim UI displays when you open the example configuration file.

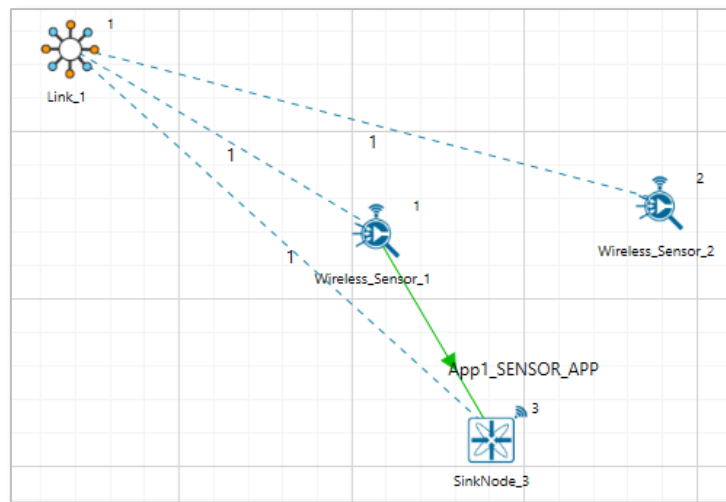


Figure 4-35: Network set up for studying the Beacon Time Analysis

Settings done in sample config file

1. The following Environment properties are already set, as Manually Via Click and Drop.
2. In SinkNode->INTERFACE_1(Zigbee)> Datalink Layer enable Beacon mode set Superframe Order (SO) -> 10 and Beacon Order (BO) -> 12

BeaconMode	Enable
SuperframeOrder	10
BeaconOrder	12

Figure 4-36: Datalink layer Properties window for Sink node

3. In Adhoc Link Properties change Channel characteristics-> Path Loss only, Path Loss Model -> Log Distance and path loss exponent ->2.
4. Generate SENSOR_APP Traffic Between Wireless_Sensor_1 to WSN_Sink_3 and set the transport layer protocol as UDP and other properties are default.
5. Enable Packet Trace and Plots.
6. Set Simulation time as 200 sec.

Theoretical Beacon Time Calculation

- Beacon Interval (BI)= Super Frame duration (Active Period) + Inactive Period (IP).
- $BI = abasesuperframe\ duration \times 2^{BO} = 15.36ms \times 2^{12} = 62914.56ms \sim 62s.$
- $SD = abasesuperframe\ duration \times 2^{SO} = 15.36ms \times 2^{10} = 15728.64ms.$
- To find Inactive period, $IP = BI - SD = 62914.56ms - 15728.64ms = 47185.92ms.$
- Super Frame duration is divided into 16 slots (1st slot is allocated for beacon frame).
- Each slot time = $\frac{15728.64ms}{16} = 983.04ms$

NetSim Results:

- Open packet trace and filter CONTROL_PACKET_TYPE to Zigbee_BEACON_FRAME, users should get four zigbee_beacon_frames at 0, 62.9, 125.8, 188.7 secs (approx.) for each Sensor_Node, because the time interval between two beacons frames is 62 seconds. Since we have 2 nodes so user can get 4 beacon frames for each node.

PACKET_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	MAC_LAYER_ARRIVAL_TIME(US)
0	Control_Packet	Zigbee_BEACON_FRAME	SINKNODE-3	Broadcast-0	SINKNODE-3	SENSOR-1	0
0	Control_Packet	Zigbee_BEACON_FRAME	SINKNODE-3	Broadcast-0	SINKNODE-3	SENSOR-1	62914560
0	Control_Packet	Zigbee_BEACON_FRAME	SINKNODE-3	Broadcast-0	SINKNODE-3	SENSOR-1	125829120
0	Control_Packet	Zigbee_BEACON_FRAME	SINKNODE-3	Broadcast-0	SINKNODE-3	SENSOR-1	188743680

Figure 4-37: Packet Trace Window of Beacon Time Analysis

- 4 beacons were transmitted, so $beacon\ time = 983.04ms \times 4 = 3932.16millisec$ (since 1 beacon = 1 time slot). Check the beacon time in IEEE 802.15.4 metrics window.

4.2.3 Static Routing in WSN

Open NetSim, Select **Examples->IOT-WSN->Wireless Sensor Networks->WSN Static Route** then click on the tile in the middle panel to load the example as shown in Figure 4-38.

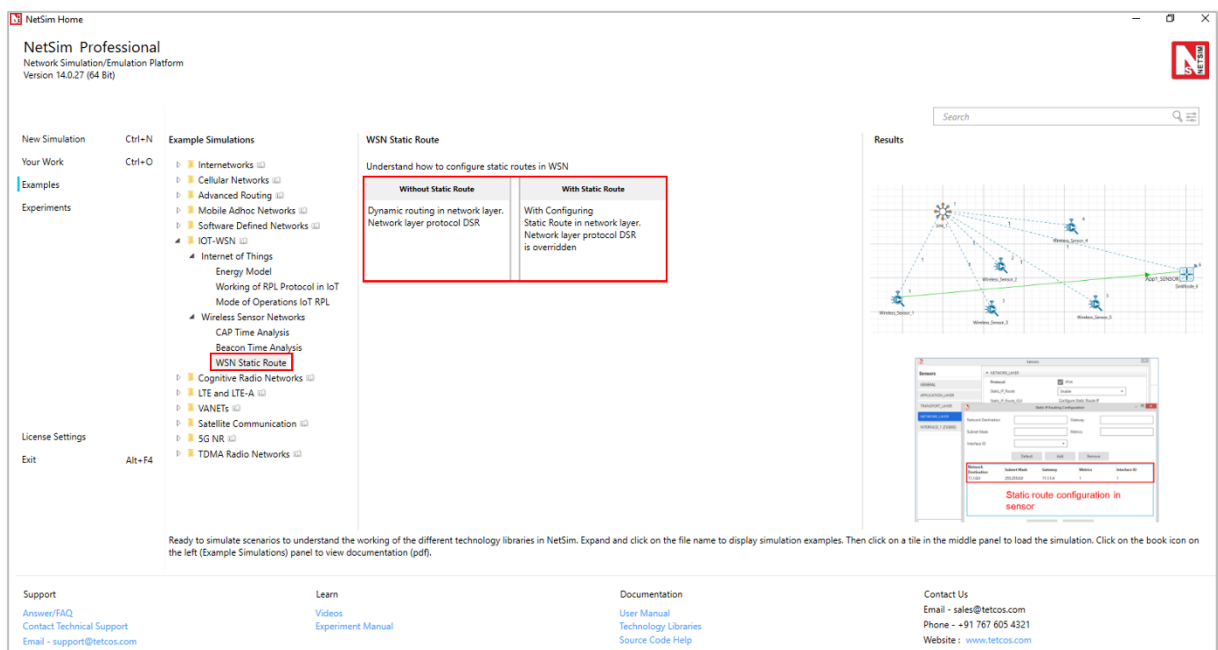


Figure 4-38: List of scenarios for the example of WSN Static Route

The following network diagram illustrates, what the NetSim UI displays when you open the example configuration file.

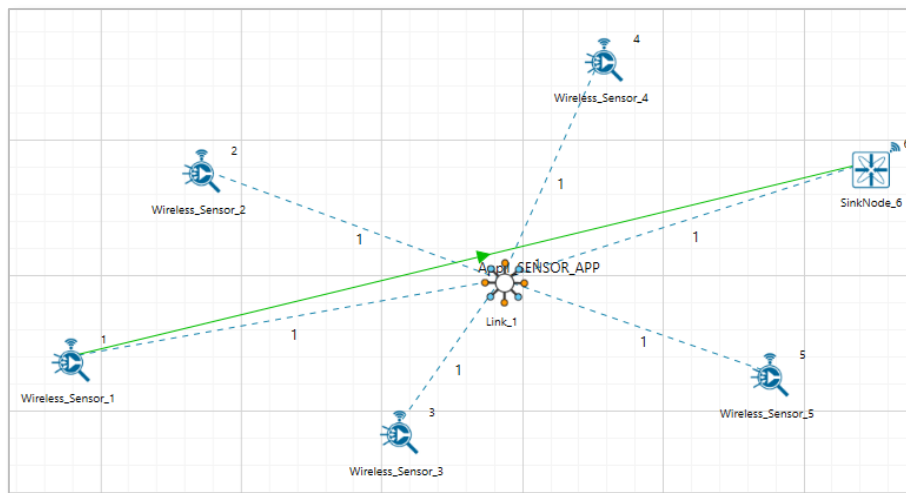


Figure 4-39: Network set up for studying the WSN Static Route

Without Static Route

Settings done in the network

- The following Environment properties are already set, as Manually Via Click and Drop.
- Set Application type as SENSOR_APP Source_Id as 1 and Destination_Id as 6
- Enable Packet trace and Plot option.
- Click on run simulation and set Simulation Time as 100 sec.

Results: Open packet trace and filter the PACKET TYPE to Sensing, observe SENSOR-1 would send packets directly to the destination SINKNODE-6.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
1	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SINKNODE-6
2	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SINKNODE-6
3	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SINKNODE-6
4	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SINKNODE-6
5	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SINKNODE-6
6	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SINKNODE-6

Figure 4-40: Packet Trace

With Static Route

Settings done in the network

- In **Without Static Route**, we have changed Wireless_Sensor properties as per the following.

Configuring Static Routes

Open Wireless_Sensor properties, go to network layer and Enable - **Static IP Route** ->click on **Configure Static Route IP** link, set **Network Destination**, **Gateway**, **Subnet Mask**, **Metrics**, and **Interface ID** as shown in below screenshot and click on **ADD**.

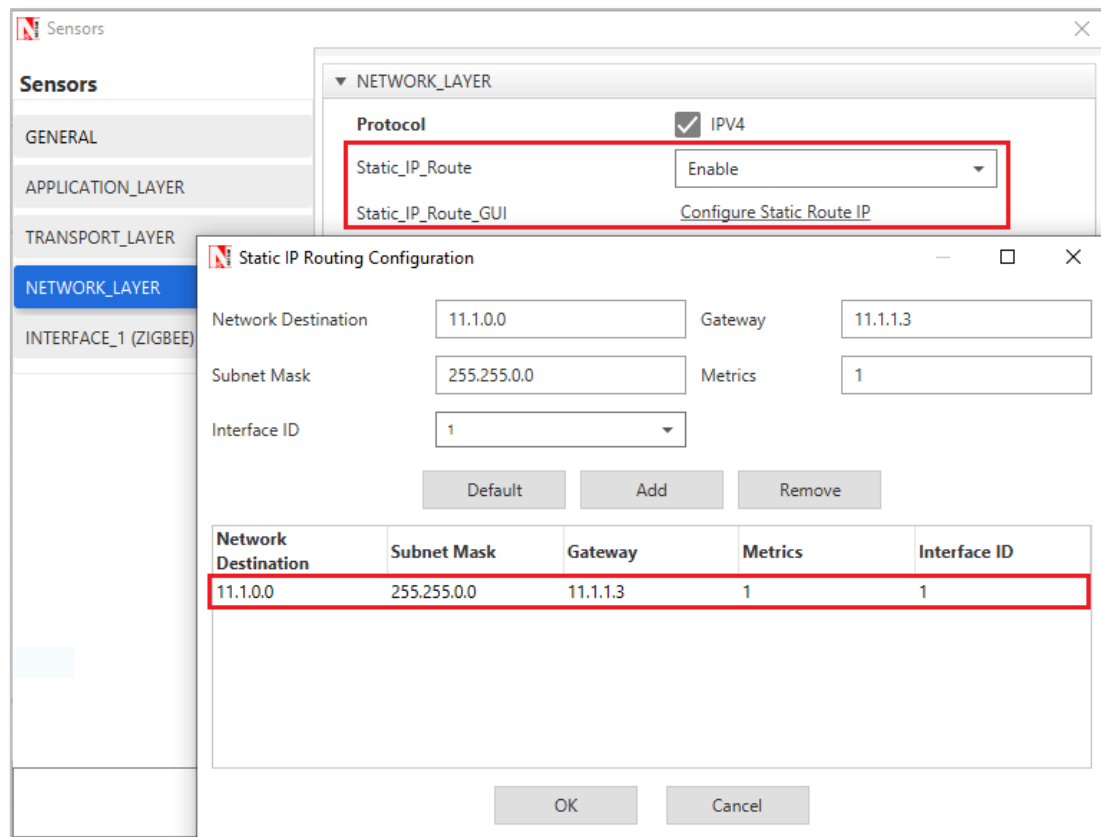


Figure 4-41: Static IP Routing Configuring window

Device	Network Destination	Gateway	Subnet Mask	Metrics	Interface ID
Wireless_Sensor_1	11.1.0.0	11.1.1.3	255.255.0.0	1	1
Wireless_Sensor_2	11.1.0.0	11.1.1.4	255.255.0.0	1	1
Wireless_Sensor_3	11.1.0.0	11.1.1.5	255.255.0.0	1	1
Wireless_Sensor_4	11.1.0.0	11.1.1.6	255.255.0.0	1	1

Table 4-7: Static Route Configuration for Sensors

- After setting the properties click on run simulation and set Simulation Time as 100 sec.

Results

Open packet trace and filter **PACKET_TYPE** column to **Sensing** and observe the packets flow as specified in the static route configuration.

SENSOR_1 → SENSOR_2 → SENSOR_3 → SENSOR_4 → SENSOR_5 → SINKNODE_6

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
1	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-1	SENSOR-2
1	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-2	SENSOR-3
1	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-3	SENSOR-4
1	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-4	SENSOR-5
1	0	Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-6	SENSOR-5	SINKNODE-6

Figure 4-42: Packet flow in Packet Trace

5 IOT-WSN Experiments in NetSim

Apart from examples, in-built experiments are also available in NetSim. Examples help the user understand the working of features in NetSim. Experiments are designed to help the user (usually students) learn networking concepts through simulation. The experiments contain objective, theory, set-up, results, and inference. The following experiments are available in the Experiments manual (pdf file).

1. One Hop IoT Network over IEEE 802.15.4
2. IoT – Multi-Hop Sensor-Sink Path
3. Performance Evaluation of a Star Topology IoT Network
4. Study the 802.15.4 Superframe Structure and analyze the effect of Superframe order on throughput.

6 Reference Documents

- [1] O. Landsiedel, K. Wehrle and S. Gotz, "Accurate Prediction of Power Consumption in Sensor Networks. University of Tübingen, Germany," 2005.
- [2] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur and R. Alexander, "IETF RFC 6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks".
- [3] "IEEE Standard 802.15.4-2011: Low-Rate Wireless Personal Area Networks (LR-WPANs)".

7 Latest FAQs

Up to date FAQs on NetSim's IoT/ WSN library is available at

<https://tetcos.freshdesk.com/support/solutions/folders/14000105117>