

# 1 IoT – Multi-Hop Sensor-Sink Path (Level 3)

**NOTE:** *It is recommended to carry out this experiment in Standard Version of NetSim.*

## 1.1 Introduction

The Internet provides the communication infrastructure for connecting computers, computing devices, and people. The Internet is itself an interconnection of a very large number of interconnected packet networks, all using the same packet networking protocol. The Internet of Things will be an extension of the Internet with sub-networks that will serve to connect “things” among themselves and with the larger Internet. For example, a farmer can deploy moisture sensors around the farm so that irrigation can be done only when necessary, thereby resulting in substantial water savings. Measurements from the sensors have to be communicated to a computer on the Internet, where inference and decision-making algorithms can advise the farmer as to required irrigation actions.

Farms could be very large, from a few acres to hundreds of acres. If the communication is entirely wireless, a moisture sensor might have to communicate with a sink that is 100s of meters away. As the distance between a transmitter and a receiver increase, the power of the signal received at the receiver decreases, eventually making it difficult for the signal processing algorithms at the receiver to decode the transmitted bits in the presence of the ever-present thermal noise. Also, for a large farm there would need to be many moisture sensors; many of them might transmit together, leading to collisions and interference.

## 1.2 Theory

The problem of increasing distance between the transmitter and the receiver is solved by placing packet routers between the sensors and the sink. There could even be multiple routers on the path from the sensor to the sink, the routers being placed so that any intermediate link is short enough to permit reliable communication (at the available power levels). We say that there is a multi-hop path from a sensor to the sink.

By introducing routers, we observe that we have a system with sensors, routers, and a sink; in general, there could be multiple sinks interconnected on a separate edge network. We note here that a sensor, on the path from another sensor to the sink, can also serve the role of a router. Nodes whose sole purpose is to forward packets might also need to be deployed.

The problem of collision and interference between multiple transmission is solved by overlaying the systems of sensors, routers, and sinks with a scheduler which determines (preferably in a distributed manner) which transmitters should transmit their packets to which of their receivers.

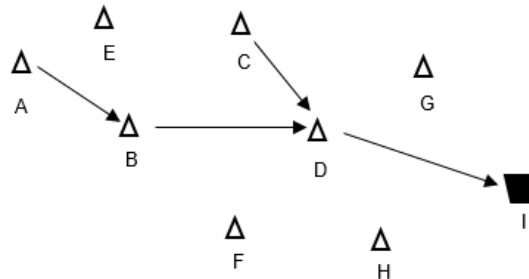


Figure 1-1: Data from Sensor A to Sink I takes the path A-B-D-I while data from sensor C to Sink I takes the path C-D-I

In this experiment, we will use NetSim Simulator to study the motivation for the introduction of packet routers, and to understand the performance issues that arise. We will understand the answers to questions such as:

1. How does packet error rate degrade as the sensor-sink distance increases?
2. How far can a sensor be from a sink before a router needs to be introduced?
3. A router will help to keep the signal-to-noise ratio at the desired levels, but is there any adverse implication of introducing a router?

### 1.3 Network Setup

Open NetSim and click on **Experiments> IOT-WSN> IoT Multi Hop Sensor Sink Path > Packet Delivery Rate and Distance** then click on the tile in the middle panel to load the example as shown in below Figure 1-2.

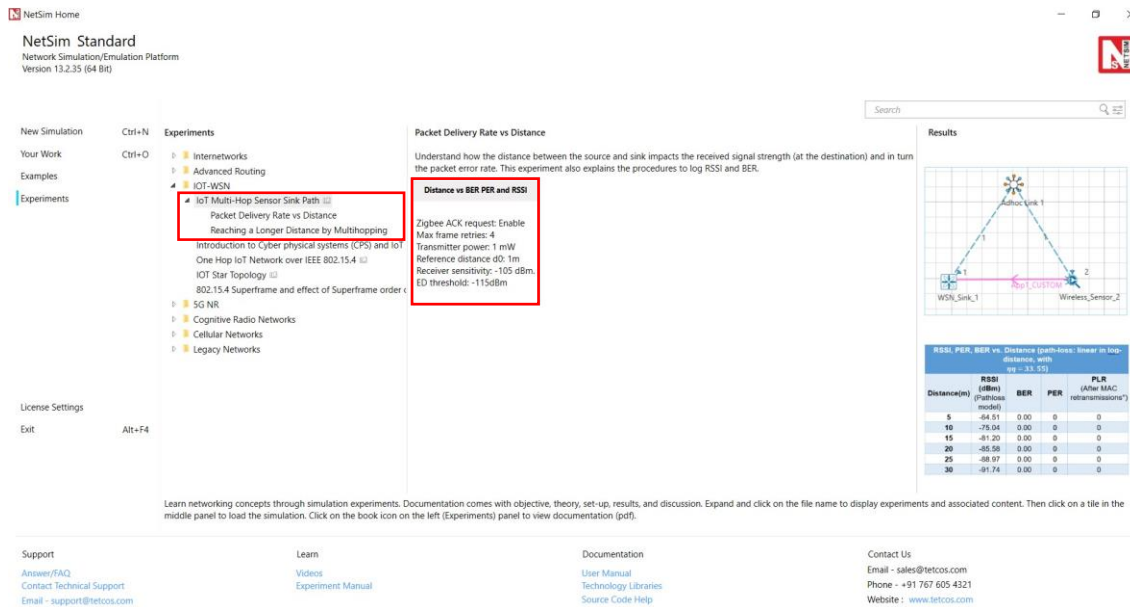


Figure 1-2: List of scenarios for the example of IoT Multi Hop Sensor Sink Path

## 1.4 Packet Delivery Rate vs. Distance

In this part, we perform a simulation to understand, **“How the distance between the source and sink impacts the received signal strength (at the destination) and in turn the packet error rate?”** We will assume a well-established path-loss model under which, as the distance varies, the received signal strength (in dBm) varies linearly. For a given transmit power (say 0dBm), at a certain reference distance (say 1m) the received power is  $c_0$ dBm and decreases beyond this point as  $-10\eta \log_{10} d$  for a transmitter-receiver distance of  $d$ . This is called a *power-law* path loss model, since in mW the power decreases as the  $\eta$  power of the distance  $d$ . The value of  $\eta$  is 2 for free space path loss and varies from 2 to 5 in the case of outdoor or indoor propagation. Values of  $\eta$  are obtained by carrying out experimental propagation studies.

### Distance vs BER PER and RSSI sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 1-3.

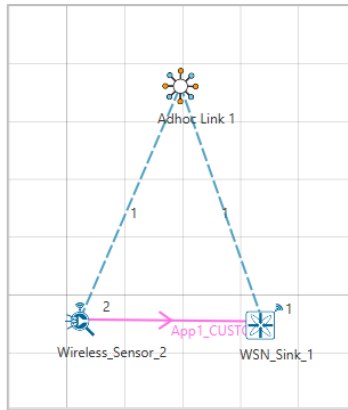


Figure 1-3: Network set up for studying the Distance vs BER PER and RSSI sample

### 1.4.1 Procedure

The following set of procedures were done to generate this sample:

**Step 1:** A network scenario is designed in the NetSim GUI comprising of a WSN Sink and 1 Wireless Sensor in **Wireless Sensor Networks**.

**NOTE:** NetSim currently supports a maximum of only one device as WSN Sink.

**Step 2:** Before we actually designed this network, in the Fast Config Window containing inputs for **Grid Settings and Sensor Placement**, the Grid Length and Side Length were set to 500 meters respectively, instead of the default 50 meters and we have chosen **Manually Via Click and Drop** option.

**Step 3:** The distance between the WSN Sink and Wireless Sensor is 5 meters.

**Step 4:** Go to Network Layer properties of Wireless Sensor 2, the Routing Protocol is set as **AODV**.

**NOTE:** The Routing Protocol parameter is Global. i.e., It will automatically be set to AODV in WSN Sink.

**Step 5:** In the Interface Zigbee > Data Link Layer of Wireless Sensor 2, **Ack Request** is set to Enable and **Max Frame Retries** is set to 4. Similarly, it is set for WSN Sink 1.

**Step 6:** In the Interface Zigbee > Physical Layer of Wireless Sensor 2, **Transmitter Power** is set to 1mW, **Reference Distance** is set to 1m, **Receiver Sensitivity** is set to -105dBm, and **ED Threshold** is set to -115dBm.

**Step 7:** Channel Characteristics: Path Loss Only, Path Loss Model: Log Distance, Path Loss Exponent: 3.5

**Step 8:** Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wireless Sensor 2 i.e., Source to WSN Sink 1 i.e., Destination with Transport Protocol set to UDP, Packet Size set to 70 Bytes and Inter Arrival Time set to 4000  $\mu$ s.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 140 Kbps. Generation Rate can be calculated using the formula:

$$\text{Generation Rate (Mbps)} = \text{Packet Size (Bytes)} * 8 / \text{Interarrival time } (\mu\text{s})$$

**Step 9:** The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2 Figure 1-4, Enable - Static IP Route - >Click on **Configure Static Route IP**.

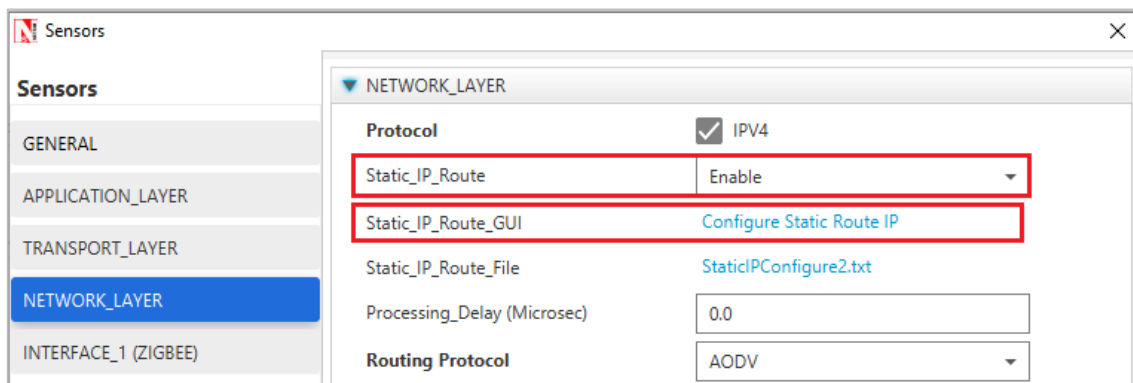


Figure 1-4: Network layer properties window

Static IP Routing Dialogue box gets open.

Enter the Network Destination, Gateway, Subnet Mask, Metrics, and Interface ID. Click on **Add**.

You will find the entry added to the below Static IP Routing Table as shown below.

Click on **OK**.

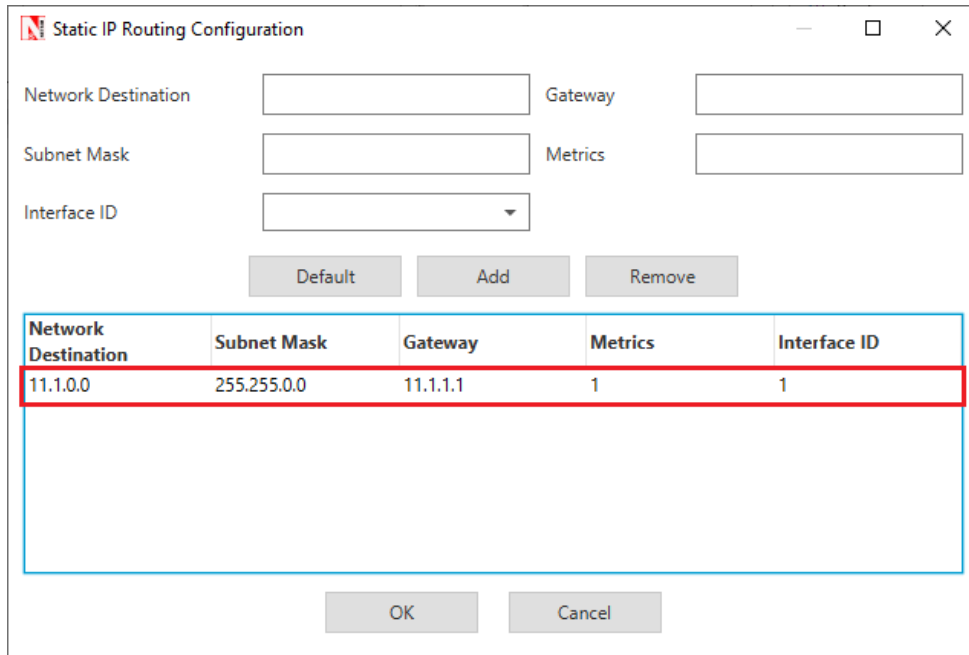


Figure 1-5: Static Route Configuration Window

**Step 9:** Logs can be enabled by clicking on the log icon in the toolbar. Select the IEEE802.15.4 Radio Measurements log and click on OK.

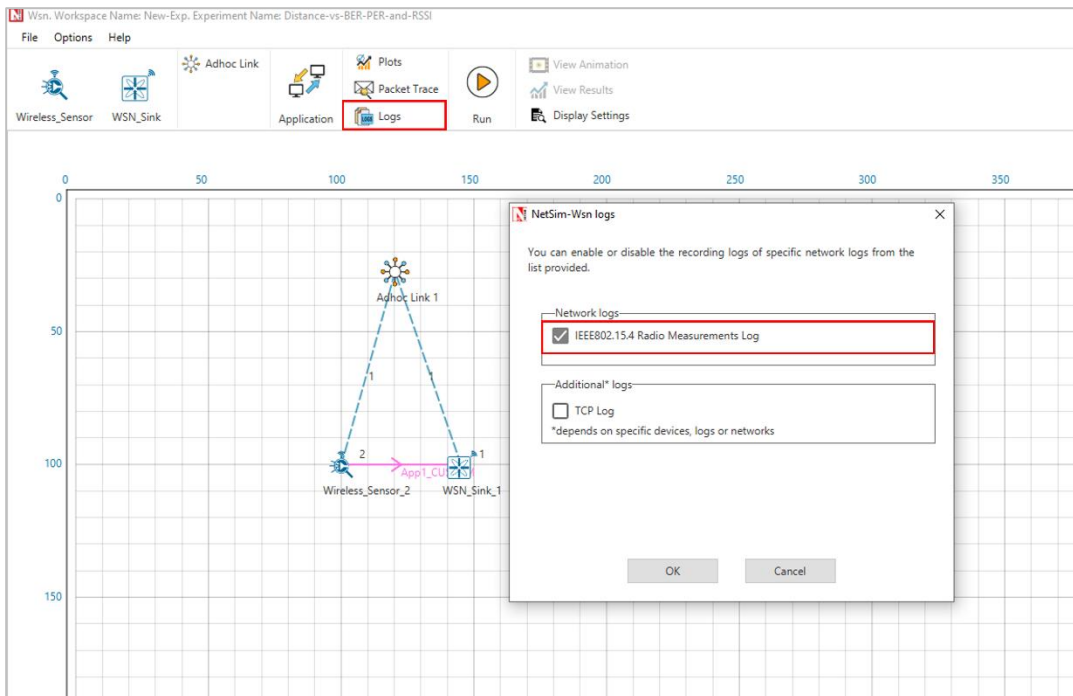


Figure 1-6: Enabling log files in NetSim GUI.

**Step 10:** Packet Trace is enabled in NetSim GUI. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis.

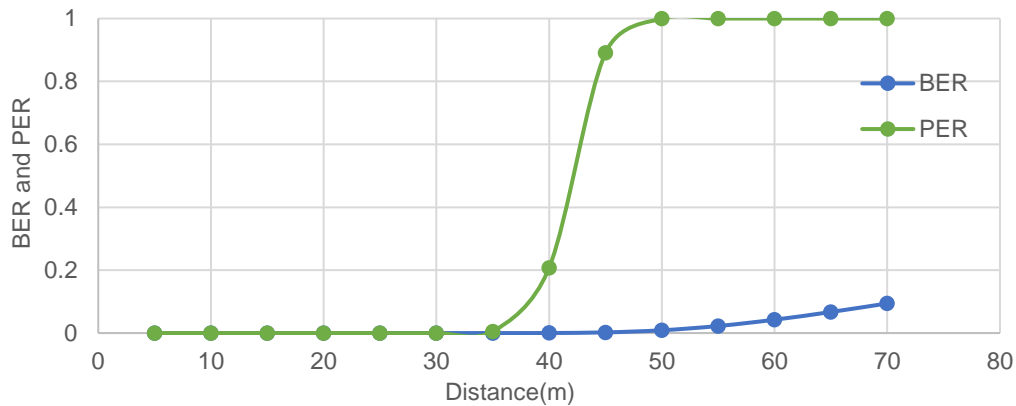
**Step 10:** Enable the plots and run the simulation for 10 Seconds. Once the simulation is complete, Go to metrics window expand Log Files option and open \_Radio\_Measurements.log file.

Output for Distance vs BER PER and RSSI sample

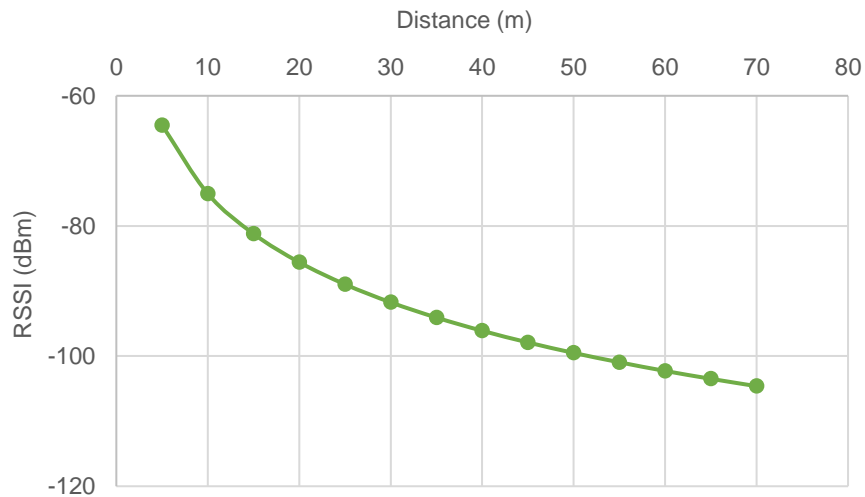
RSSI, PER, BER vs. Distance (path-loss: linear in log-distance, with $\eta = 3.5$ )				
Distance(m)	RSSI (dBm) (Pathloss model)	BER	PER	PLR (After MAC retransmissions*)
5	-64.51	0.00	0	0
10	-75.04	0.00	0	0
15	-81.20	0.00	0	0
20	-85.58	0.00	0	0
25	-88.97	0.00	0	0
30	-91.74	0.00	0	0
35	-94.08	0.000005	0.0051	0
40	-96.11	0.000229	0.2076	0
45	-97.90	0.002175	0.8905	0.447
50	-99.51	0.008861	0.9999	1
55	-100.95	0.022370	1	1
60	-102.28	0.042390	1	1
65	-103.49	0.067026	1	1
70	-104.62	0.094075	1	1
75	-	-	-	-
80	-	-	-	-

Table 1-1: RSSI, PER, BER from ZIGBEE\_BER\_LOG.txt vs. Distance

## Comparison Charts



(a)



(b)

Figure 1-7: (a) Distance Vs. BER, PER and (b) Distance vs. RSSI

\* The IEEE 802.15.4 MAC implements a retransmission scheme that attempts to recover errored packets by retransmission. If all the retransmission attempts are also errored, the packet is lost.

The table above reports the RSSI (Received Signal Strength), BER (Bit Error Rate), and Packet Error Rate (PER), and the Packet Loss Rate (PLR) as the distance between the sensor to the sink is increased from 5m to 50m with path loss exponent  $\eta = 3.5$ . We see that the BER is 0 until a received power of about -92dBm. At a distance of 35m the received power is -94 dBm, and we notice a small BER of  $5 \times 10^{-6}$ . As the distance is increased further the BER continues to grow and at 45m the BER is about 0.002175, yielding  $PER = 0.89$ , and  $PLR = 0.44$ . Here



$PER$  is obtained from the following formula (which assumes independent bit errors across a packet)

$$PER = 1 - (1 - BER)^{PL},$$

Where,

$PL$  – packet length in bits at the PHY layer

$$PL \text{ (bits)} = (70 \text{ (payload)} + 39 \text{ (overhead)}) * 8$$

The  $PLR$  in the above table has been obtained from NetSim, which implements the details of the IEEE 802.15.4 MAC acknowledgement and reattempt mechanism. This mechanism is complex, involving a MAC acknowledgement, time-outs, and multiple reattempts. Analysis of the  $PLR$ , therefore, is not straightforward. Assuming that the probability of MAC acknowledgement error is small (since it is a small packet), the  $PLR$  can be approximated as  $PER^{K+1}$ , where  $K$  is the maximum number of times a packet can be retransmitted.

$$PLR = \frac{\text{Total number of Lost Packet}}{\text{Total number of Packet Sent by Source MAC}}$$

*Total number of Lost packets*

*= Total number of Packet Sent by SourceMAC*

*– Packets Received at Destination MAC*

### Steps to calculate Packet Loss Rate

- Open Packet Trace from the Results Dashboard. Filter the PACKET TYPE column as Custom and note down the packet id of the last packet sent from the PACKET ID column.

451	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
452	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
452	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
452	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
452	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
452	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
453	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
454	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
454	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
454	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
454	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
454	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
455	0 Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1

Figure 1-8: Packet Trace

This represents the total number of packets sent by the source.

- Note down the Packets Received from the Application Metrics in the Results Dashboard Figure 1-9.

Application_Metrics_Table							
Application_Metrics							
Application ID	Throughput Plot	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (microsecond)	Jitter (microsecond)
1	<a href="#">Application Throughput plot</a>	App1_CUSTOM	2500	254	0.014224	4133764.897638	31947.185771

Figure 1-9: Application metrics table in Result Dashboard

This represents the total number of packets received at the destination.

- Calculate the total number of Lost Packets and PLR as follows:

For the above case,

$$\text{Total number of Packet Sent by SourceMAC} = 455$$

$$\text{Packets Received at Destination MAC} = 254$$

$$\text{Total number of Lost packets} = 455 - 254 = 201$$

$$PLR = \frac{201}{455} = 0.441$$

## 1.5 Inference

It is clear that Internet applications, such as banking and reliable file transfer, require that all the transmitted data is received with 100% accuracy. The Internet achieves this, in spite of unreliable communication media (no medium is 100% reliable) by various protocols above the network layer. Many IoT applications, however, can work with less than 100% packet delivery without affecting the application. Take, for example, the farm moisture sensing application mentioned in the introduction. The moisture levels vary slowly; if one measurement is lost, the next received measurement might suffice for the decision-making algorithm. This sort of thinking also permits the IoT applications to utilize cheap, low power devices, making the IoT concept practical and affordable.

With the above discussion in mind, let us say that the application under consideration requires a measurement delivery rate of at least 80%. Examining the table above, we conclude that the sensor-sink distance must not be more than 40 meters. Thus, even a 1 acre farm ( $61m \times 61m$ ) would require multi-hopping to connect sensors to a sink at the edge of the farm.

In Part 2 of this experiment, we will study the placement of a single router between the sensor and the sink, to increase the sensor-sink distance beyond 40 meters.

## 1.6 Reaching a Longer Distance by Multihopping

### Direct sensor sink link sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 1-10.

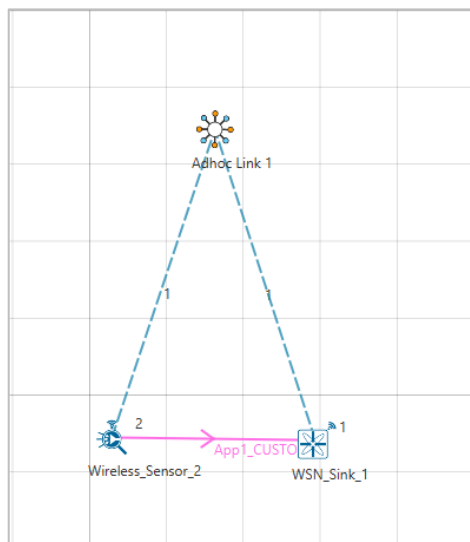


Figure 1-10: Network set up for studying the Direct sensor sink link sample

### 1.7.1 Procedure

The following changes in settings are done from the previous sample:

**Step 1:** The distance between the WSN Sink and Wireless Sensor is 40 meters.

**Step 2:** In the Interface Zigbee > Data Link Layer of Wireless Sensor 2, **Ack Request** is set to Enable and **Max Frame Retries** is set to 3.

**Step 3:** The Ad hoc Link properties are set as follows Figure 1-11.

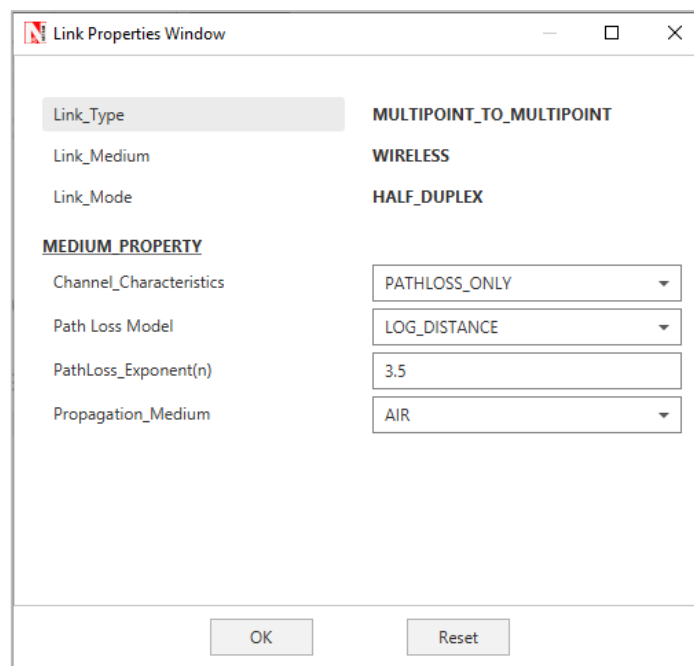


Figure 1-11: Wireless Link properties

**Step 4:** Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wireless Sensor 2 i.e. Source to WSN Sink 1 i.e. Destination with Packet Size set to 70 Bytes and Inter Arrival Time set to 100000  $\mu$ s.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 5.6 Kbps. Generation Rate can be calculated using the formula:

$$\text{Generation Rate (Mbps)} = \text{Packet Size (Bytes)} * 8 / \text{Interarrival time } (\mu\text{s})$$

**Step 5:** Enable the plots and run the Simulation for 100 Seconds. Once the simulation is complete, note down the Packet Generated value and Throughput value from the **Application Metrics**.

Note down the Packet Received, Packet Errored, and Packet Collided from the **Link Metrics**.

### Router between sensor and sink sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 1-12.

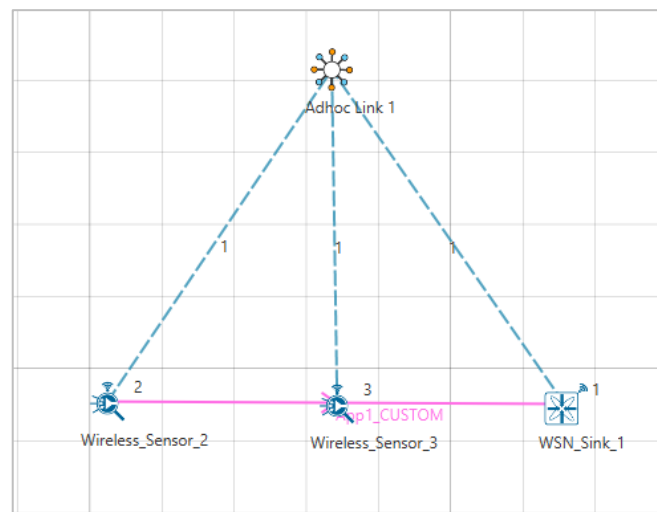


Figure 1-12: Network set up for studying the Router between sensor and sink sample

## 1.7.2 Procedure

The following changes in settings are done from the previous sample:

**Step 1:** One more Wireless Sensor is added to this network. The distance between Wireless Sensor 2 and Wireless Sensor 3 is 40 meters and the distance between Wireless Sensor 3 and the WSN Sink is 40 meters.

**Step 2:** The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2 Figure 1-13, Enable - Static IP Route - >Click on **Configure Static Route IP**.

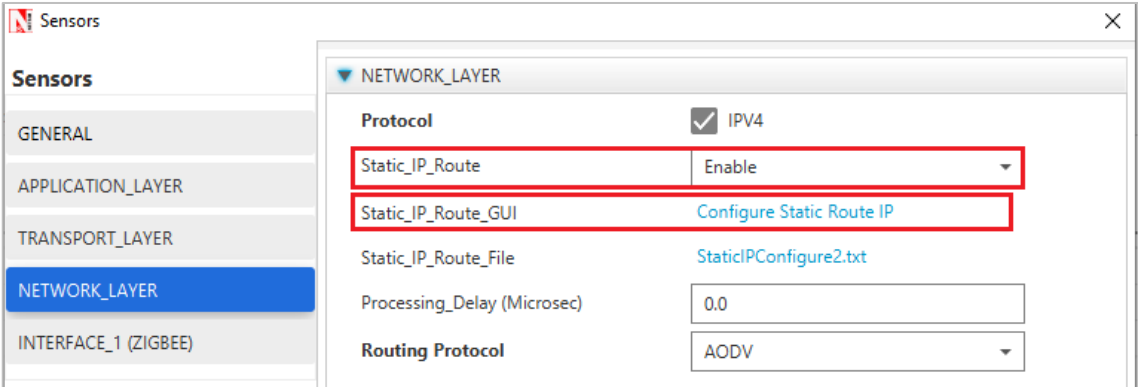


Figure 1-13: Network layer properties window

Static IP Routing Dialogue box gets open.

Enter the Network Destination, Gateway, Subnet Mask, Metrics, and Interface ID. Click on **Add**.

You will find the entry added to the below Static IP Routing Table as shown below:

Click on **OK**.

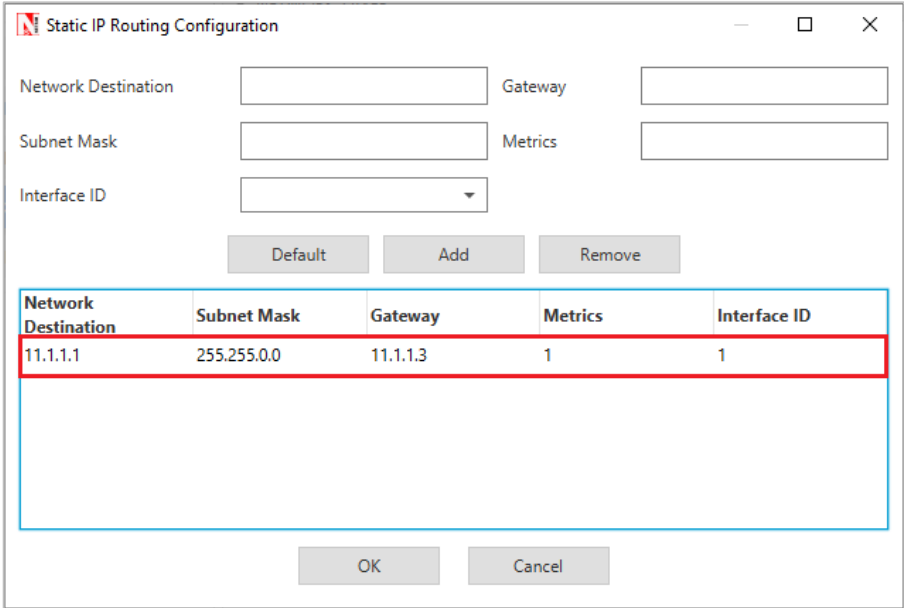


Figure 1-14: Static Route configuration for Wireless Sensor 2

Similarly, Static IP is set for Wireless Sensor 3 as shown below Figure 1-15.

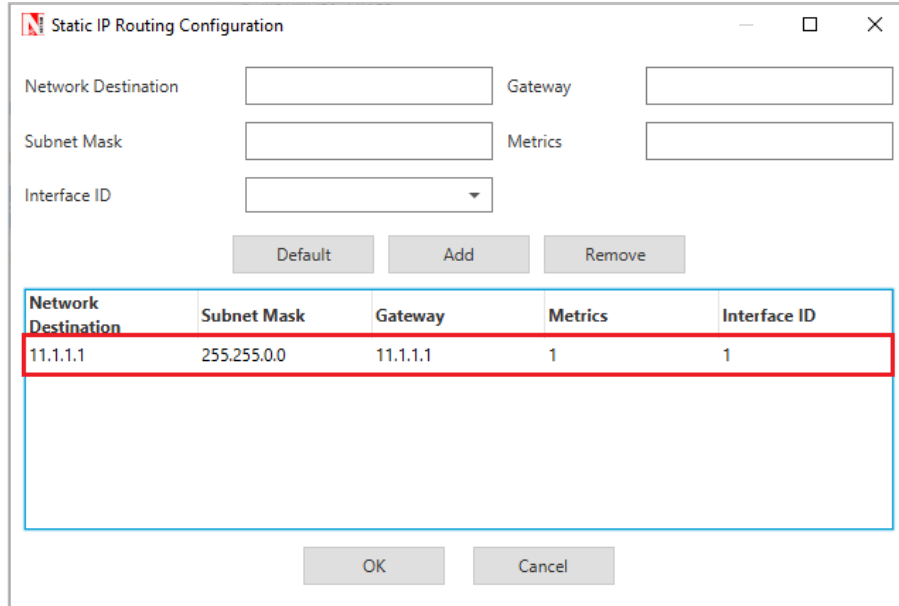


Figure 1-15: Static Route configuration for Wireless Sensor 3

**Step 3:** Enable the plots and run the Simulation for 100 Seconds. Once the simulation is complete, note down the Packet Generated value and Throughput value from the **Application Metrics**.

Note down the Packet Received, Packet Errored, and Packet Collided from the Link Metrics Table 1-2.

### 1.7 Output for Router between sensor and sink sample

	Source-Sink Distance (m)	Packets Generated	Packets Received	Packet s Errored (PHY)	Packet s Collided	Packet Loss (MAC)	P L R	Mean Delay ( $\mu s$ )
<b>Direct sensor-sink link</b>	40	1000	1012	244	0	0	0	6514.45
<b>Router between sensor and sink</b>	80 (router at midpoint)	1000	1015	540	0	0	0	14239.94

Table 1-2: Packet Generated/Received/Errored/Collided and Mean delay from result dashboard

**NOTE:** Packet loss (PHY) is the number of packets that were received in error and then recovered by retransmission. Packets received is slightly higher than packets generated on account of retransmissions of successful packets in case of ACK errors.

## 1.8 Inference

In **Distance vs BER PER and RSSI** sample of this experiment, we learnt that if the sensor device uses a transmit power of 0dBm, then for one-hop communication to the sink, the sensor-sink distance cannot exceed 40m. If the sensor-sink distance needs to exceed 40m (see the example discussed earlier), there are two options:

1. The transmit power can be increased. There is, however, a maximum transmit power for a given device. Wireless transceivers based on the CC 2420 have a maximum power of 0dBm (i.e., about 1 mW), whereas the CC 2520 IEEE 802.15.4 transceiver provides maximum transmit power of 5dBm (i.e., about 3 mW). Thus, given that there is always a maximum transmit power, there will always be a limit on the maximum sensor-sink distance.
2. Routers can be introduced between the sensor and the sink, so that packets from the sensor to the sink follow a *multihop* path. A router is a device that will have the same transceiver as a sensor, but its microcontroller will run a program that will allow it to forward packets that it receives. Note that a sensor device can also be programmed to serve as a router. Thus, in IOT networks, sensor devices themselves serve as routers.

In this part of the experiment, we study the option of introducing a router between a sensor and the sink to increase the sensor-sink distance. We will compare the performance of two networks, one with the sensor communicating with a sink at the distance of 40m, and another with the sensor-sink distance being 80m, with a sensor at the mid-point between the sensor and the sink.

**Direct sensor sink link sample** simulates a one hop network with a sensor-sink distance of 40m. We recall from Part 1 that, with the transceiver model implemented in NetSim, 40m is the longest one hop distance possible for 100% packet delivery rate. In **Router between sensor and sink** sample, to study the usefulness of routing we will set up network with a sensor-sink distance of 80m with a packet router at the midpoint between the sensor and the sink.

The measurement process at the sensor is such that one measurement (i.e., one packet) is generated every 100ms. The objective is to deliver these measurements to the sink with 100% delivery probability. From Part 1 of this experiment, we know that a single hop of 80m will not provide the desired packet delivery performance.



The Table at the beginning of this section shows the results. We see that both networks are able to provide a packet delivery probability of 100%. It is clear, however, that since the second network has two hops, each packet needs to be transmitted twice, hence the mean delay between a measurement being generated and it being received at the sink is doubled. Thus, the longer sensor-sink distance is being achieved, for the same delivery rate, at an increased delivery delay.

The following points may be noted from the table:

1. The number of packets lost due to PHY errors. The packet delivery rate is 100% despite these losses since the MAC layer re-transmission mechanism is able to recover all lost packets.
2. There are no collisions. Since both the links (sensor-router and router-sink) use the same channel and there is no co-ordination between them, it is possible, in general for sensor-router and router-sink transmissions to collide. This is probable when the measurement rate is large, leading to simultaneously nonempty queues at the sensor and router. In this experiment we kept the measurement rate small such that the sensor queue is empty when the router is transmitting and vice versa. This avoids any collisions.