21 IoT – Multi-Hop Sensor-Sink Path

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

21.1 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 in 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 increases, 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 a large number of moisture sensors; many of them might transmit together, leading to *collisions* and *interference.*

21.2 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.

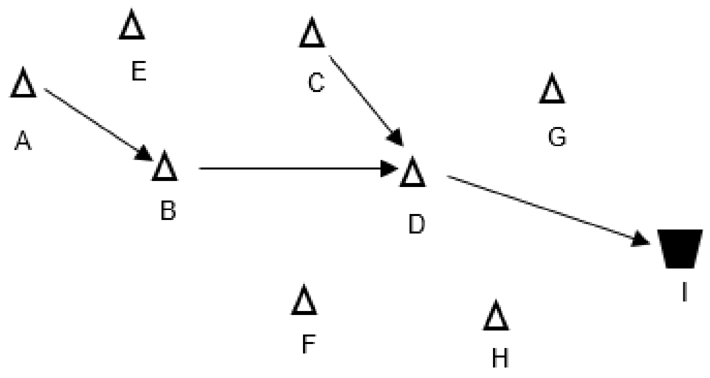


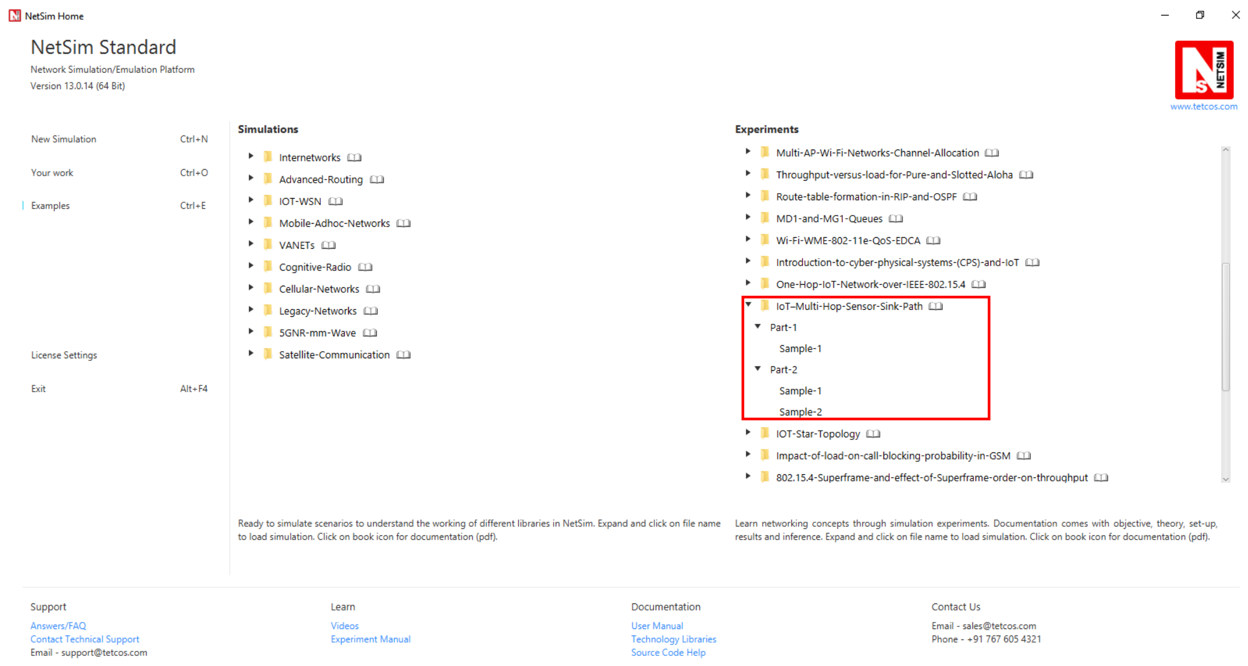
Figure 21-1: Data from Sensor A to Sink I takes the path A-B-D-I while data from sensor C to Sink I takes the path C-D-I

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?

21.3 Network Setup

Open NetSim and click **Examples > Experiments > IoT–Multi-Hop-Sensor-Sink-Path > Part-1 > Sample-1** as shown below **Figure 21-2**.


NetSim Home

NetSim Standard
Network Simulation/Emulation Platform
Version 13.0.14 (64 Bit)

www.tetcos.com

New Simulation	Ctrl+N
Your work	Ctrl+O
Examples	Ctrl+E
License Settings	
Exit	Alt+F4

Simulations
- Internetworks
- Advanced-Routing
- IOT-WSN
- Mobile-Adhoc-Networks
- VANETs
- Cognitive-Radio
- Cellular-Networks
- Legacy-Networks
- 5GNR-mm-Wave
- Satellite-Communication

Ready to simulate scenarios to understand the working of different libraries in NetSim. Expand and click on file name to load simulation. Click on book icon for documentation (pdf).

Experiments
- Multi-AP-Wi-Fi-Networks-Channel-Allocation
- Throughput-versus-load-for-Pure-and-Slotted-Aloha
- Route-table-formation-in-RIP-and-OSPF
- MD1-and-MG1-Queues
- Wi-Fi-WME-802-11e-QoS-EDCA
- Introduction-to-cyber-physical-systems-(CPS)-and-IoT
- One-Hop-IoT-Network-over-IEEE-802.15.4
- IoT–Multi-Hop-Sensor-Sink-Path
  - Part-1
    - Sample-1
  - Part-2
    - Sample-1
    - Sample-2
- IOT-Star-Topology
- Impact-of-load-on-call-blocking-probability-in-GSM
- 802.15.4-Superframe-and-effect-of-Superframe-order-on-throughput

Learn networking concepts through simulation experiments. Documentation comes with objective, theory, set-up, results and inference. Expand and click on file name to load simulation. Click on book icon for documentation (pdf).

Support	Learn	Documentation	Contact Us
Answers/FAQ	Videos	User Manual	Email - sales@tetcos.com
Contact Technical Support	Experiment Manual	Technology Libraries	Phone - +91 767 605 4321
Email - support@tetcos.com		Source Code Help	

Figure 21-2: Experiments List

21.4 Part 1 – 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_0dBm 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 η power of the distance d. The value of η is 2 for free space path loss and varies from 2 to 5 in the case of outdoor or indoor propagation. Values of η are obtained by carrying out experimental propagation studies.

Sample 1

NetSim UI displays the configuration file corresponding to this experiment as shown below **Figure 21-3**.

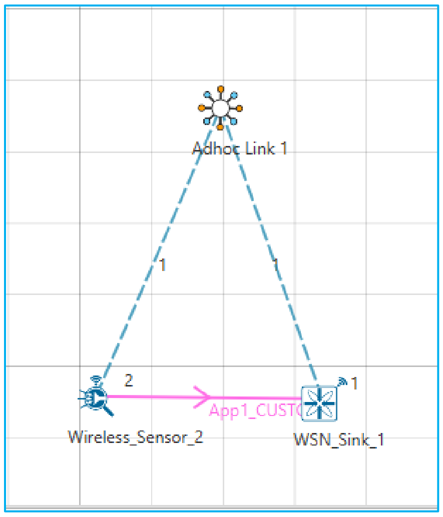


Figure 21-3: Network topology for Sample 1

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

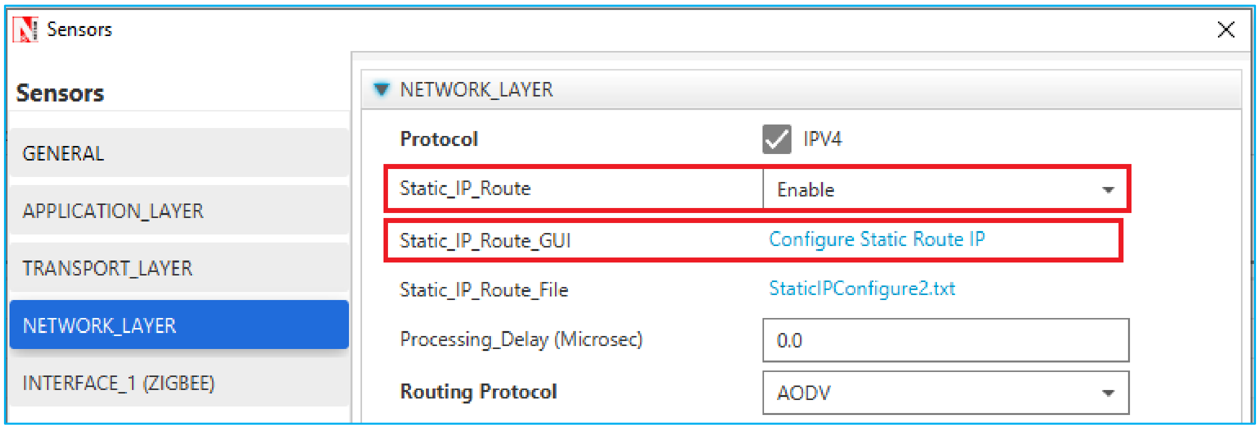
A CUSTOM Application is generated from Wireless Sensor 2 i.e. Source to WSN Sink 1 i.e., Destination with Transport Protocol set to UDP, Packet Size set to 70 Bytes and Inter Arrival Time set to 4000 µs.

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:

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

Step 8: The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2 **Figure 21-4,** Enable - Static IP Route ->Click on **Configure Static Route IP**.



Protocol ✓ IPV4
Static_IP_Route Enable
Static_IP_Route_GUI Configure Static Route IP
Static_IP_Route_File StaticIPConfigure2.txt
Processing_Delay (Microsec) 0.0
Routing Protocol AODV

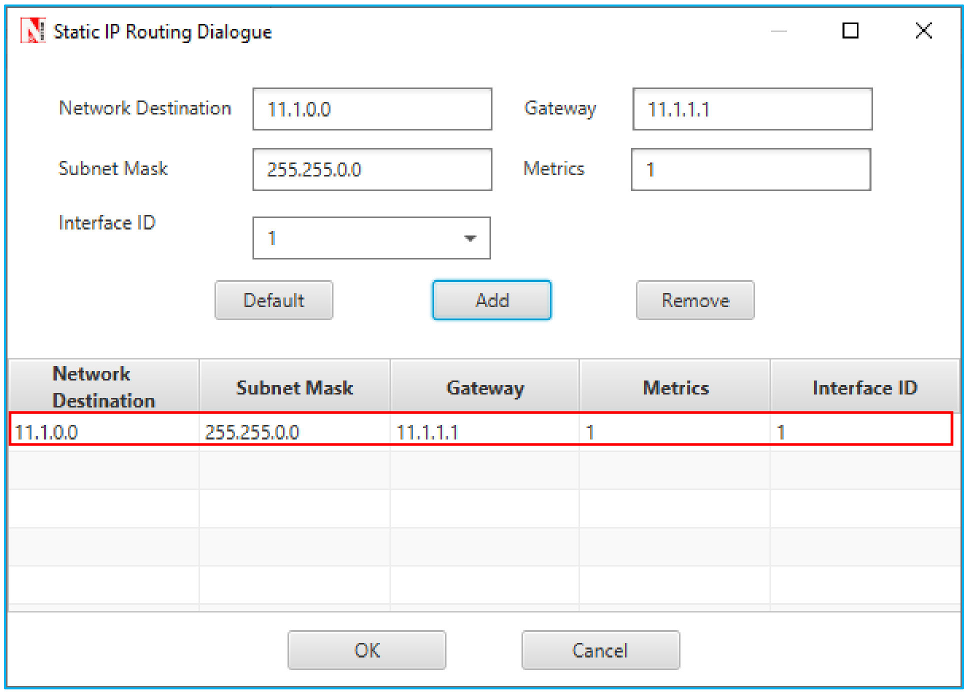
Figure 21-4: Network layer properties window

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**.



Network Destination: 11.1.0.0
Gateway: 11.1.1.1
Subnet Mask: 255.255.0.0
Metrics: 1
Interface ID: 1

Network Destination	Subnet Mask	Gateway	Metrics	Interface ID
11.1.0.0	255.255.0.0	11.1.1.1	1	1

Figure 21-5: Static Route Configuration Window

Step 9: 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. Plots are enable in NetSim GUI.

Note: *Before we click on* ***Run*** *simulation, user need to modify the code as per the* ***"Procedure to log RSSI and BER"*** *given below.*

NOTE: The following changes need to be done manually by the user inorder to carry out this experiment.

Procedure to log RSSI and BER (Possible in Standard / Pro Versions only):

RSSI and BER in ZigBee project can be logged into a text file. The following code changes are required to log these parameters into a txt 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 **Win32** or **x64** according to the NetSim build which you are using.

NOTE: We recommend Visual Studio Community Edition 2017 or Higher.

- Go to the Zigbee Project in the Solution Explorer. Open 802_15_4.c file and add the follwing lines of code highlighted in red, inside the **fn_NetSim_Zigbee_init()** function as shown below:

```
_declspec (dllexport) int fn_NetSim_Zigbee_Init()
{
FILE* fp;
//RSSI BER SNR LOG
fp = fopen("ZIGBEE_BER_LOG.txt", "w+");
if (fp)
{
fprintf(fp,"PACKET_ID,\tTRANSMITTER,\t\tRECEIVER,\tRX_POWER(dBm),\t
TOTAL_RX_POWER(dBm), \tBER");
fclose(fp);
}
//RSSI BER SNR LOG
        return fn_NetSim_Zigbee_Init_F();
}
```

- Add the lines of code highlighted in red inside the **fn_NetSim_Zigbee_Run()** function under **PHYSICAL_IN_EVENT** as shown below:

```
case PHYSICAL_IN_EVENT:
{
NetSim_PACKET *pstruPacket;
PACKET_STATUS nPacketStatus;
double SNR;
double dBER;
FILE* fp;

pstruPacket = pstruEventDetails->pPacket;
if (pstruPacket->nReceiverId && pstruPacket->nReceiverId != pstruEventDetails-
>nDeviceId)
{
fnNetSimError("Different device packet received..");
assert(false);
return 0;
}
```

```
if (!ZIGBEE_CHANGERADIOSTATE(pstruEventDetails->nDeviceId,
WSN_PHY(pstruEventDetails->nDeviceId)->nRadioState, RX_ON_IDLE))
return 0;

if (WSN_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower -
GET_RX_POWER_mw(pstruPacket->nTransmitterId, pstruPacket->nReceiverId,
pstruEventDetails->dEventTime) >= WSN_PHY(pstruEventDetails->nDeviceId)-
>dReceiverSensivity)
pstruPacket->nPacketStatus = PacketStatus_Collided;
nPacketStatus = pstruPacket->nPacketStatus;
ZIGBEE_SINR(&SNR,
WSN_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower,
GET_RX_POWER_mw(pstruPacket->nTransmitterId, pstruPacket->nReceiverId,
pstruEventDetails->dEventTime));

dBER = fn_NetSim_Zigbee_CalculateBER(SNR);

//RSSI BER SNR LOG
double rxpwr = MW_TO_DBM(WSN_PHY(pstruEventDetails->nDeviceId)-
>dTotalReceivedPower);
double total_rxpwr = GET_RX_POWER_dbm(pstruPacket->nTransmitterId,
pstruPacket->nReceiverId, pstruEventDetails->dEventTime);
fp = fopen("ZIGBEE_BER_LOG.txt", "a+");
if (fp)
{
fprintf(fp, "\n%lld,\t\t%s,\t%s,\t%lf,\t%lf,\t\t%lf", pstruPacket->nPacketId,
DEVICE_NAME(pstruPacket->nTransmitterId),
DEVICE_NAME(pstruPacket->nReceiverId),
rxpwr, total_rxpwr, dBER);
fclose(fp);
}
//RSSI BER SNR LOG
```

if (fn_NetSim_Packet_DecideError(dBER, pstruEventDetails->dPacketSize))

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

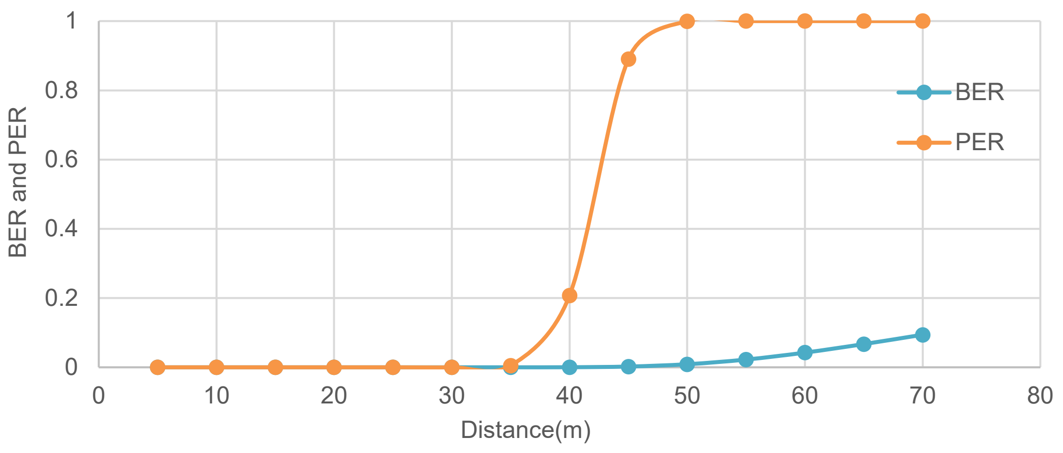
Step 10: Run the simulation for 10 Seconds. Once the simulation is complete, it will generate a text file named **ZIGBEE_BER_LOG.txt** containing **RSSI** and **BER** in the binary folder of NetSim. i.e. **<NetSim Install Directory>/bin**.

21.6 Output

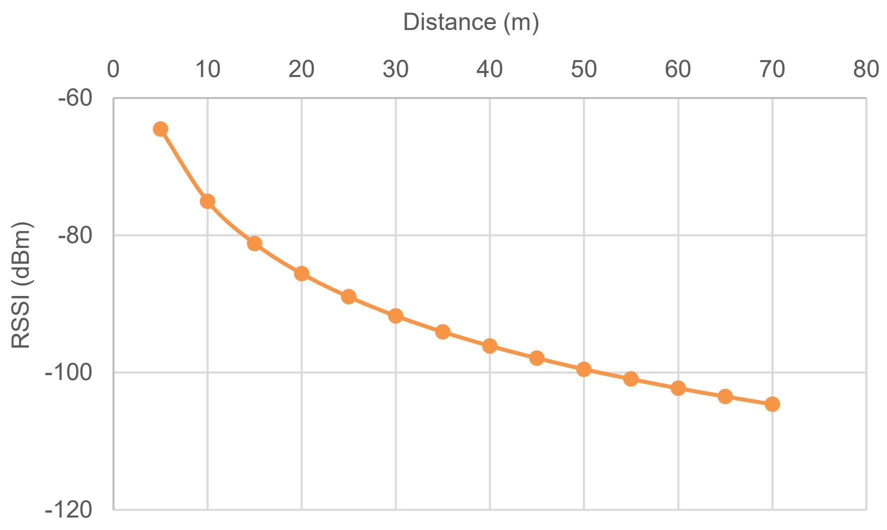
RSSI, PER, BER vs. Distance (path-loss: linear in log-distance, with $\eta = 3.5$)				
Distance(m)	**RSSI (dBm)** (pathloss model)	**BER**	**PER**	**PLR** (after MAC retransmissions*)
5	-64.51	0.00	0	0
10	-75.04	0.00	0	0
15	-81.20	0.00	0	0
20	-85.58	0.00	0	0
25	-88.97	0.00	0	0
30	-91.74	0.00	0	0
35	-94.08	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 21-1: RSSI, PER, BER from ZIGBEE_BER_LOG.txt vs. Distance

Comparison Charts:



(a)



(b)

Figure 21-6: (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 5×10^{-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,

$$PL - \text{ } packet\ length\ in\ bits\ at\ the\ PHY\ layer$$

$$PL\ (bits) = \big(70\ (payload) + 57 (overhead)\big) * 8$$

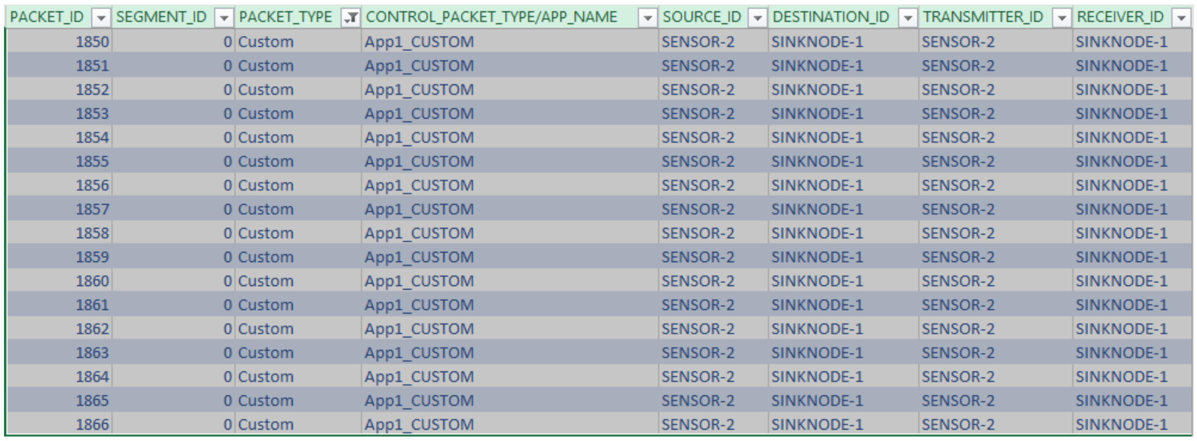
The PLR in the above table has been obtained from NetSim, which implements the details of the IEEE 802.15.4 MAC acknowledgement and reattempt mechanism. This mechanism is complex, involving a MAC acknowledgement, time-outs, and multiple reattempts. Analysis of the $PLR,$ therefore, is not straightforward. 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.

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

$$\begin{aligned} &Total\ number\ of\ Lost\ packets \\ &\qquad = Total\ number\ of\ Packet\ Sent\ by\ SourceMAC \\ &\qquad - Packets\ Received\ at\ Destination\ MAC \end{aligned}$$

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.

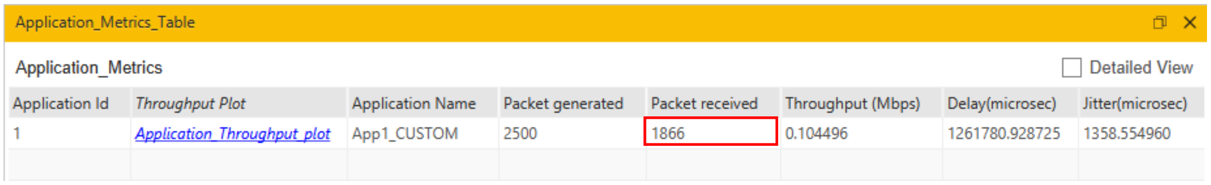


PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
1850	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1851	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1852	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1853	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1854	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1855	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1856	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1857	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1858	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1859	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1860	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1861	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1862	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1863	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1864	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1865	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1866	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1

Figure 21-7: 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 21-8.**



Application_Metrics_Table

Application_Metrics

Detailed View

Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)
1	Application_Throughput_plot	App1_CUSTOM	2500	1866	0.104496	1261780.928725	1358.554960

Figure 21-8: 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,

$$Total\ number\ of\ Packet\ Sent\ by\ SourceMAC = 463$$

$$Packets\ Received\ at\ Destination\ MAC = 256$$

$$Total\ number\ of\ Lost\ packets = 463 - 256 = 207$$

$$PLR = \frac{207}{463} = 0.447$$

21.7 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, so as to increase the sensor-sink distance beyond 40 meters.

21.8 Part 2 – Reaching a Longer Distance by Multihopping

Sample1

NetSim UI displays the configuration file corresponding to this experiment as shown below **Figure 21-9.**

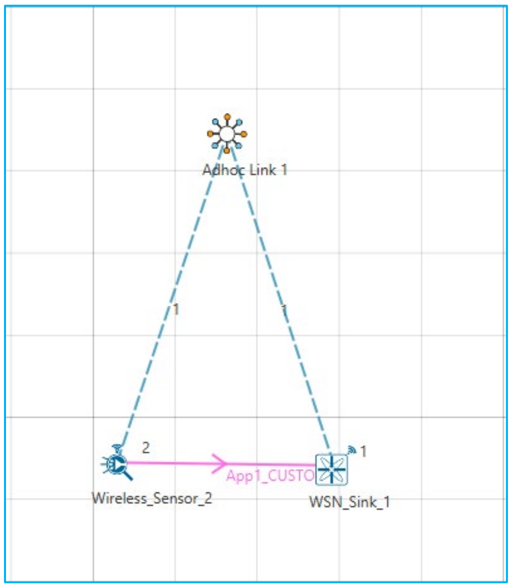


Figure 21-9: Network topology for Sample 1

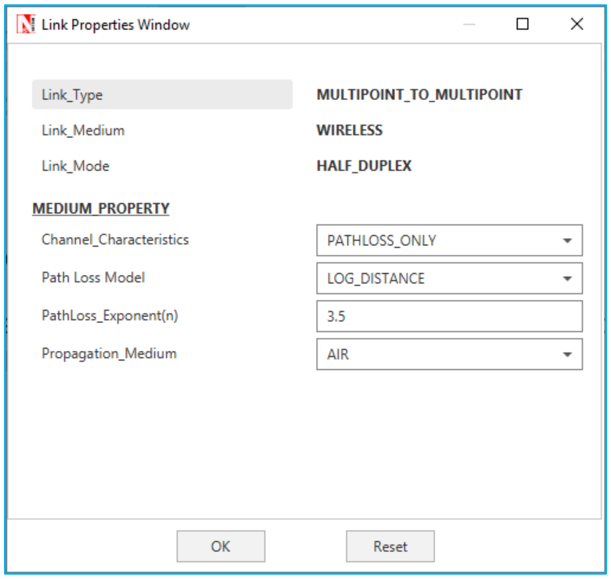
21.9 Procedure:

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 21-10.**



Link_Type	MULTIPOINT_TO_MULTIPOINT
Link_Medium	WIRELESS
Link_Mode	HALF_DUPLEX
MEDIUM_PROPERTY	
Channel_Characteristics	PATHLOSS_ONLY
Path Loss Model	LOG_DISTANCE
PathLoss_Exponent(n)	3.5
Propagation_Medium	AIR

Figure 21-10: 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 µs.

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:

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

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**.

Sample-2

NetSim UI displays the configuration file corresponding to this experiment as shown below **Figure 21-11.**

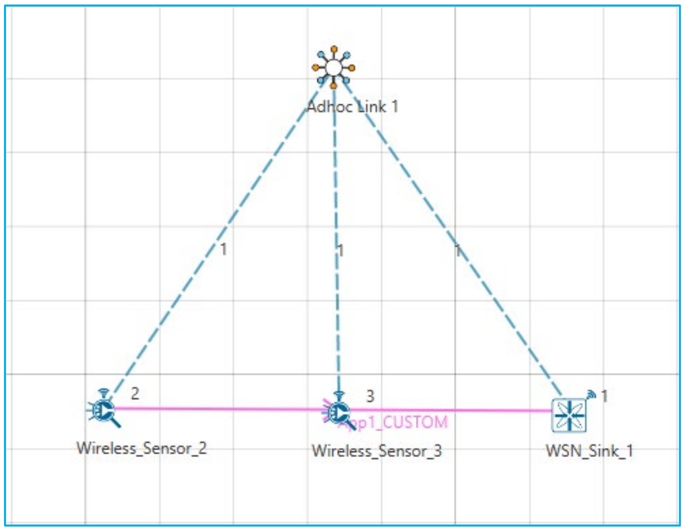


Figure 21-11: Network topology for Sample 2

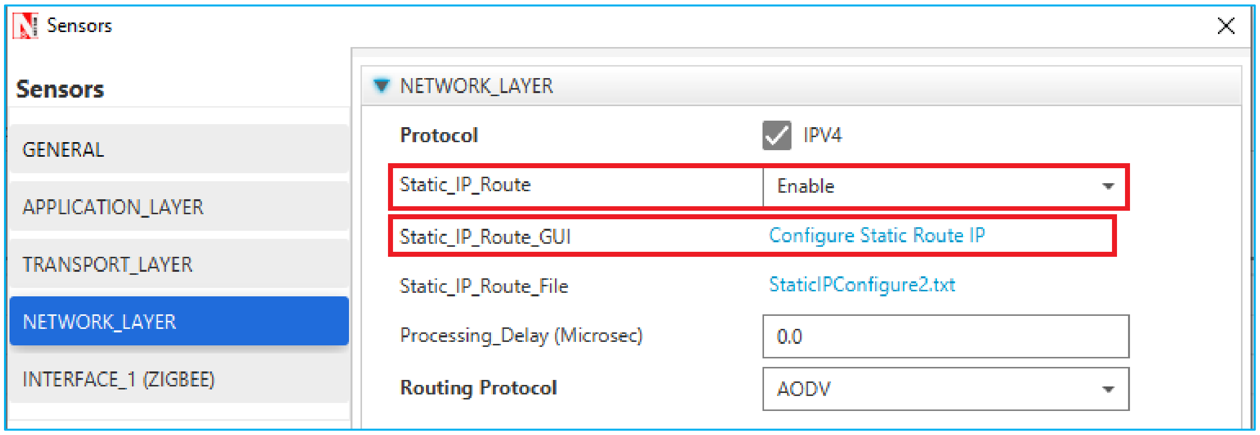
21.10 Procedure

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 21-12**, Enable - Static IP Route ->Click on **Configure Static Route IP**.



Protocol	✓ IPV4
Static_IP_Route	Enable
Static_IP_Route_GUI	Configure Static Route IP
Static_IP_Route_File	StaticIPConfigure2.txt
Processing_Delay (Microsec)	0.0
Routing Protocol	AODV

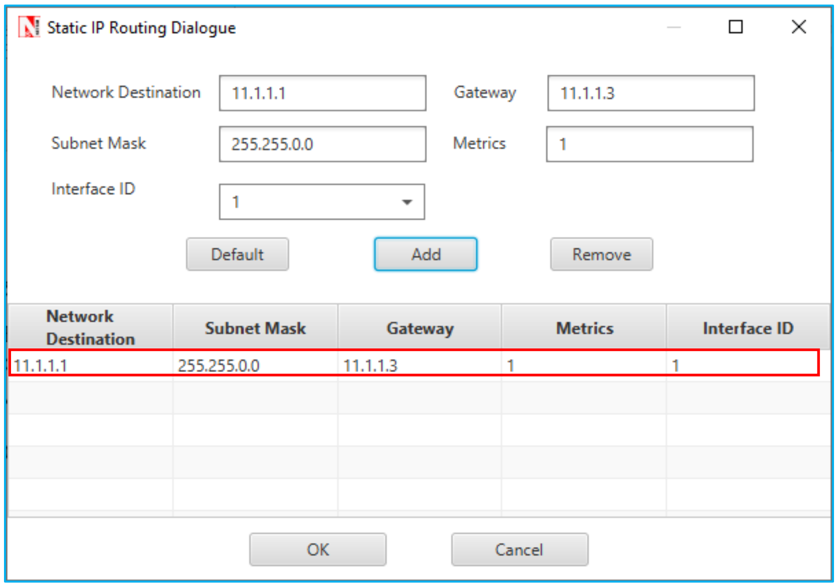
Figure 21-12: Network layer properties window

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**.

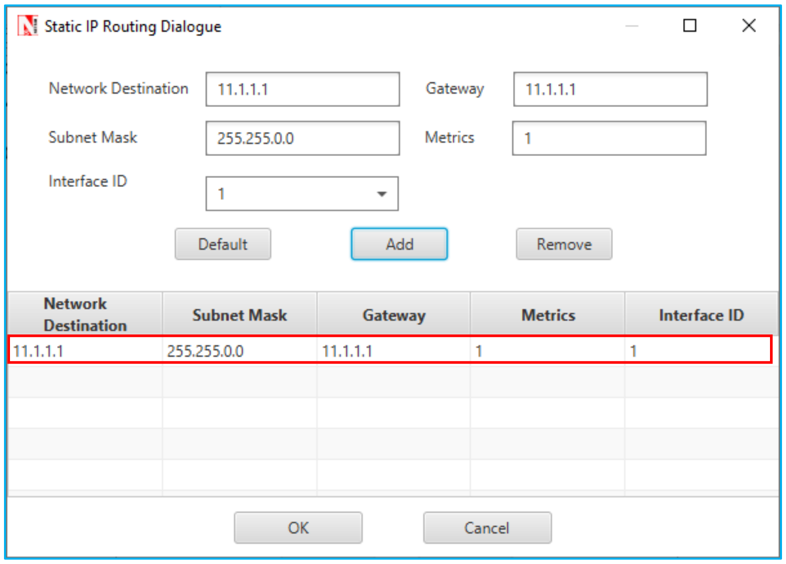

Static IP Routing Dialogue

Network Destination: 11.1.1.1
Gateway: 11.1.1.3
Subnet Mask: 255.255.0.0
Metrics: 1
Interface ID: 1

Network Destination	Subnet Mask	Gateway	Metrics	Interface ID
11.1.1.1	255.255.0.0	11.1.1.3	1	1

Figure 21-13: Static Route configuration for Wireless Sensor 2

Similarly, Static IP is set for Wireless Sensor 3 as shown below **Figure 21-14**.


Static IP Routing Dialogue

Network Destination: 11.1.1.1
Gateway: 11.1.1.1
Subnet Mask: 255.255.0.0
Metrics: 1
Interface ID: 1

Network Destination	Subnet Mask	Gateway	Metrics	Interface ID
11.1.1.1	255.255.0.0	11.1.1.1	1	1

Figure 21-14: 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 21-2**.

21.11 Output

	Source-Sink Distance (m)	Packets Generated	Packets Received	Packets Errored (PHY)	Packets Collided	Packet Loss (MAC)	PLR	Mean Delay (μs)
Direct sensor-sink link	40	1000	1012	244	0	0	0	6514.45
Router between sensor and sink	80 (router at midpoint)	1000	1015	540	0	0	0	14239.94

Table 21-2: 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.**

21.12 Inference

In Part 1 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.

Part 2, Sample 1 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 sample 2, 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.