Routing protocol for Internet-of-things networks considering the energy constraint of the devices

--Sunayan Shaik (01622923)

Objective of project:

Design a Routing Protocol for internet of things considering the energy as a constraint. Create a topology and simulate packet flow using various sources nodes which is consumed by a gateway sink. Control the path taken by packet to reach gateway using the energy available on a node as a parameter.

Environment:

ns-3, Ubuntu 16.04

Objective of project:

This protocol is designed by considering that IOT devices are static and low energy and hence their transmission range is small. So I divided all the IOT nodes into multiple tiers and each tier can send packets only to one of the nodes in downstream tier. From here packets will be routed further to downstream tiers till they reach gateway. This can be compared with an analogy of a home and assume the router is present near the front door and there are three rooms. The IOT devices from the farthest room sends packets to the next closest room and so on till data is reached to router.
The concept of load balancing used in web services is an inspiration for this design but here I am using the amount of energy available on a particular node as a parameter to route traffic.

Project Design:

High-level design of the project:

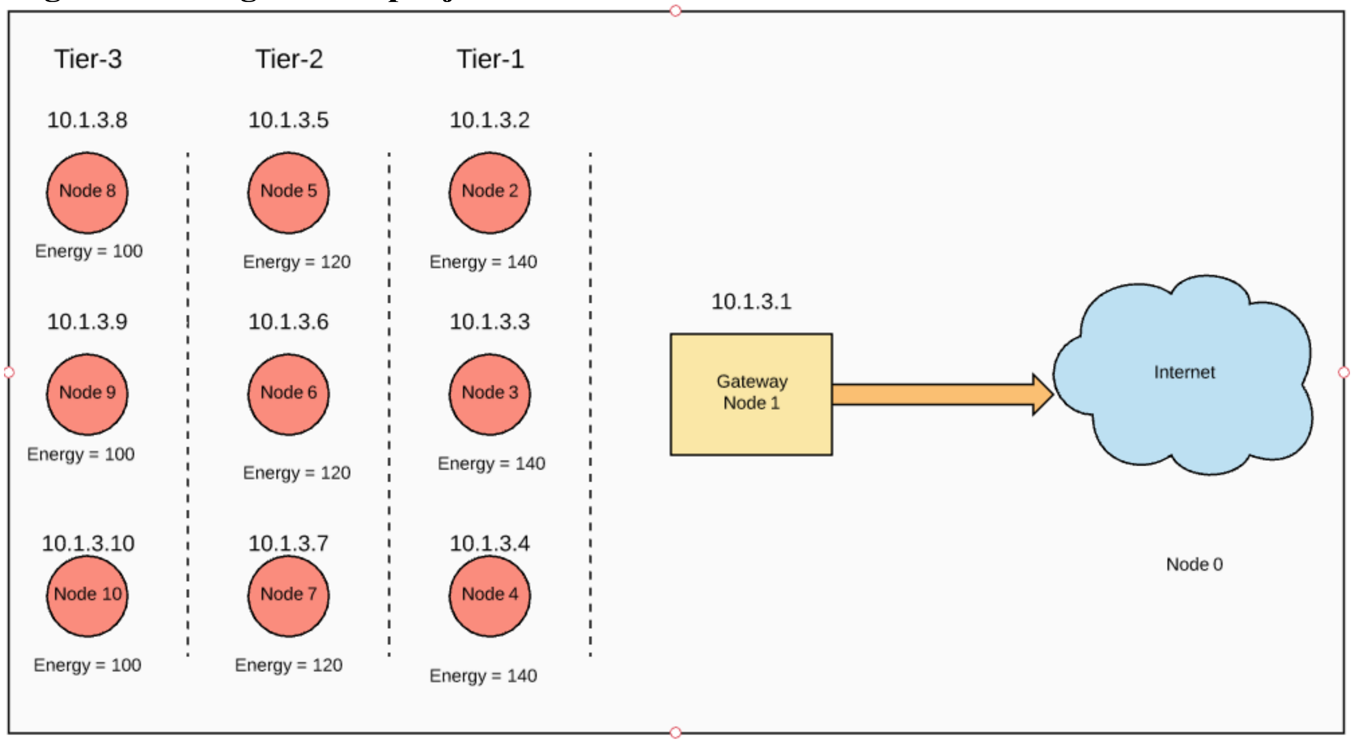


Figure 1.1

The topology implemented in this project is divided into three major tiers namely- Tier-1, Tier-2, Tier-3. Each tier consists of a set of 3 nodes. The gateway is considered as first node. The IP addresses of each node including gateway is as shown in figure 1.1 above.

Each node is provided with a certain amount of energy initially. As Nodes closer to gateway route more traffic than the farthest tiers they are given more energy by default. And the nodes that are farthest need less energy as they don't have to route traffic from downstream tier nodes.

Routing Protocol:
→ When a packet is generated from a source IOT node (say in Tier 3) we check the energy levels of the nodes in the downstream tier (Tier 2). Among those nodes we get the node with highest available energy and forward packet to that node.
→ When packet reaches a node to be routed (say in Tier 2) we check the energy levels of the nodes in the downstream tier (Tier 1). Among those nodes we get the node with highest available energy and forward packet to that node.
→ The available energy on the node is reduced by a small amount (here by 10 units) after either packet is generated or packet is transmitted from that node.

Implementation using NS-3:

Simulation in NS-3:
In NS-3 simulation is generated by Creating Nodes, Node devices, Channels, installing internet on the nodes, Creating a Source to send packets and a Destination to receive packets.

Routing in NS-3:
https://www.nsnam.org/docs/models/html/routing-overview.html
In NS3 Routing is controlled using Ipv4RoutingProtocol class. By default, if we do not specify which routing to use it picks up default routing. This can be overridden by passing a Routing Object which is inherited from Ipv4RoutingProtocol class and by adding custom implementations for RouteOutput and RouteInput methods.

Ipv4RoutingHelper class can be used to populate and configure custom Ipv4RoutingProtocol class.

NS-3 Module structure & Module usage with nodes:
NS-3 code structure is divided into modules and those modules are used to simulate network flow. I created a ns-3 module (iot-energy-optimal-routing) which performs the above mentioned routing protocol.

iot-energy-optimal-routing module is constructed following the steps in
https://www.nsnam.org/docs/manual/html/new-modules.html

Important classes that are used to construct this module :
1. *IotEnergyOptimalRoutingHelper* is a child class of *Ipv4RoutingHelper*.
2. *IotEnergyOptimalRouting* is a child class of *Ipv4RoutingProtocol*.
3. *IotEnergyOptimalRouteProcessor* is used by *IotEnergyOptimalRouting* class to Store and maintain energy levels of each node after packet traversal.

Snippet for Installing Internet with custom Ipv4RoutingProtocol

```
IotEnergyOptimalRoutingHelper iotEnergyOptimalRoutingHelper;
Ptr<IotEnergyOptimalRouteProcessor> iotEnergyOptimalRouteProcessor =
CreateObject<IotEnergyOptimalRouteProcessor> ();
iotEnergyOptimalRoutingHelper.Set ("RoutingProcessor", PointerValue
(iotEnergyOptimalRouteProcessor));

InternetStackHelper iotNodesInternetStackHelper;
iotNodesInternetStackHelper.SetRoutingHelper(iotEnergyOptimalRoutingHelper);
iotNodesInternetStackHelper.Install (iotNodes);
```

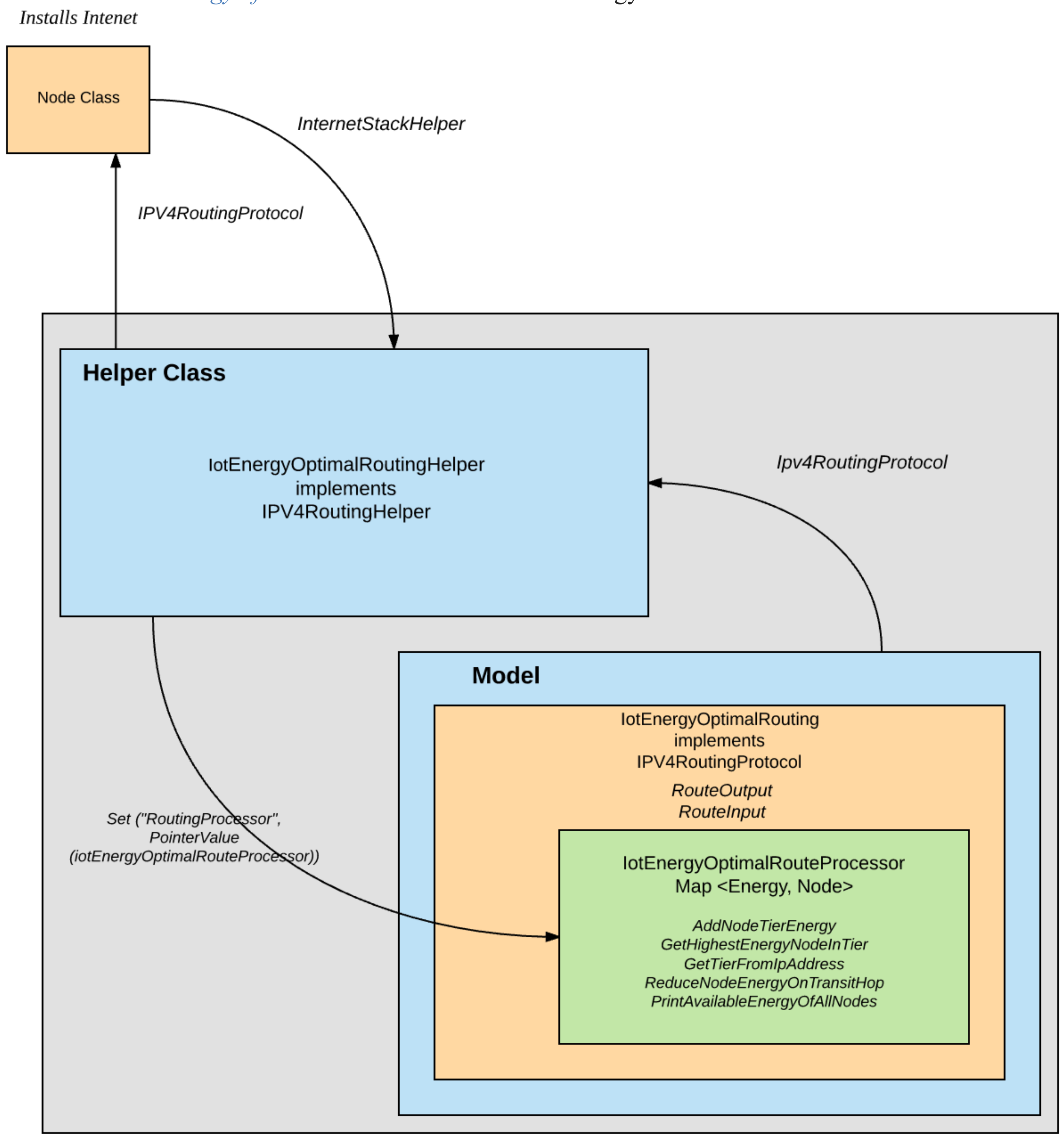
Important methods:

IotEnergyOptimalRouting

RouteOutput → Gets Called when packet is generated from a node.
RouteInput → Gets Called when packet has reached this node to be forwarded to node.

IotEnergyOptimalRouteProcessor

AddNodeTierEnergy → Adds and stores tier and energy info of node into a map.
GetHighestEnergyNodeInTier → Gets a Node IP address in a tier with highest available energy.
GetTierFromIpAddress → Gets the Tier info from IP address of a node.
ReduceNodeEnergyOnTransitHop → Once packet is transmitted from a node its energy is reduced by 10 units
PrintAvailableEnergyOfAllNodes → Prints available energy for a node.



Installation steps on ns-3 and running the project on Ubuntu 16.04 :

The code was implemented and tested in latest ns-3-dev version on Ubuntu 16.04. Instructions in the below link were followed to install latest ns-3-dev version. https://www.nsnam.org/wiki/Installation

Steps of installation and running simulation of this project:

Download all the dependency libraries:

We need to install a few libraries in Ubuntu in order to have ns-3 work as expected. The following bash(download-install-dependency-libraries.sh) has various libraries that are needed to be installed. This is combined in the source code directory submitted. Execute the following command inside project directory and provide the system password to install all the corresponding libraries using apt-get:

```
./ download-install-dependency-libraries.sh
```

This project can be executed from any folder. For instance, considering **/home/suny/cs646/project** as the base folder for installing NS-3(Please use any relative path).

Clone ns-3-allinone repo:

```
mkdir -pv /home/suny/cs646/project/repos
cd /home/suny/cs646/project/repos
hg clone http://code.nsnam.org/ns-3-allinone
```

Download all the corresponding source code of ns-3-dev:

```
cd /home/suny/cs646/project/repos/ns-3-allinone
./download.py -n ns-3-dev
```

Add iot-energy-optimal-routing module to ns-3:

Copy the iot-energy-optimal-routing module provided in the submission folder and place it in **ns-3-allinone/ns-3-dev/src** folder.

```
cd <Submission_folder>
cp -r iot-energy-optimal-routing /home/suny/cs646/project/repos/ns-3-allinone/ns-
3-dev/src/iot-energy-optimal-routing
```

Please make sure iot-energy-optimal-routing folder is copied successfully into **/home/suny/cs646/project/repos/ns-3-allinone/ns-3-dev/src folder.

Build source code: (Trigger build.py file From ns-3-dev folder. This step will take some time)

```
cd /home/suny/cs646/project/repos/ns-3-allinone/
./build.py
```

Copy topology simulation file to scratch folder:

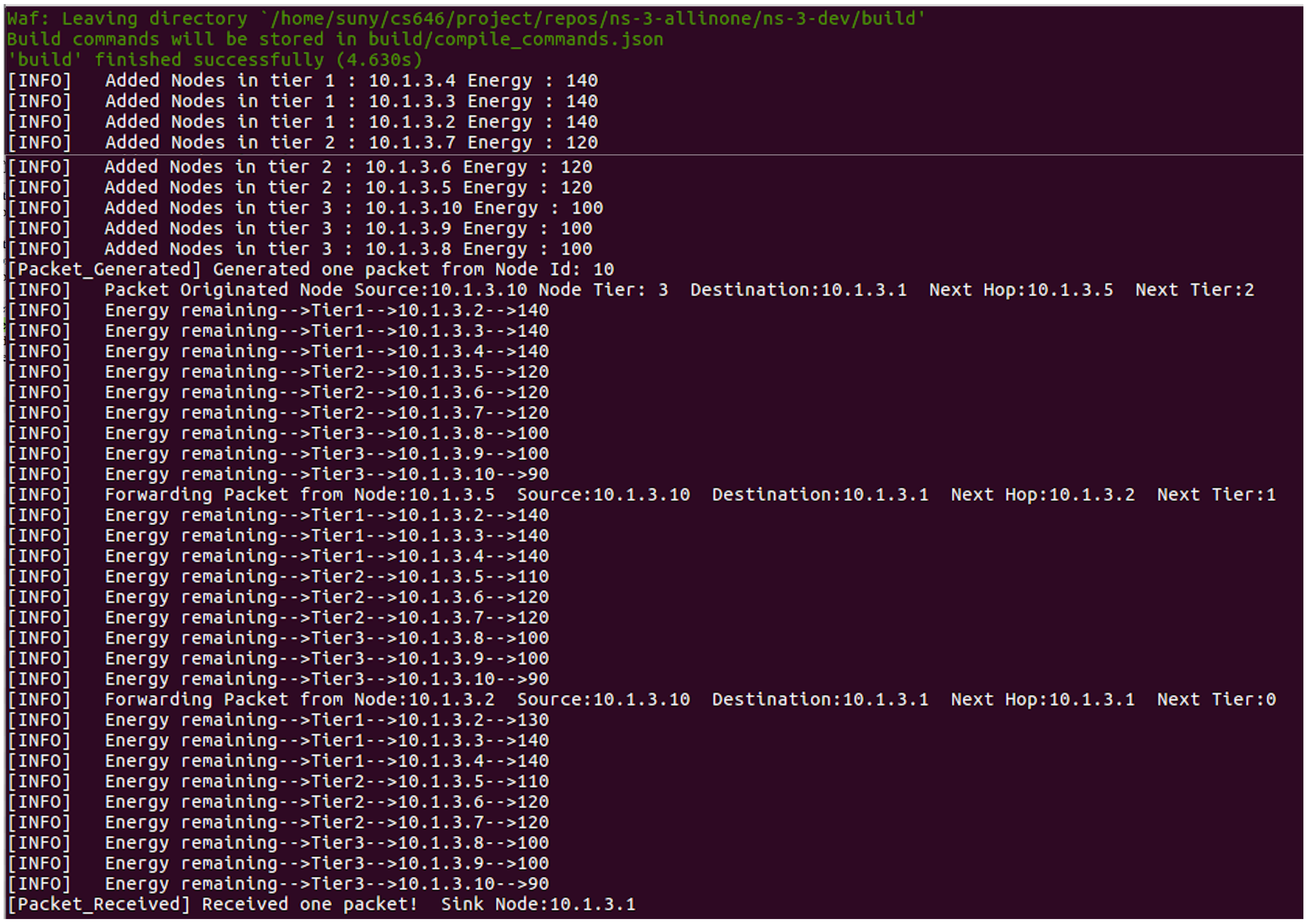
Copy the topology simulation file(***iot-energy-optimal-route-example-topology.cc***) from ***ns-3-dev/src/iot-energy-optimal-routing/examples/iot-energy-optimal-route-example-topology.cc*** to ***ns-3-dev/scratch/iot-energy-optimal-route-example-topology.cc***

```
cp /home/suny/cs646/project/repos/ns-3-allinone/ns-3-dev/src/iot-energy-optimal-
routing/examples/iot-energy-optimal-route-example-topology.cc
/home/suny/cs646/project/repos/ns-3-allinone/ns-3-dev/scratch/iot-energy-optimal-
route-example-topology.cc
```

Run topology simulation file using waf; (don't include .cc in waf command)

```
cd /home/suny/cs646/project/repos/ns-3-allinone/ns-3-dev
./waf --run scratch/iot-energy-optimal-route-example-topology
```

Simulation of packet traversal will be displayed on output screen with energy levels remaining in each node.
** Output on consecutive run of this script may differ as we randomly select the first node to send packet if energies are same for nodes in the tier. (Path may be different but overall simulation will go to node with highest energy in next consecutive packets).



```
Waf: Leaving directory `/home/suny/cs646/project/repos/ns-3-allinone/ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.630s)
[INFO]   Added Nodes in tier 1 : 10.1.3.4 Energy : 140
[INFO]   Added Nodes in tier 1 : 10.1.3.3 Energy : 140
[INFO]   Added Nodes in tier 1 : 10.1.3.2 Energy : 140
[INFO]   Added Nodes in tier 2 : 10.1.3.7 Energy : 120
[INFO]   Added Nodes in tier 2 : 10.1.3.6 Energy : 120
[INFO]   Added Nodes in tier 2 : 10.1.3.5 Energy : 120
[INFO]   Added Nodes in tier 3 : 10.1.3.10 Energy : 100
[INFO]   Added Nodes in tier 3 : 10.1.3.9 Energy : 100
[INFO]   Added Nodes in tier 3 : 10.1.3.8 Energy : 100
[Packet_Generated] Generated one packet from Node Id: 10
[INFO]   Packet Originated Node Source:10.1.3.10 Node Tier: 3  Destination:10.1.3.1  Next Hop:10.1.3.5  Next Tier:2
[INFO]   Energy remaining-->Tier1-->10.1.3.2-->140
[INFO]   Energy remaining-->Tier1-->10.1.3.3-->140
[INFO]   Energy remaining-->Tier1-->10.1.3.4-->140
[INFO]   Energy remaining-->Tier2-->10.1.3.5-->120
[INFO]   Energy remaining-->Tier2-->10.1.3.6-->120
[INFO]   Energy remaining-->Tier2-->10.1.3.7-->120
[INFO]   Energy remaining-->Tier3-->10.1.3.8-->100
[INFO]   Energy remaining-->Tier3-->10.1.3.9-->100
[INFO]   Energy remaining-->Tier3-->10.1.3.10-->90
[INFO]   Forwarding Packet from Node:10.1.3.5  Source:10.1.3.10  Destination:10.1.3.1  Next Hop:10.1.3.2  Next Tier:1
[INFO]   Energy remaining-->Tier1-->10.1.3.2-->140
[INFO]   Energy remaining-->Tier1-->10.1.3.3-->140
[INFO]   Energy remaining-->Tier1-->10.1.3.4-->140
[INFO]   Energy remaining-->Tier2-->10.1.3.5-->110
[INFO]   Energy remaining-->Tier2-->10.1.3.6-->120
[INFO]   Energy remaining-->Tier2-->10.1.3.7-->120
[INFO]   Energy remaining-->Tier3-->10.1.3.8-->100
[INFO]   Energy remaining-->Tier3-->10.1.3.9-->100
[INFO]   Energy remaining-->Tier3-->10.1.3.10-->90
[INFO]   Forwarding Packet from Node:10.1.3.2  Source:10.1.3.10  Destination:10.1.3.1  Next Hop:10.1.3.1  Next Tier:0
[INFO]   Energy remaining-->Tier1-->10.1.3.2-->130
[INFO]   Energy remaining-->Tier1-->10.1.3.3-->140
[INFO]   Energy remaining-->Tier1-->10.1.3.4-->140
[INFO]   Energy remaining-->Tier2-->10.1.3.5-->110
[INFO]   Energy remaining-->Tier2-->10.1.3.6-->120
[INFO]   Energy remaining-->Tier2-->10.1.3.7-->120
[INFO]   Energy remaining-->Tier3-->10.1.3.8-->100
[INFO]   Energy remaining-->Tier3-->10.1.3.9-->100
[INFO]   Energy remaining-->Tier3-->10.1.3.10-->90
[Packet_Received] Received one packet!  Sink Node:10.1.3.1
```

Output Explained:

Initial setup of the topology design is created by adding 10 nodes under each tier and IP addresses allotted to each node as per the topology design specified in figure1.1.
For example,
Added Nodes in tier 1 : 10.1.3.4 Energy : 140
This indicates the tier to which a node with their IP addresses belongs. Each node is initialized with a value of energy.
[Packet_Generated] Generated one packet from Node Id: 10
[INFO]   Packet Originated Node Source:10.1.3.10 Node Tier: 3 Destination:10.1.3.1 Next Hop:10.1.3.5 Next Tier:2
The above lines specifies the NodeId of the node where the packet was generated, the IP address of the Node Source, Node Tier , destination IP address, IP address of the next hop and its Tier information is displayed as output to keep track of the path of the packet followed when its forwarded to a destination.

[INFO] Energy remaining-->Tier3-->10.1.3.10-->90
The updated energy values are displayed in the output. When a packet is generated or forwarded on a node, the energy of each node decrements by a value of 10.

[INFO] Forwarding Packet from Node:10.1.3.5 Source:10.1.3.10 Destination:10.1.3.1 Next Hop:10.1.3.2 Next Tier:1
This indicates the packet has hoped to **10.1.3.5** on tier 2. And **10.1.3.5** forwards the packet to **10.1.3.2** on downstream tier.

[Packet_Received] Received one packet! Sink Node:10.1.3.1
The above line indicates that the forwarded packet has been received to gateway (sink) node which marks the end of one complete flow of a packet from source to sink.

Pictorial representation of simulations in different scenarios:

In the example simulation program packets are generated from various IOT nodes (Source) from different tiers and the following diagrams illustrate Packet flow from each tier to gateway. (Energy values shown diagrams are before packet has transferred and which are used to select the path to travel)

Case 1:

When a packet is generated at a node in Tier-3, the path followed while forwarding the packet from source to destination are as shown in figure 1.2 & 1.3. (If the energy is same it randomly goes to one of the node)

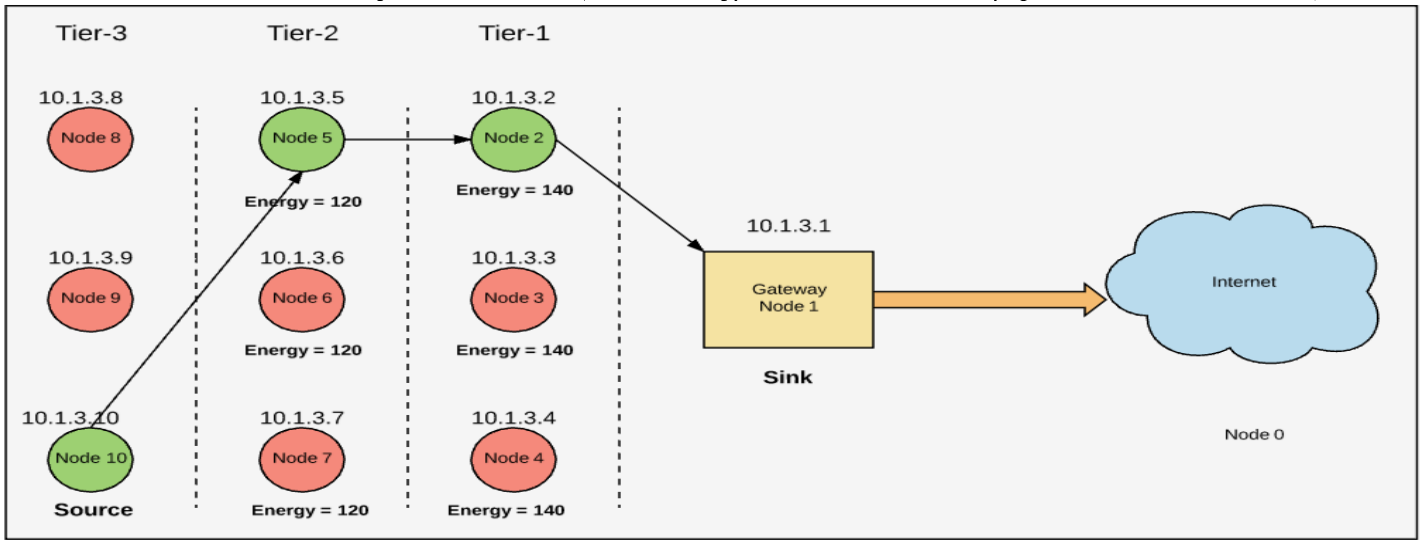


Figure 1.2 *Scenario of simulation when a source is in Tier-3 for first packet*

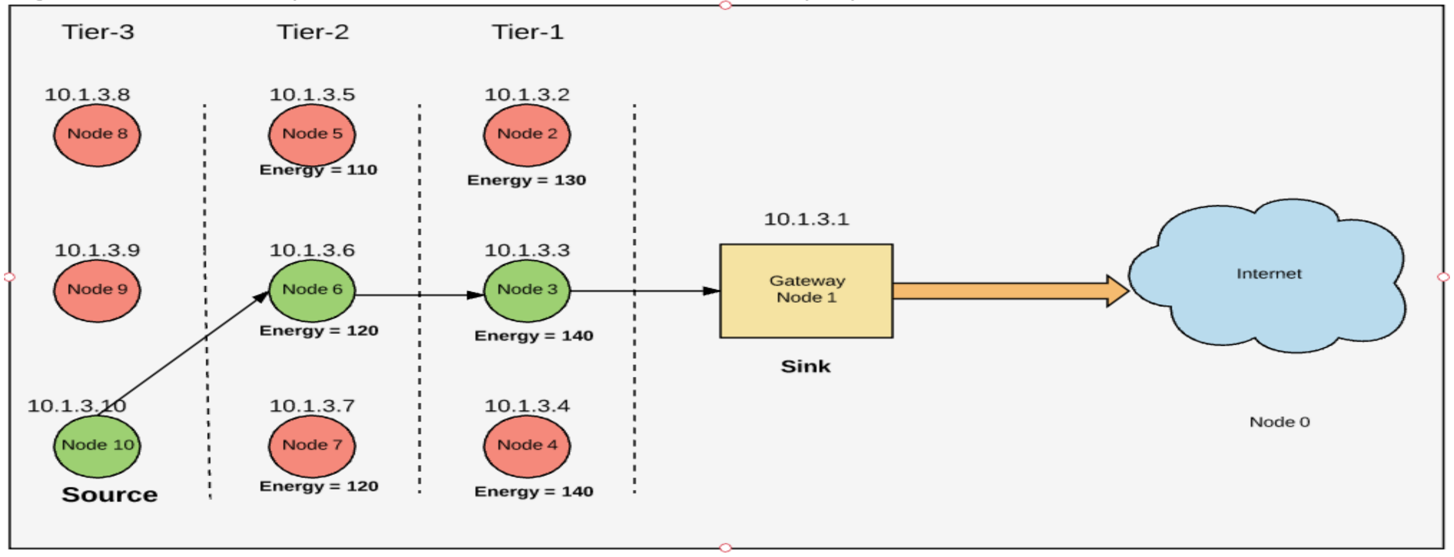


Figure 1.3 *Scenario of simulation when a source is in Tier-3 for second packet.*

Case 2:
When a packet is generated at a node in Tier-2, the path followed while forwarding the packet from source to destination are as shown in figure 1.4 & 1.5.

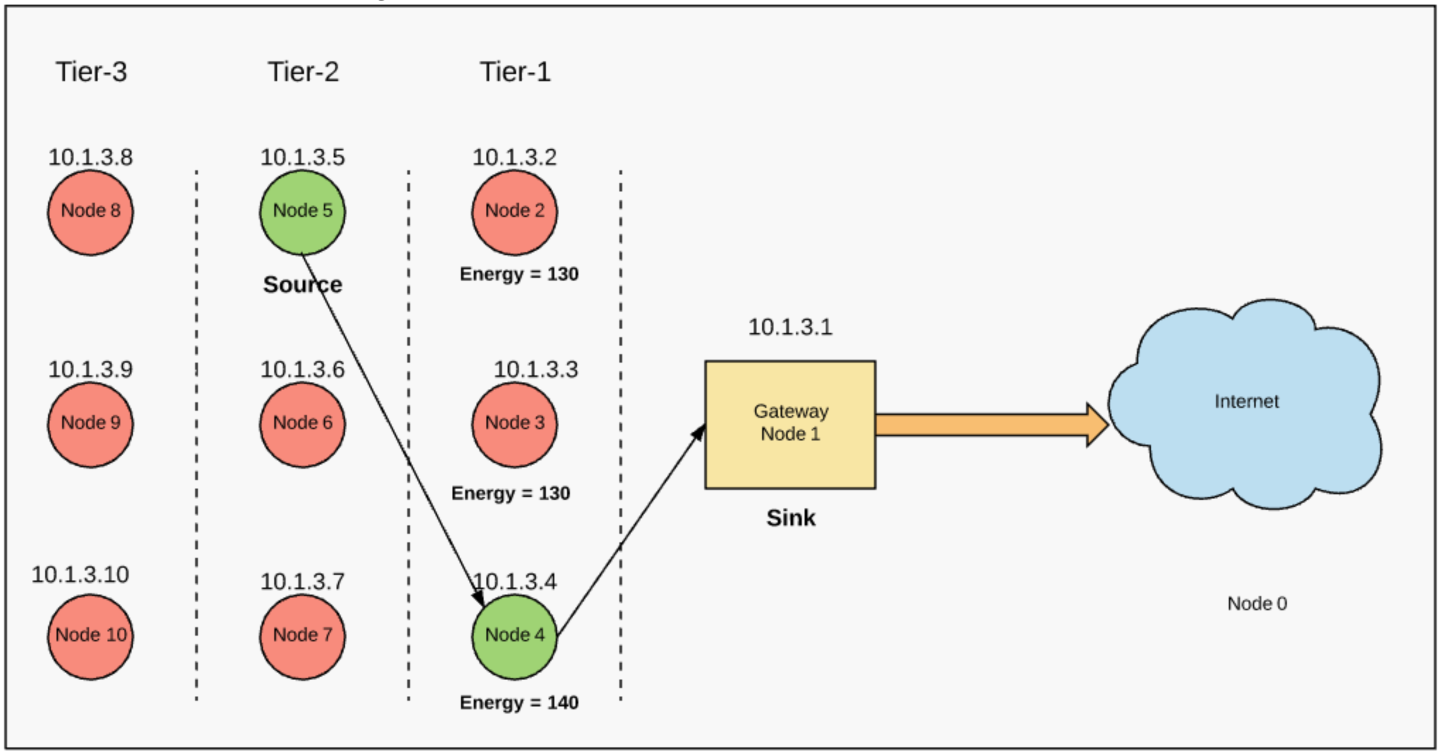


Figure 1.4 *Scenario of simulation when a source is in Tier-2 for first packet*

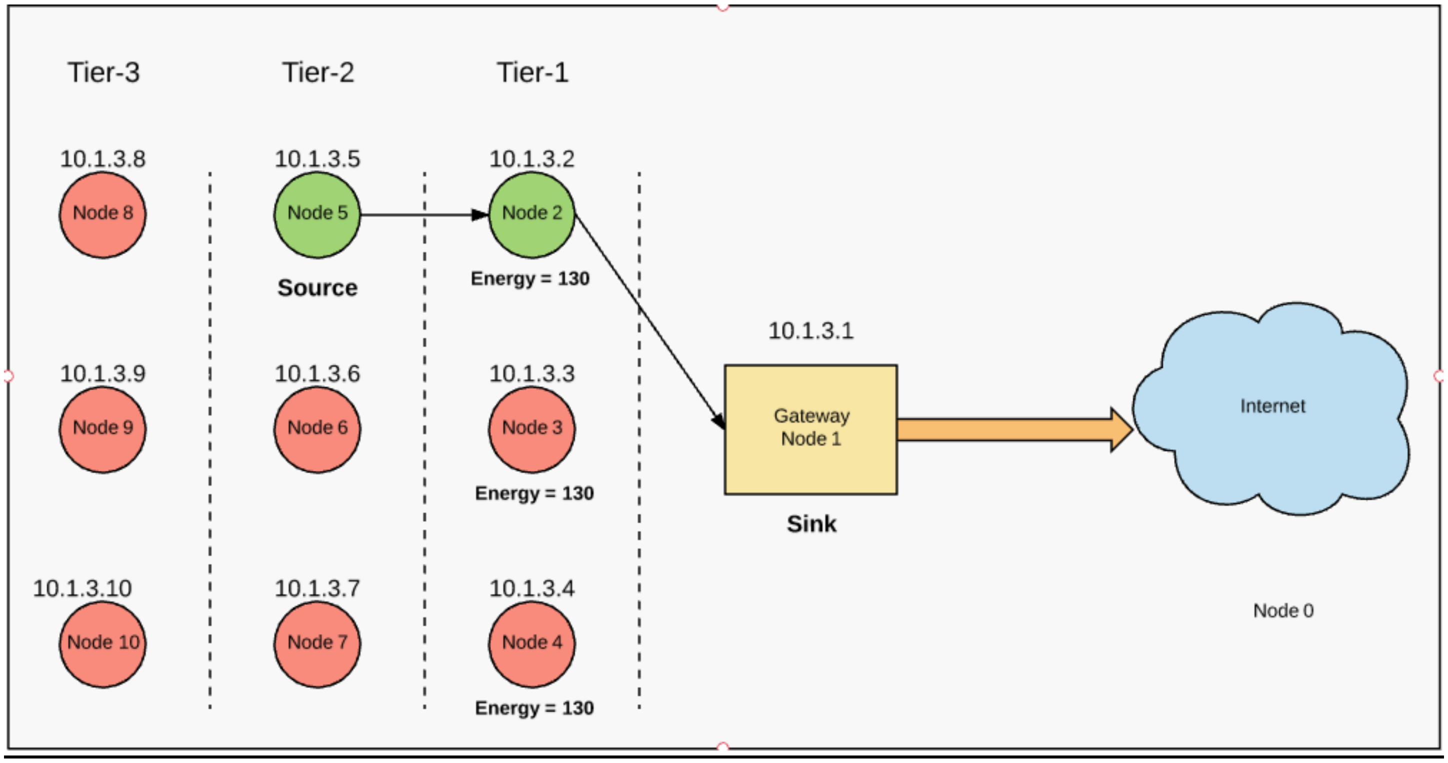


Figure 1.5 *Scenario of simulation when a source is in Tier-2 for second packet*

Case 3:
When a packet is generated at a node in Tier-2, the path followed while forwarding the packet from source to destination are as shown in figure 1.6.

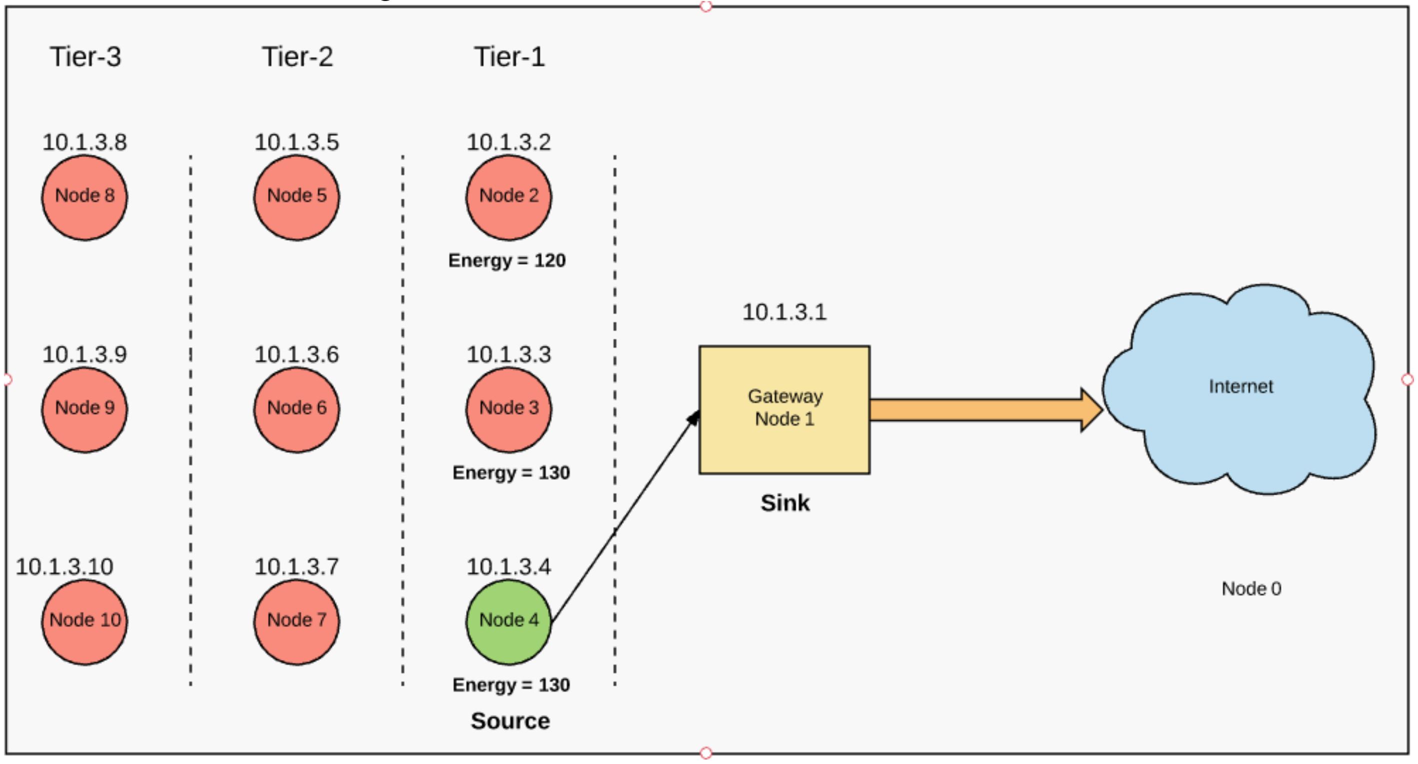


Figure 1.6 *Scenario of simulation when a source is in Tier-1*

TCP DUMP FILES:
In the *iot-energy-optimal-route-example-topology.cc* provided tracing is enabled by default.
So after running the program we can find the trace files in the ns-3-dev folder.

```
ls -lart /home/suny/cs646/project/repos/ns-3-allinone/ns-3-dev/ | grep .pcap
```

Trace files can be run using the following command (*iot_energy_optimal_topology_example_gateway-0-1.pcap* for example).

```
tcpdump -nn -tt -r iot_energy_optimal_topology_example_gateway-0-1.pcap
```

Experiences and Thoughts:
Gained a good knowledge on Routing in real world scenarios.
Gained expertise on designing topologies on NS-3 and building Modules.

References:

Load Balancing:
https://en.wikipedia.org/wiki/Load_balancing_(computing)

NS-3 Documentation:
https://www.nsnam.org/doxygen/modules.html
https://www.nsnam.org/docs/manual/html/new-modules.html
https://www.nsnam.org/docs/release/3.8/tutorial/tutorial_27.html
https://www.nsnam.org/docs/release/3.10/manual/html/routing.html
https://www.nsnam.org/doxygen/classns3_1_1_ipv4_routing_protocol.html
https://www.nsnam.org/doxygen/classns3_1_1_internet_stack_helper.html
https://www.nsnam.org/docs/manual/html/attributes.html