School of
Applied Computing

Network Simulation

NetSim

Exercise 8

Faculty of Architecture, Computing and Engineering

School of Applied Computing

Mount Pleasant Campus

Swansea

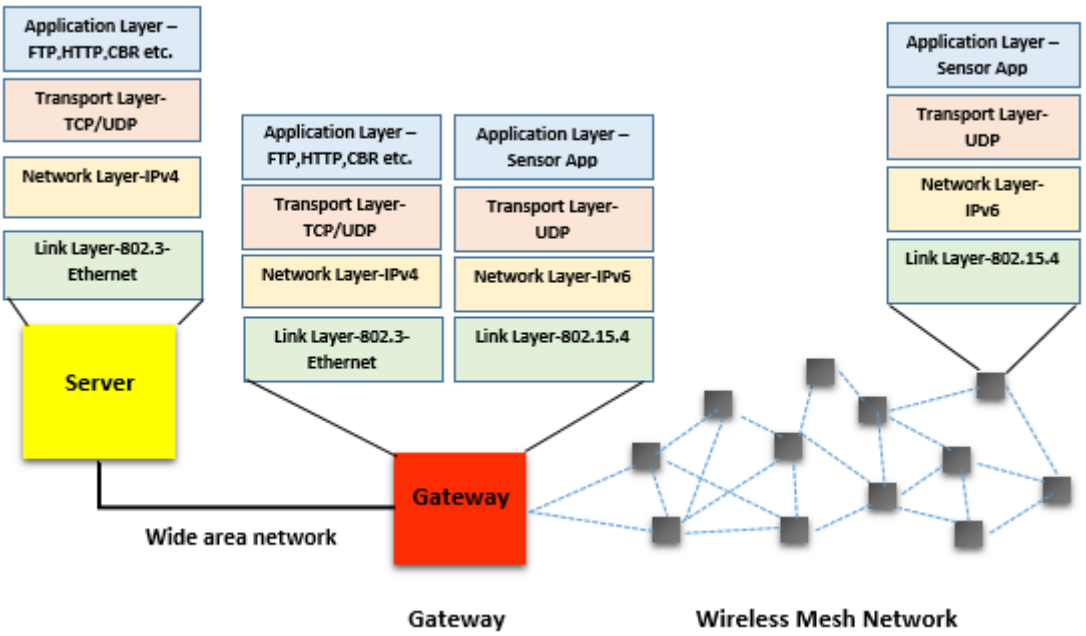
SA1 6ED

8. Introduction and working of internet of things (IoT)

8.1 Introduction

Internet of Things (**IoT**) is a network of physical devices, vehicles, buildings and other items—embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. An IoT network allows objects to be sensed and/or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit.

8.1.1 Simple IOT scenario



8.1.2 Components

1. **Sensors**

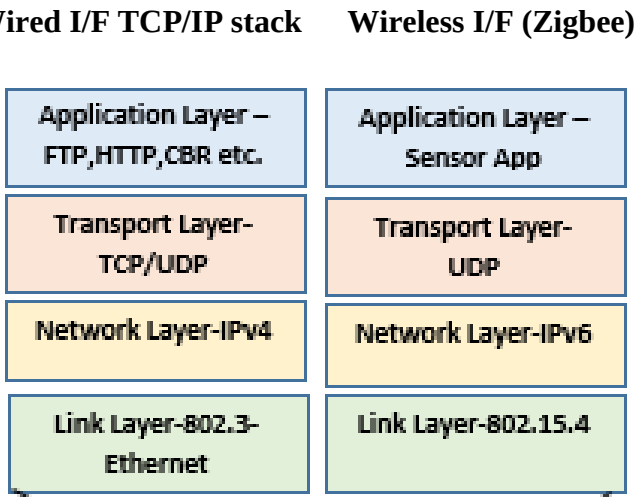
Sensors are used to detect physical phenomena such as light, heat, pressure, temperature, humidity etc. Sensors are regarded as a revolutionary information gathering method to build the information and communication system which will greatly improve the reliability and efficiency of infrastructure systems. It follows IPv6 addressing system. IP addresses are

the backbone to the entire IoT ecosystem. IPv6's huge increase in address space is an important factor in the development of the Internet of Things.

2. **LowPAN Gateway**

These are the Gateways to Internet for all the things/devices that we want to interact with. Gateway help to bridge the internal network of sensor nodes with the external Internet i.e., it will collect the data from sensors and transmitting it to the internet infrastructure.

A 6LowPAN Gateway will have 2 interfaces, one is Zigbee interface connected to sensors (follows 802.15.4 MAC and PHY) and the other is WAN interface connected to ROUTER.



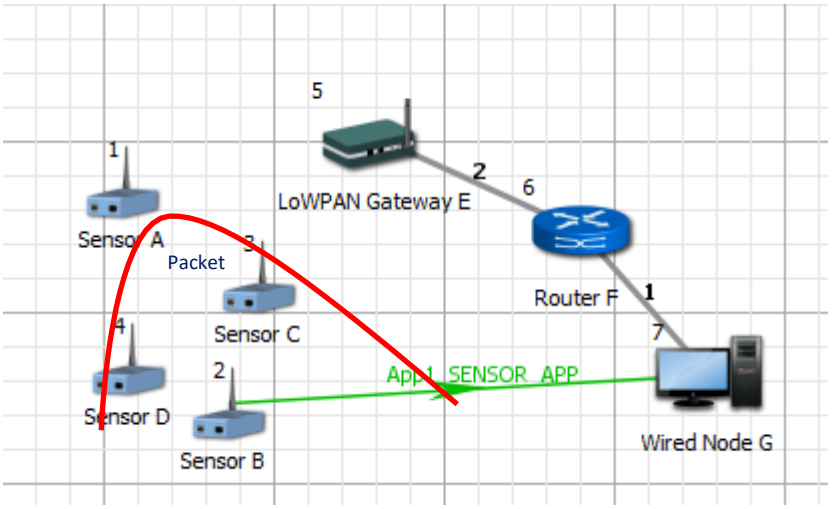
6LowPAN Gateway Stack at wired and wireless Interfaces

6LoWPAN is an acronym of IPv6 over Low Power Wireless Personal Area Network. The 6LoWPAN concept originated from the idea that "the Internet Protocol should be applied even to the smallest devices, and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.

8.2 PART A: To Model and Simulate an IOT Network Scenario in NetSim

Step 1: Go to Simulation -> New -> IOT

Step 2: Click and drop 4 sensors, 1 LowPAN Gateway, 1 router and 1 wired node. Connect Router, LowPAN Gateway and wired node using wired links

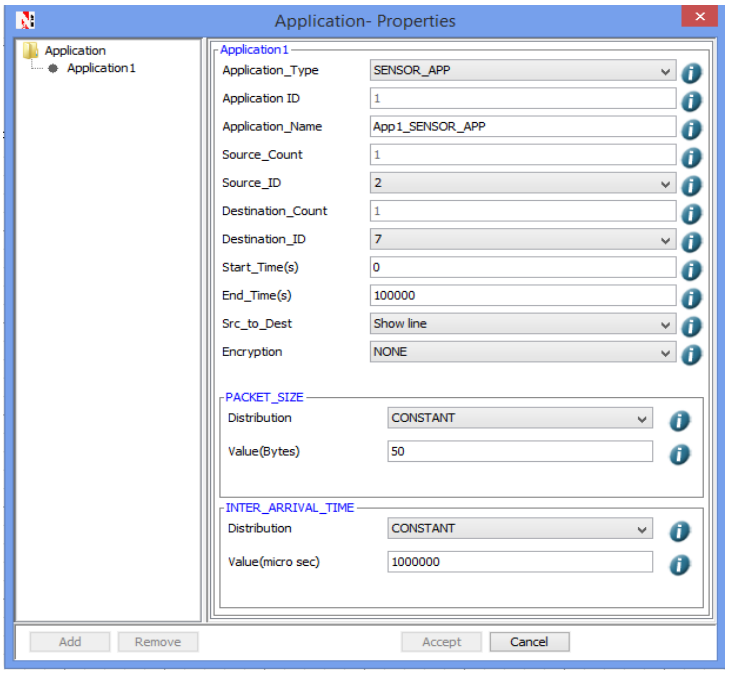


Step 3:

Application Properties:

To add application, drop the Application icon. Edit the Application properties as given in table. All other properties are default.

Application Type	Sensor_App
Source ID	2
Destination ID	7



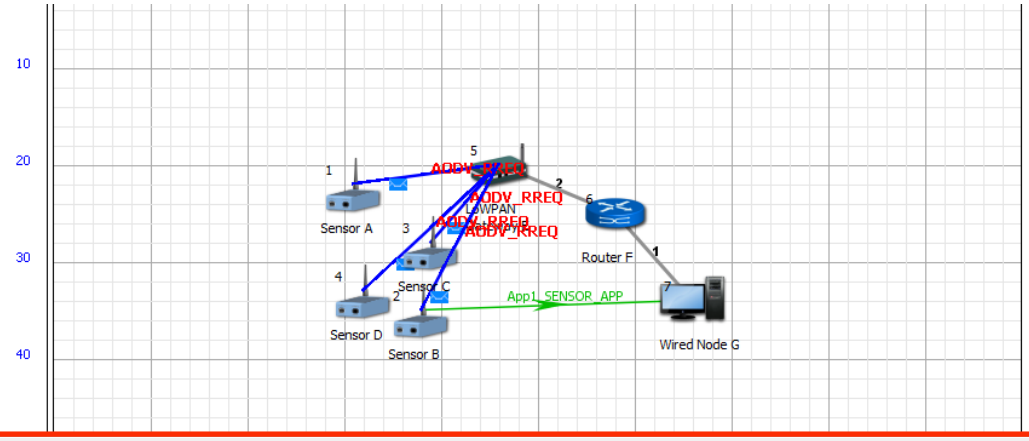
Step 4:

Click on Run Simulation icon and set simulation time = 100s

8.3 Output

At the end of simulation, NetSim provides various performance metrics such as Network Metrics, Link Metrics, Application Metrics, and Protocol Metrics etc.

In Packet Animation Table, users can see the packet flow.

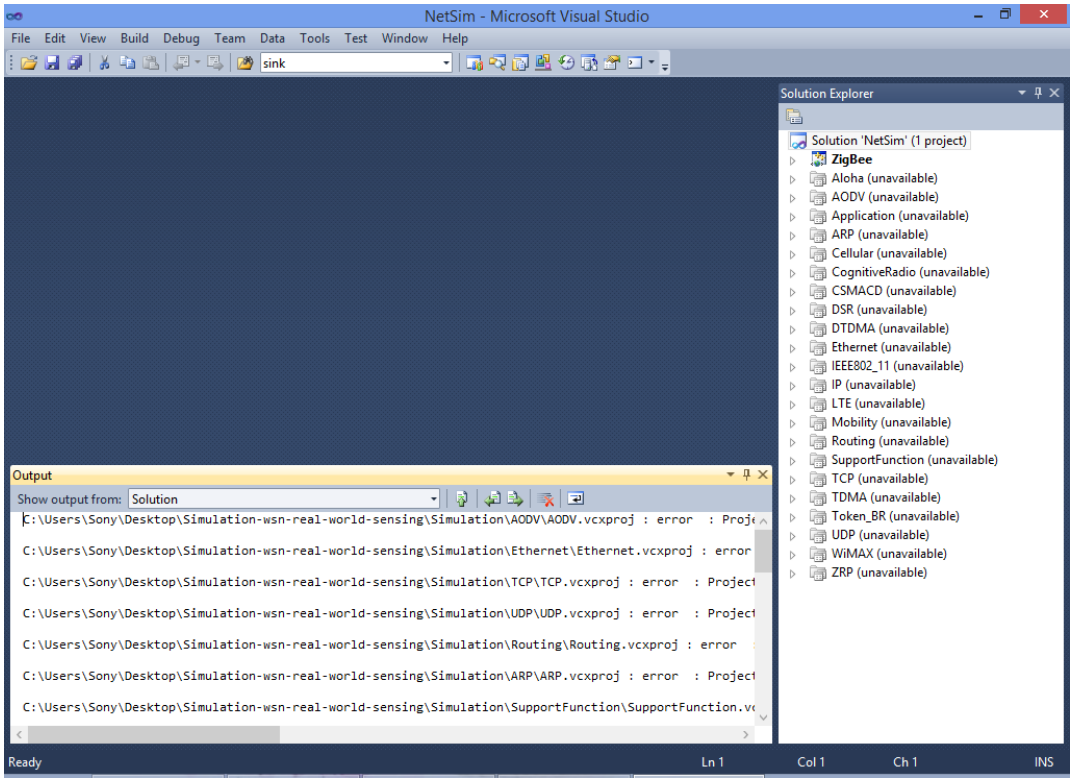


Source_Name	Destination_Name	Packet_Info	Status	Simulation_Time (Micro Sec)
LoWPAN Gateway E	Sensor A	AODV_RREQ	SUCCESS	6214753.00
LoWPAN Gateway E	Sensor D	AODV_RREQ	SUCCESS	6214753.00
LoWPAN Gateway E	Sensor C	AODV_RREQ	SUCCESS	6214753.00
LoWPAN Gateway E	Sensor B	AODV_RREQ	SUCCESS	6214753.00
LoWPAN Gateway E	Sensor B	Zigbee_ACK	SUCCESS	6010146.00
Router F	Wired Node G	APP1_SENSOR_APP-6	SUCCESS	6009966.20
LoWPAN Gateway E	Router F	APP1_SENSOR_APP-6	SUCCESS	6009954.00
Sensor B	LoWPAN Gateway E	APP1_SENSOR_APP-6	SUCCESS	6007106.00
Sensor D	Sensor A	AODV_RREP	SUCCESS	6005473.00
Sensor D	Sensor C	AODV_RREP	SUCCESS	6005473.00
Sensor D	Sensor B	AODV_RREP	SUCCESS	6005473.00
Sensor D	LoWPAN Gateway E	AODV_RREP	SUCCESS	6005473.00
Sensor A	Sensor D	AODV_RREP	SUCCESS	6003873.00
Sensor A	Sensor C	AODV_RREP	SUCCESS	6003873.00
Sensor A	Sensor B	AODV_RREP	SUCCESS	6003873.00

8.4 PART B: To print IP addresses of Source, Destination, Transmitter and Receiver devices and understand the flow from IPv6 to IPv4 network (NetSim standard version is required for this section)

1. Go to NetSim installed directory. The NetSim install directory would look something like < C:\Program Files (x86)\NetSim Standard\bin>
2. Open the simulation folder

3. Open the NetSim.sln file inside the Simulation folder, it displays a message –One or more projects in the solution were not loaded completely‖
4. Click on OK and when this opens in MS Visual Studio 2010, it would look like



5. Open IP project

Code Modifications to print ip addresses:

6. Add the following lines of code in NETWORK_OUT_EVENT and NETWORK_IN_EVENT events in fn_NetSim_IP_Run() function present in IP.c file

```
if(packet->nPacketType != PacketType_Control)
{
        IP_TO_STR(packet->pstruNetworkData->szSourceIP,s_ip);
        IP_TO_STR(packet->pstruNetworkData->szDestIP,d_ip);
        IP_TO_STR(packet->pstruNetworkData->szGatewayIP,t_ip);
        IP_TO_STR(packet->pstruNetworkData->szNextHopIp,r_ip);
        if(!fp_ip)
        {
                fp_ip=fopen("ip.csv", "w");
```

```
                    fprintf(fp_ip,
"Packet_Id,Device_Id,Interface_id,Source_IP,Destination_IP,Transmit
ter_IP,Receiver_IP\n");
            }
            fprintf(fp_ip,"%lld,%d,%d,%s,%s,%s,%s\n",packet-
>nPacketId,
            pstruEventDetails->nDeviceId,pstruEventDetails-
>nInterfaceId, s_ip, d_ip, t_ip, r_ip);
        fflush(fp_ip);
}
```

In NETWORK_OUT_EVENT

```
(Global Scope)                                  fn_NetSim_IP_Run()
{
    wireshark_trace.convert_sim_to_real_packet(packet,
        wireshark_trace.pcapWriterlist[pstruEventDetails->nDeviceId-1][pstruEventDetails->nInterfaceId-1],
        pstruEventDetails->dEventTime);
}

if(packet->nPacketType != PacketType_Control)
{
    IP_TO_STR(packet->pstruNetworkData->szSourceIP,s_ip);
    IP_TO_STR(packet->pstruNetworkData->szDestIP,d_ip);
    IP_TO_STR(packet->pstruNetworkData->szGatewayIP,t_ip);
    IP_TO_STR(packet->pstruNetworkData->szNextHopIp,r_ip);

    if(!fp_ip)
    {
        fp_ip=fopen("ip.csv", "w");
        fprintf(fp_ip, "Packet_Id,Device_Id,Interface_id,Source_IP,Destination_IP,Transmitter_IP,Receiver_IP\n");
    }

    fprintf(fp_ip,"%lld,%d,%d,%s,%s,%s,%s\n",packet->nPacketId,
    pstruEventDetails->nDeviceId,
    pstruEventDetails->nInterfaceId,
    s_ip,d_ip,t_ip,r_ip);
    fflush(fp_ip);
}


if(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->ppstruInterfaceList[pstruEventDetails->nInterfaceId-1]-
{
    //Call the local network protocol
    fnCallProtocol(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->ppstruInterfaceList[pstruEventDetails->
}
75 %
Output
```

In NETWORK_IN_EVENT

```
(Global Scope)                                  fn_NetSim_IP_Run()
        break;
    case NETWORK_IN_EVENT:
        {
            NetSim_PACKET* packet = pstruEventDetails->pPacket;
            if(pstruEventDetails->nInterfaceId && NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->ppstruInterfaceList[
            {
                //Call the local network protocol
                fnCallProtocol(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->ppstruInterfaceList[pstruEventDetails->
            }
            if(pstruEventDetails->pPacket==NULL)
                return 0;
            if(packet->nPacketType != PacketType_Control)
            {
                IP_TO_STR(packet->pstruNetworkData->szSourceIP,s_ip);
                IP_TO_STR(packet->pstruNetworkData->szDestIP,d_ip);
                IP_TO_STR(packet->pstruNetworkData->szGatewayIP,t_ip);
                IP_TO_STR(packet->pstruNetworkData->szNextHopIp,r_ip);

                if(!fp_ip)
                {
                    fp_ip=fopen("ip.csv", "w");
                    fprintf(fp_ip, "Packet_Id,Device_Id,Interface_ID,Source_IP,Destination_IP,Transmitter_IP,Receiver_IP\n");
                }

                fprintf(fp_ip,"%lld,%d,%d,%s,%s,%s,%s\n",packet->nPacketId,
                pstruEventDetails->nDeviceId,
                pstruEventDetails->nInterfaceId,
                s_ip,d_ip,t_ip,r_ip);
                fflush(fp_ip);
            }
            packet = pstruEventDetails->pPacket;
            //Decrease the TTL
            packet->pstruNetworkData->nTTL--;
75 %
```

7. Add the following code in IP.c file

FILE *fp_ip=NULL;

```
IP.c   DSR.c   DSR.h
(Global Scope)
    //Initiallize the ICMP
    fn_NetSim_IP_ICMP_Init();
    //Initiallize the VPN
    fn_NetSim_IP_VPN_Init();
    return 1;
}
/**
This function is called by NetworkStack.dll, whenever the event gets triggered
inside the NetworkStack.dll for IP.It includes NETWORK_OUT,NETWORK_IN and TIMER_EVENT.
*/
FILE *fp_ip=NULL;
_declspec(dllexport) int fn_NetSim_IP_Run()
{
    char s_ip[_NETSIM_IP_LEN], d_ip[_NETSIM_IP_LEN],t_ip[_NETSIM_IP_LEN],r_ip[_NETSIM_IP_LEN];

    switch(pstruEventDetails->nEventType)
    {
    case NETWORK_OUT_EVENT:
        {
            NetSim_PACKET* packet = pstruEventDetails->pPacket;
            NETWORK_LAYER_PROTOCOL nLocalNetworkProtcol;
            NETSIM_IPAddress dest;
            nLocalNetworkProtcol = fnGetLocalNetworkProtocol(pstruEventDetails);
            if(nLocalNetworkProtcol)
            {
                fnCallProtocol(nLocalNetworkProtcol);
```

8. Add the following code in fn_NetSim_IP_Run() function present in IP.c file

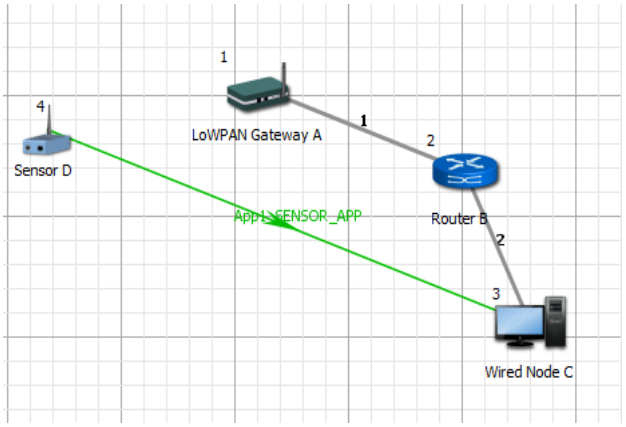
   char s_ip[_NETSIM_IP_LEN], d_ip[_NETSIM_IP_LEN],t_ip[_NETSIM_IP_LEN],r_ip[_NETSIM_IP_LEN];

```
IP.c   DSR.c   DSR.h
(Global Scope)                    fn_NetSim_IP_Run()

    //Initiallize the ICMP
    fn_NetSim_IP_ICMP_Init();
    //Initiallize the VPN
    fn_NetSim_IP_VPN_Init();
    return 1;
}
/**
This function is called by NetworkStack.dll, whenever the event gets triggered
inside the NetworkStack.dll for IP.It includes NETWORK_OUT,NETWORK_IN and TIMER_EVENT.
*/
FILE *fp_ip=NULL;
_declspec(dllexport) int fn_NetSim_IP_Run()
{
    char s_ip[_NETSIM_IP_LEN], d_ip[_NETSIM_IP_LEN],t_ip[_NETSIM_IP_LEN],r_ip[_NETSIM_IP_LEN];

    switch(pstruEventDetails->nEventType)
    {
    case NETWORK_OUT_EVENT:
        {
            NetSim_PACKET* packet = pstruEventDetails->pPacket;
            NETWORK_LAYER_PROTOCOL nLocalNetworkProtcol;
            NETSIM_IPAddress dest;
            nLocalNetworkProtcol = fnGetLocalNetworkProtocol(pstruEventDetails);
            if(nLocalNetworkProtcol)
            {
                fnCallProtocol(nLocalNetworkProtcol);
100 %
```

9. Right click on IP in Solution Explorer and select rebuild.
10. Upon rebuilding, libIP.dll will get created in the path...\simulation\DLL
11. Now copy the libIP.dll from this DLL folder and paste it in NetSim bin folder present in the NetSim installed directory. The NetSim install directory would look something like < C:\Program Files (x86)\NetSim Standard\bin>
12. Note that there exists a libIP.dll in this bin folder. This is the default file being shipped with NetSim. The user is replacing this file with the newly built file.
13. Therefore, take care to rename the original libIP.dll file, so that it isn't lost. For example, you may rename it as libIP_default.dll
14. Open NetSim and create a Network Scenario in NetSim IOT Network with 1 sensor, 1 LoWPAN Gateway, 1 Router and 1 Wired Node. Create 1 Sensor application from Sensor D to Wired Node C.



15. Run the simulation.
16. Now you can see ip.csv file in the bin folder of NetSim installed directory. It looks like

	A	B	C	D	E	F
1	Packet_Id	Device_Id	Interface_id	Source_IP	Destination_IP	Transmitter_IP
2	1	4	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	21AF:435D:7F02:92B1:4219:FCD2:C7A3:3A14	21AF:435D:7F02:92B1:4219:AF26:
3	1	1	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	21AF:435D:7F02:92B1:4219:FCD2:C7A3:3A14	21AF:435D:7F02:92B1:4219:AF26:
4	2	4	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
5	2	1	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
6	2	1	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2
7	2	2	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2
8	2	2	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.3.1.1
9	2	3	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.3.1.1
10	3	4	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
11	3	1	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
12	3	1	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2
13	3	2	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2
14	3	2	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.3.1.1
15	3	3	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.3.1.1
16	4	4	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
17	4	1	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
18	4	1	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2
19	4	2	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2
20	4	2	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.3.1.1
21	4	3	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.3.1.1
22	5	4	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
23	5	1	1	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	21AF:435D:7F02:92B1:4219:AF26:
24	5	1	2	21AF:435D:7F02:92B1:4219:AF26:0551:EE95	11.3.1.2	11.1.1.2

17. Users can understand how the IP addresses are changing from IPv6 to IPv4 and vice versa with the help of ip.csv file.
18. Sensor and LoWPAN gateways 1st interface follows IPv6 addressing.
19. LoWPAN gateways 2nd interface, Router, Wired Node follows IPv4 addressing.
20. As shown in the above image, users can identify the changing of IP addresses from source to destination.