SUMMER COURSE IOT

PROJECT DOCUMENTATION

SMART TRASH POOL SYSTEM(STPS)

SI.NO	Name	SRN	Section
1	**CHINMAYA R**	**PES2UG21CS144**	**C**
2	**CHANDANA DONGRE**	**PES2UG21CS134**	**C**

Open the Cooja simulation and create the motes according the coordinates required

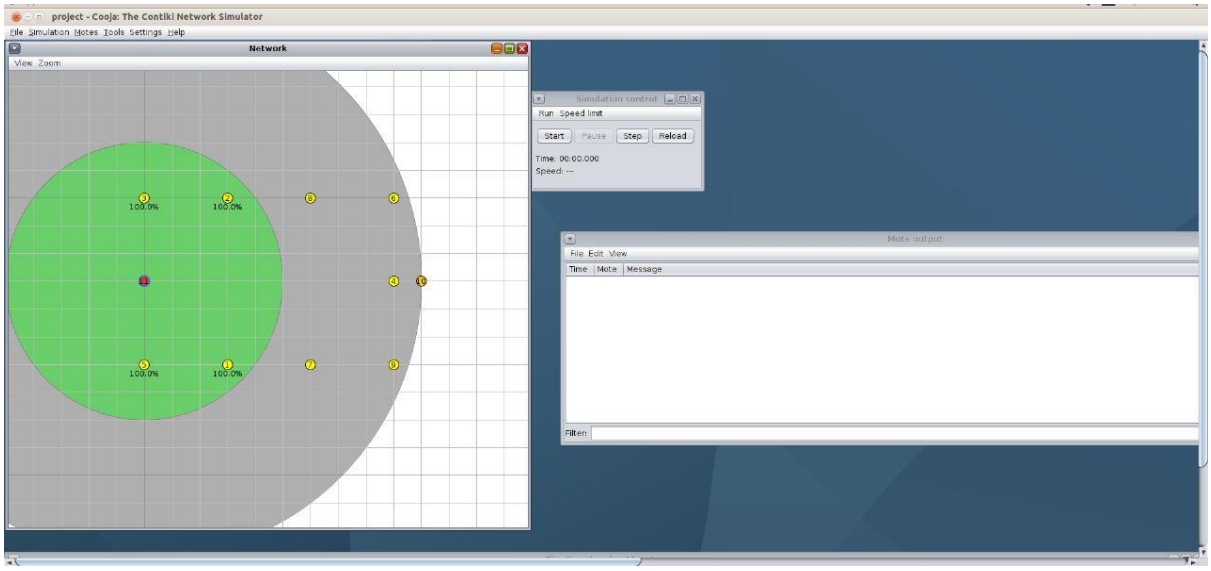
In this model we use the following key:

1. Dust bins using 9 sky-websense Mote (sky 1-9)

2.Vehicle using 1 mobile sender (sky 11)

3. Border-router (sky 10)

After generating motes needed start the Simulation.



This the mote output after starting the simulation:

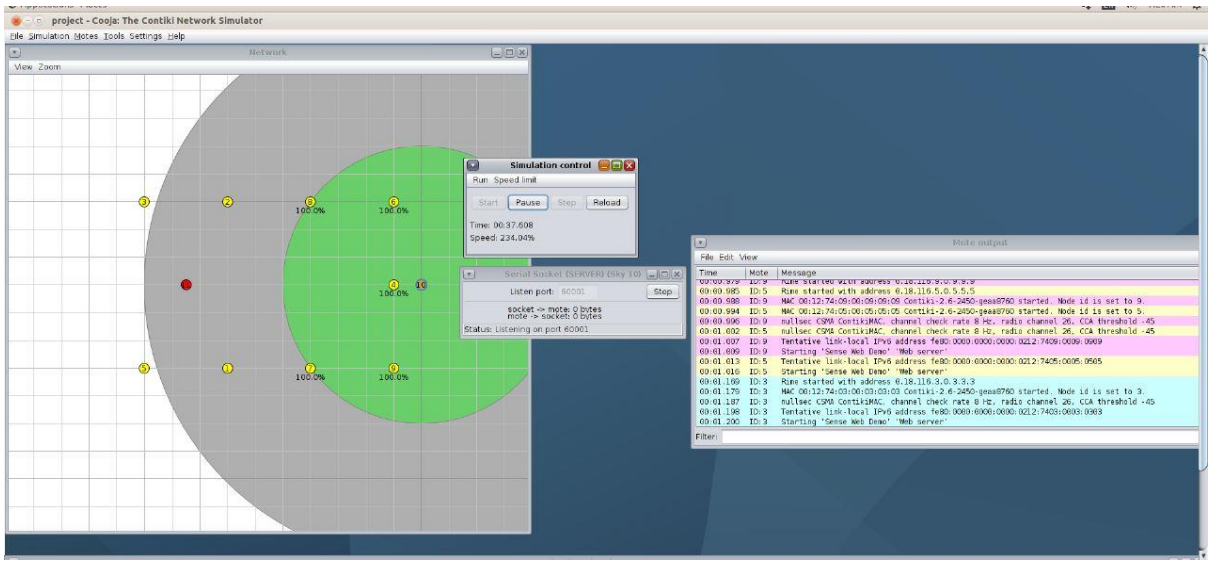
Edit the skywebsense.c file to add mass parameter. Changed code shown below:

```
}
static int
get_mass(void){
   int res = (rand() % 2 + 1);
    int weight = res * 9.8;
return weight;
}
/*---------------------------------------------------------------------------*/
static const char *TOP = "<html><head><title>Contiki Web Sense</title></head><body>\n";
static const char *BOTTOM = "</body></html>\n";
/*---------------------------------------------------------------------------*/
/* Only one single request at time */
static char buf[256];
static int blen;
#define ADD(...) do {                                                   \
    blen += snprintf(&buf[blen], sizeof(buf) - blen, __VA_ARGS__);      \
  } while(0)

static
PT_THREAD(send_values(struct httpd_state *s))
{
  PSOCK_BEGIN(&s->sout);

  SEND_STRING(&s->sout, TOP);

  if(strncmp(s->filename, "/index", 6) == 0 ||
     s->filename[1] == '\0') {
    /* Default page: show latest sensor values as text (does not
       require Internet connection to Google for charts). */
    blen = 0;
        int temp = get_mass();
    ADD("<h1>Current readings</h1>\n"
        "Weight %d Kg",
        temp);
    if(temp>15){
        ADD("<br>Alert!!! Bin's about to fill");
}
```

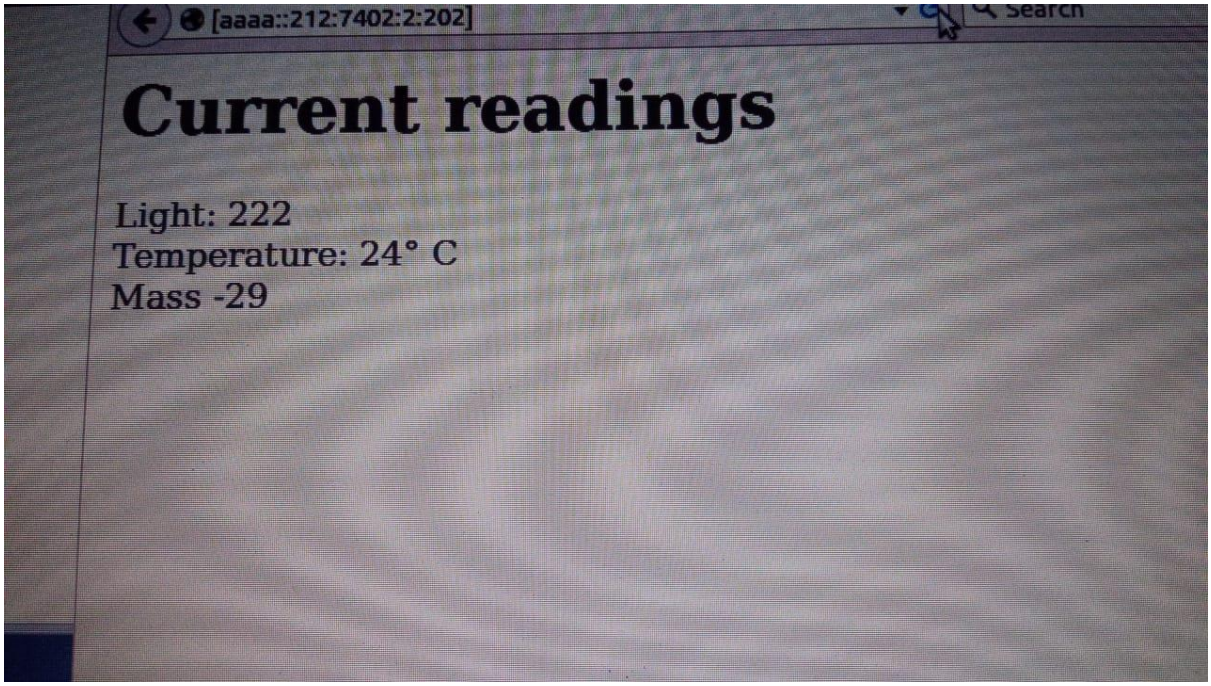
Make connect the border router in Contiki/examples/ipv6/border-router directory using command make connect-router-cooja and open the serial socket SERVER tool and start on the listening port 60001. It configures and gives the server IPv6 addresses as shown below:

```
user@instant-contiki:~/contiki/examples/ipv6/rpl-border-router$ make connect-router-cooja
TARGET not defined, using target 'native'
sudo ../../../tools/tunslip6 -a 127.0.0.1 aaaa::1/64
[sudo] password for user:
slip connected to ``127.0.0.1:60001''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
          inet6 addr: aaaa::1/64 Scope:Global
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
 aaaa::212:740a:a:a0a
 fe80::212:740a:a:a0a
```

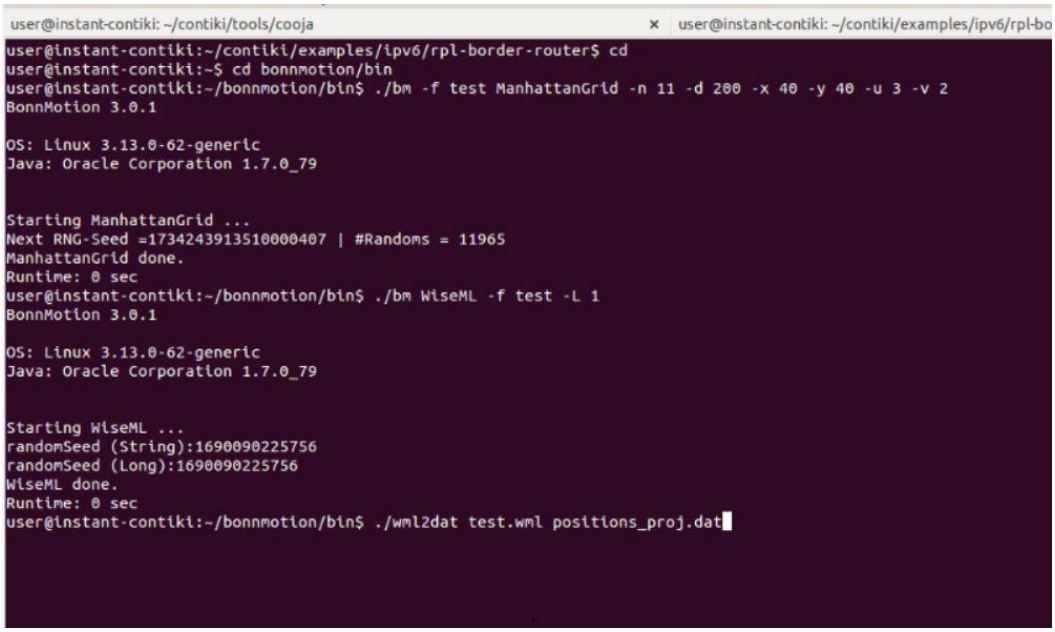
Switch on the server port 60001



Using the server IPv6 addresses type the ip after the route has been determined to get the readings. We get the output on web page (ContikiRPL) and also get the result of each sky-websense(Contiki Web Sense) Mote as shown below:



Design a mobility model arrange the coordinates as required and save the positions_proj.dat file



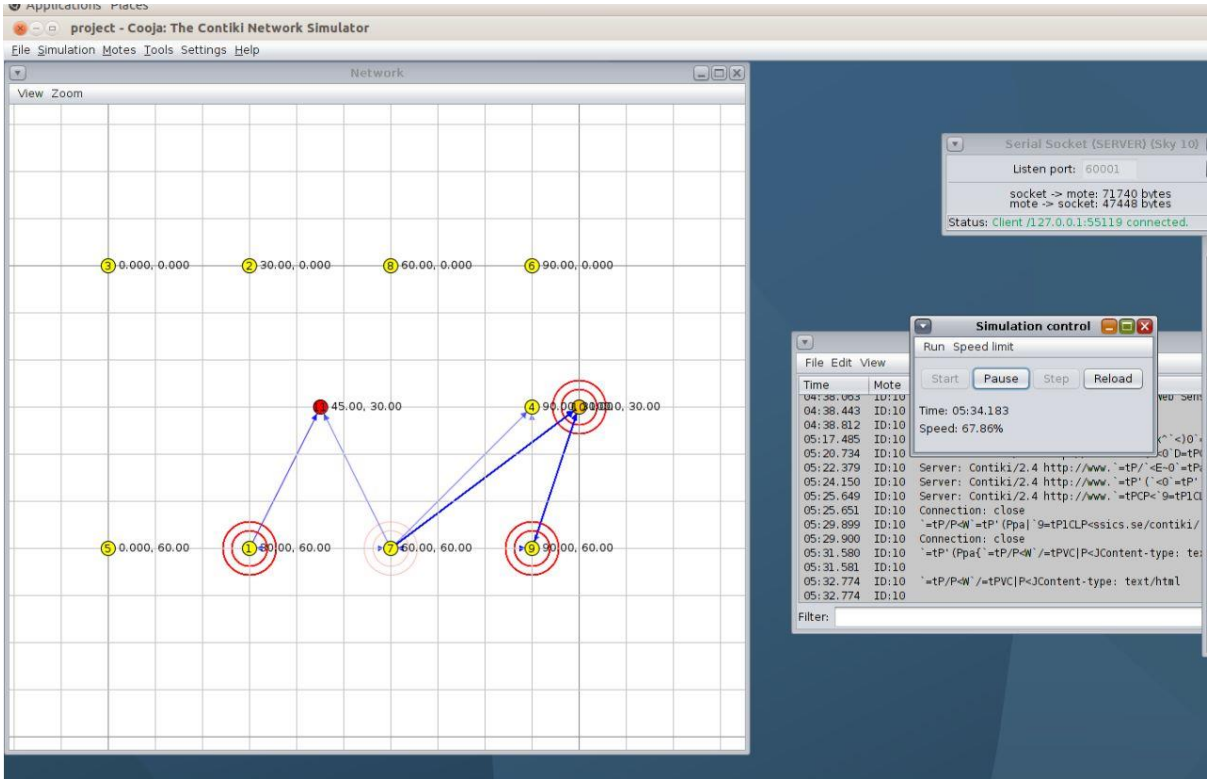
One Mobile sender:

sky-websense.c × positions_proj.dat ×

```
0 0 30 60
1 0 30 0
2 0 0 0
3 0 90 30
4 0 0 60
5 0 90 0
6 0 60 60
7 0 60 0
8 0 90 60
9 0 100 30
10 0 0 30
0 1 30 60
1 1 30 0
2 1 0 0
3 1 90 30
4 1 0 60
5 1 90 0
6 1 60 60
7 1 60 0
8 1 90 60
9 1 100 30
10 1 15 30
0 2 30 60
1 2 30 0
2 2 0 0
3 2 90 30
4 2 0 60
5 2 90 0
6 2 60 60
7 2 60 0
8 2 90 60
9 2 100 30
10 2 15 30
0 3 30 60
1 3 30 0
2 3 0 0
3 3 90 30
4 3 0 60
5 3 90 0
6 3 60 60
7 3 60 0
```

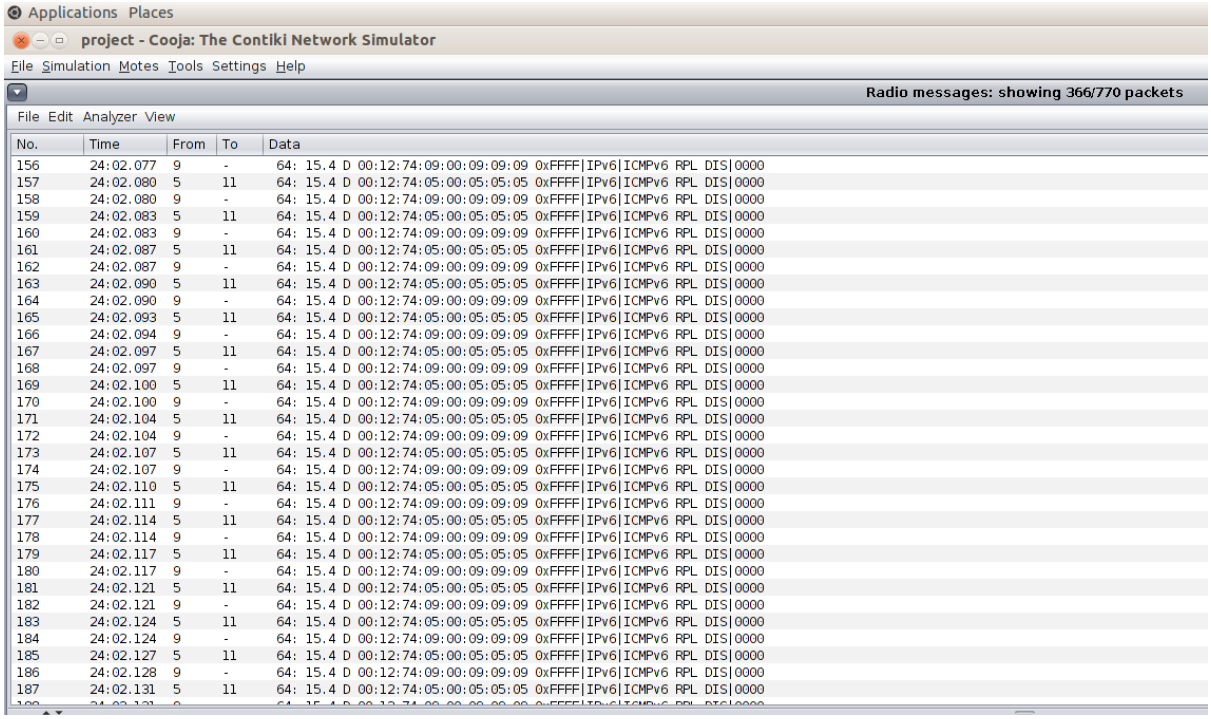
Import the mobility model to the simulation And the final output will look like this:

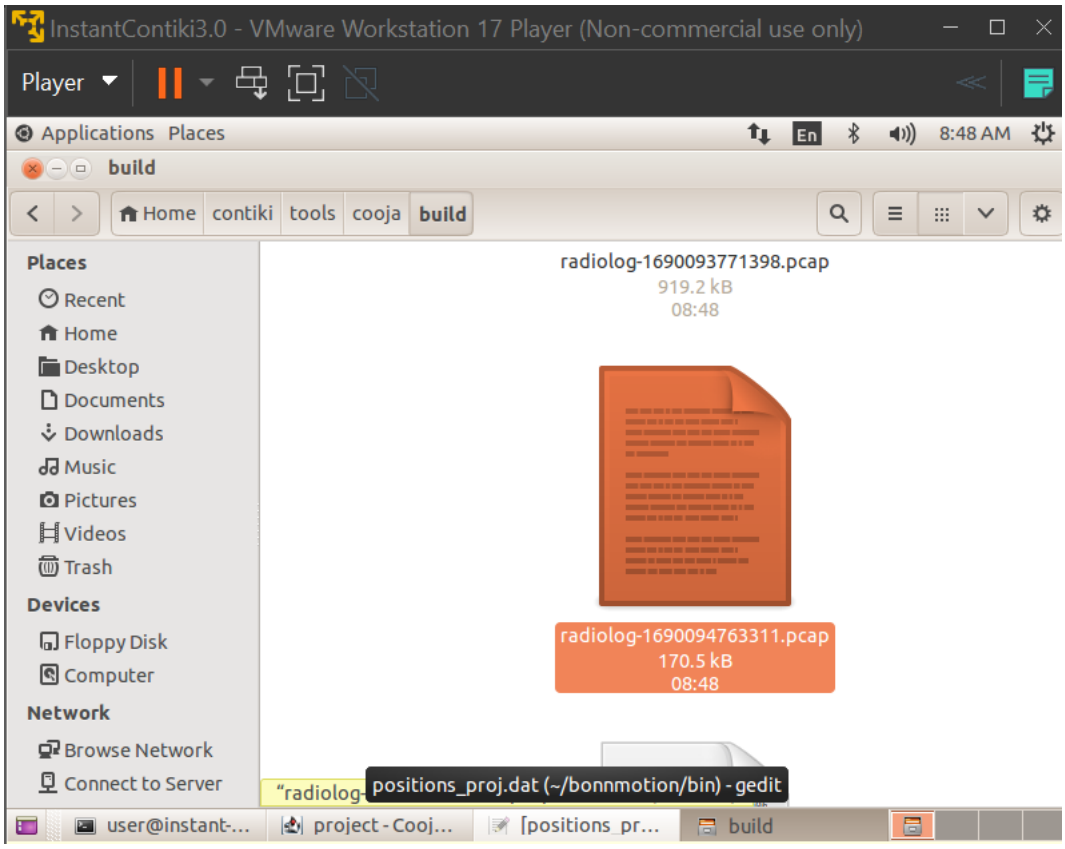
After starting the simulation mote 11 move according to their mobility model



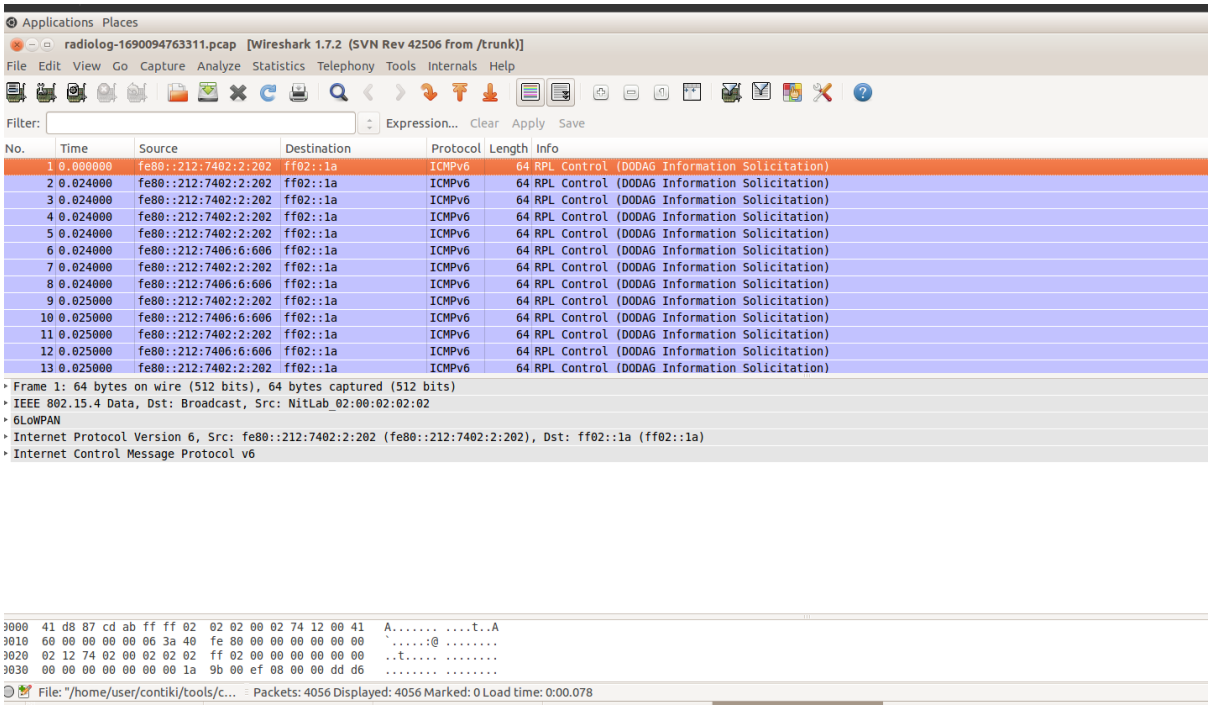
Generating PCAP using Radio messages:

Open radio messages choose the 6lopan analyser and run the simulation and after 15 minutes stop the simulation and we can open the pcap file to start the analysis of the packets along with the protocols used

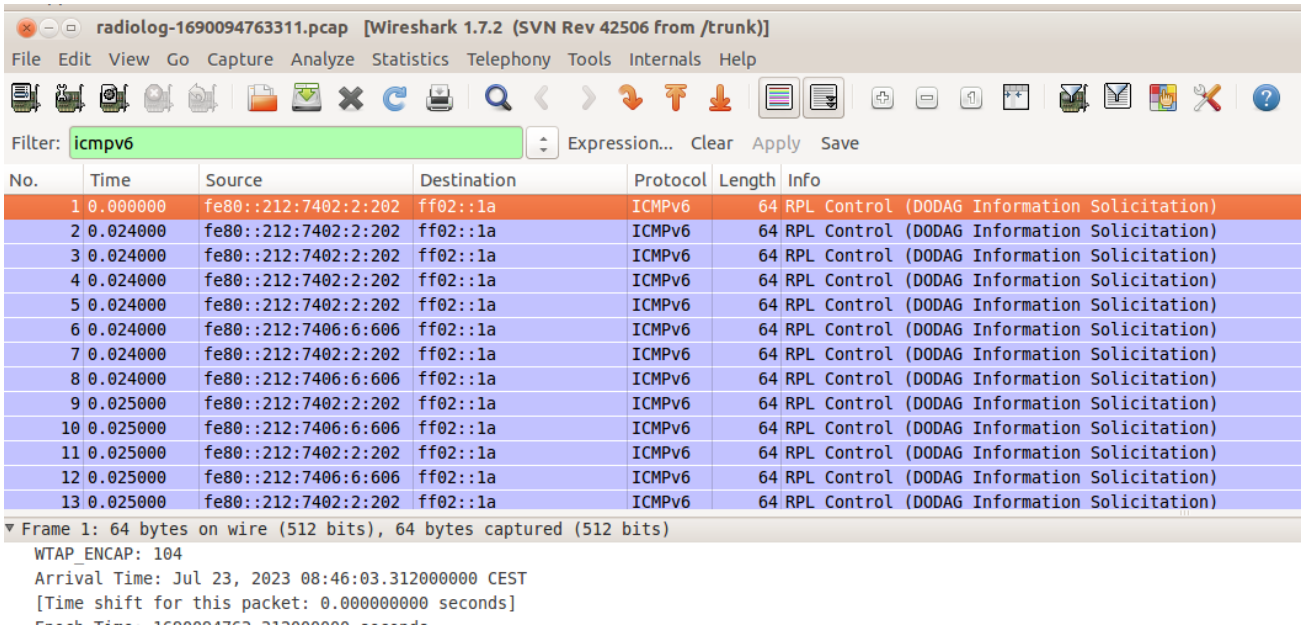




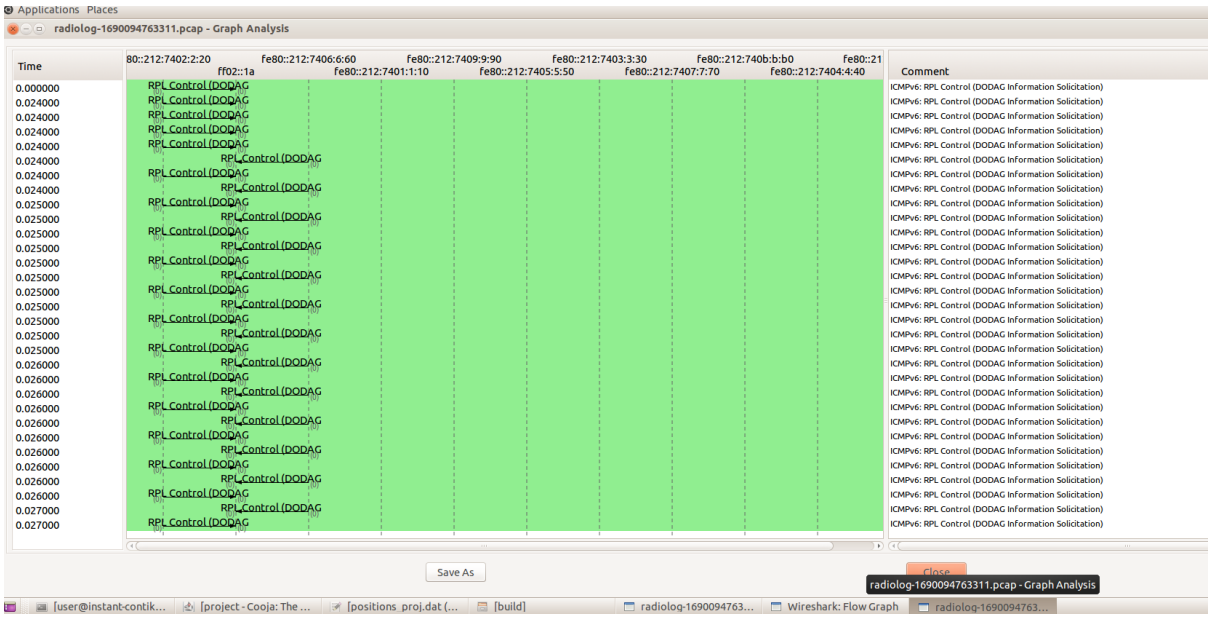
Initial the wireshark looks like this:



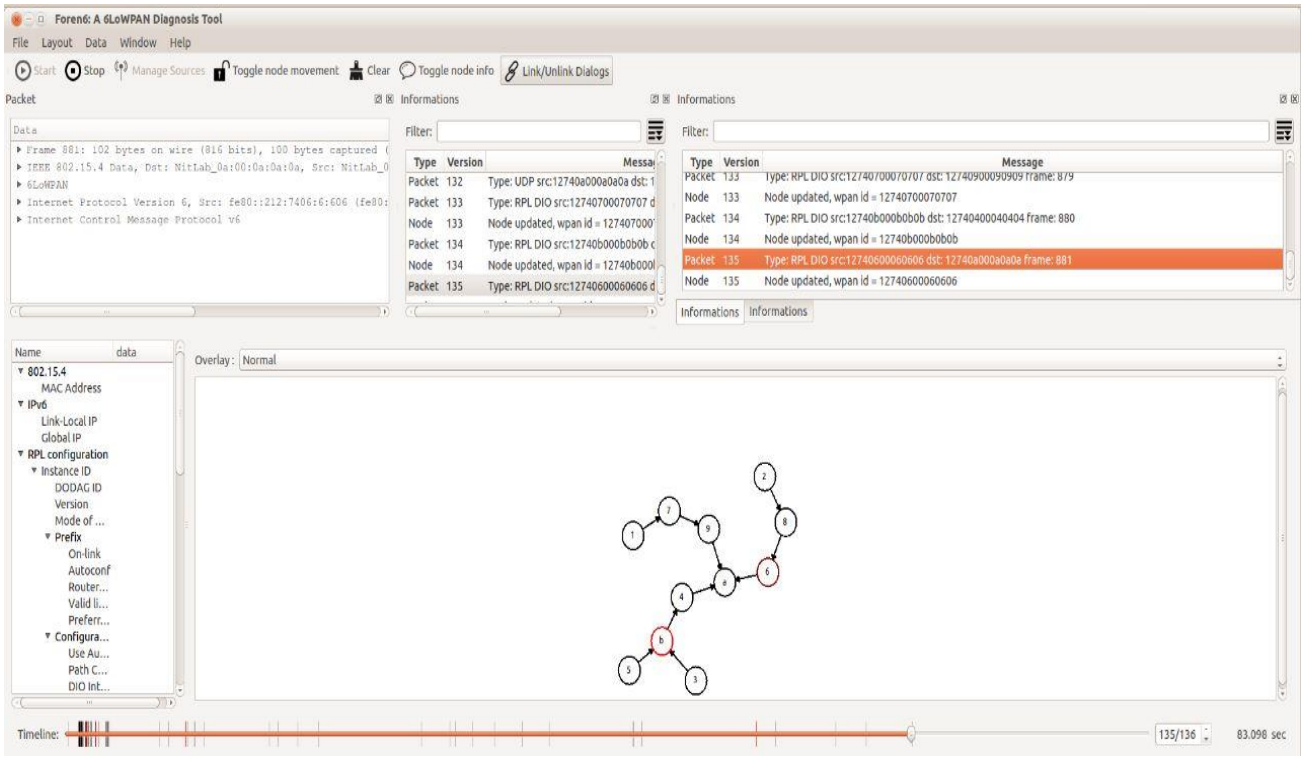
Using the filter of the icmpv6 protocol to filter out the packets:



In Wireshark we can check the flow graph for DODAG using statistics option:



Now we analyse the same PCAP file in foren6 which allows more interactive form of analysis and is easy to understand. The details are available at a glance in the different tabs you can select any packet and the data is available in the other tab. You can also choose the time from timeline and select the red mote to see the warnings or lost packet from that particular mote


Foren6: A 6LoWPAN Diagnosis Tool
File Layout Data Window Help
Start Stop Manage Sources Toggle node movement Clear Toggle node info Link/Unlink Dialogs
Packet
Informations
Filter:
Type Version Message
Packet 132 Type: UDP src:12740a000a0a0a dst: 1
Packet 133 Type: RPL DIO src:12740700070707 d
Node 133 Node updated, wpan id = 127407000
Packet 134 Type: RPL DIO src:12740b000b0b0b c
Node 134 Node updated, wpan id = 12740b000
Packet 135 Type: RPL DIO src:12740600060606 d
Informations
Filter:
Type Version Message
Node 133 Node updated, wpan id = 12740700070707
Packet 134 Type: RPL DIO src:12740b000b0b0b dst: 12740400040404 frame: 880
Node 134 Node updated, wpan id = 12740b000b0b0b
Packet 135 Type: RPL DIO src:12740600060606 dst: 12740a000a0a0a frame: 881
Node 135 Node updated, wpan id = 12740600060606
Informations
Informations
Name data
802.15.4
MAC Address
IPv6
Link-Local IP
Global IP
RPL configuration
Instance ID
DODAG ID
Version
Mode of ...
Prefix
On-link
Autoconf
Router...
Valid li...
Preferr...
Configura...
Use Au...
Path C...
DIO Int...
Overlay : Normal
Timeline:
135/136
83.098 sec