**Project-2**

**RPL Attacks**

**Prepared By : Lohith Reddy Kalluru and Shubham Makwana**

**Attack:**

RPL is a routing protocol for low-power and lossy constrained node networks.It creates a tree-like routing topology called the destination-oriented directed acyclic graph (DODAG), which is in the direction towards one or more nodes known as root node or sink node. RPL protocols are used with resource constraint nodes.

• **DODAG Information Object (DIO)**:it stores information including rank of a node, RPL Instance, the IPv6 address of the root or sink and so on.

• **Destination Advertisement Object (DAO)**: it consists of information that can used for downward traffic towards child nodes.

• **DODAG Information Solicitation (DIS)**: Used by nodes to request graph related information from the neighboring nodes.

• **Destination Advertisement Object Acknowledgement (DAOACK)**: Sent by a DAO recipient in response to a DAO message.

Whenever a new node enters into a rpl network, it starts sending a DIS message to all nodes and waits for DIO message to be received whcih contains details regarding node id and objective code point. The DIO messages are broadcasted at particular intervals based on the trickle algorithm. The node on recieving the info , calculates the rank . and based on that it selects a parent . To send a message downwards , node should send a DAO message containing the routable fixes up the tree.To prevent the loops ,RPL does not allow the data going in down direction and sent from a desecendent . RPL uses two header options which

Is flow of direction (O)and rank error(R) . Rank error is a flag set when there is a mismatch in the rank of sender and direction of flow.

When a malicious node is introduced into the network ,it can manipulate the header-options used by RPL to track DODAG. Once the header-options are manipulated , the malicious node can cause denial of service attacks ,drain power from nodes and also can target certain nodes by creating black hole.

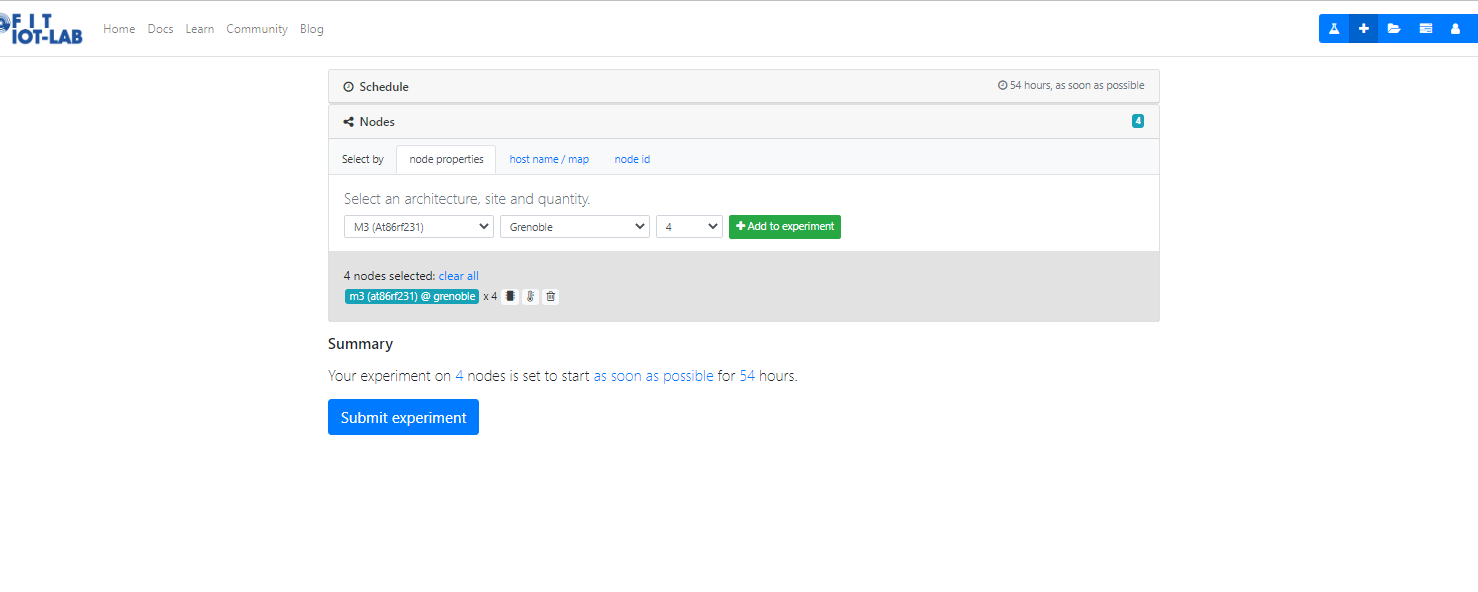
**DODAG attack using fit iot lab**

Here we have implemented a flood attack on a real testbed using the m3 node ( at86rf231) at site=grenoble. We flashed 1 up-UDP-server and 3 RPL-UDP-clients. Here we have considered a flooding attack. The power consumption is shown in the figure. We modify the DIS-related code to define multiple DIS constants which causes the node to send DIS messages repeatedly. We also modify the timers to send the multiple DIS messages. We start a serial -aggregator and power consumption monitor which is used to capture the packets and also measure the power consumed at each node.

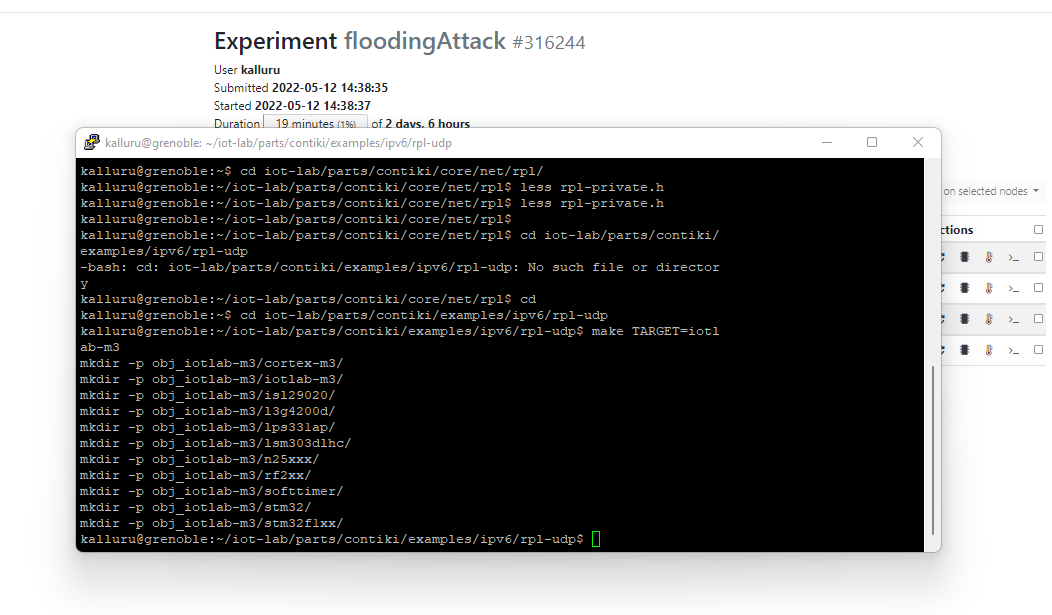
**Process**

1. **Implementation of Flooding Attack (DDOAG attack)**

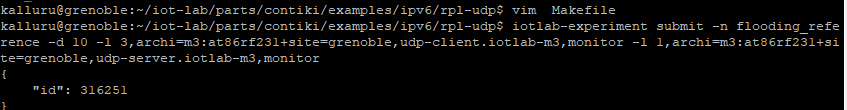
**Created 4 nodes with grenoble m3**



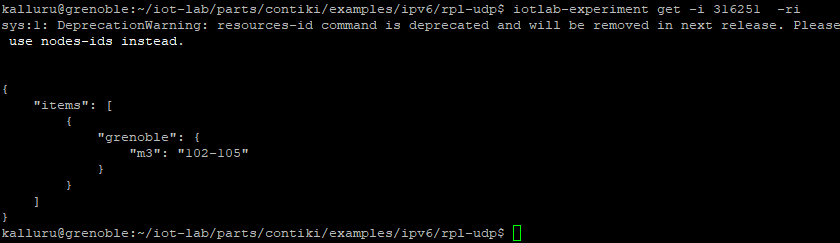
**Compile firmware for target iotlab-m3**

****

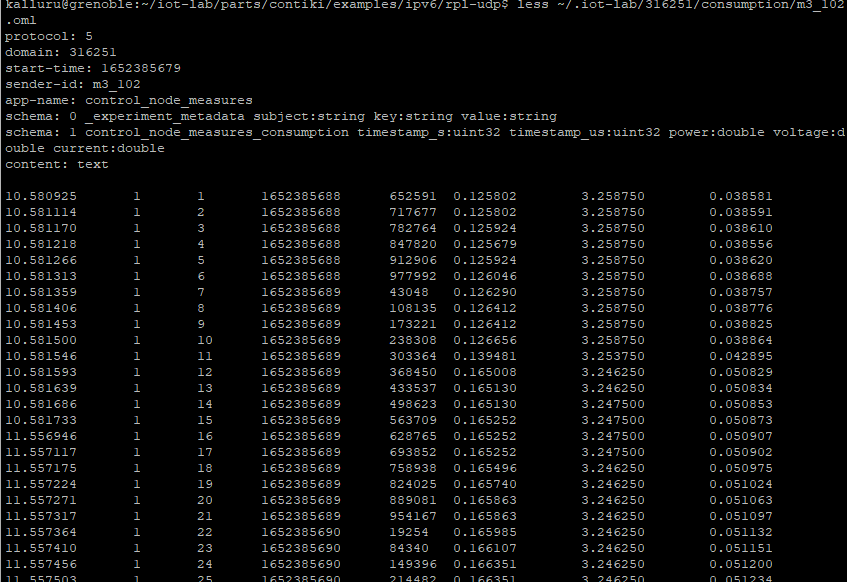
**Consumption monitoring**



**iotlab-experiment get -i 316251 -ri to check the nodes**

****

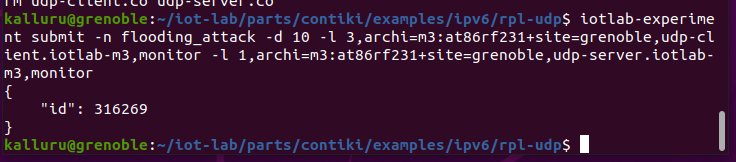
**Measured consumption data is shown below without running the flood attack (m3-102)**

****

**Compile the firmware for flooding attack code**

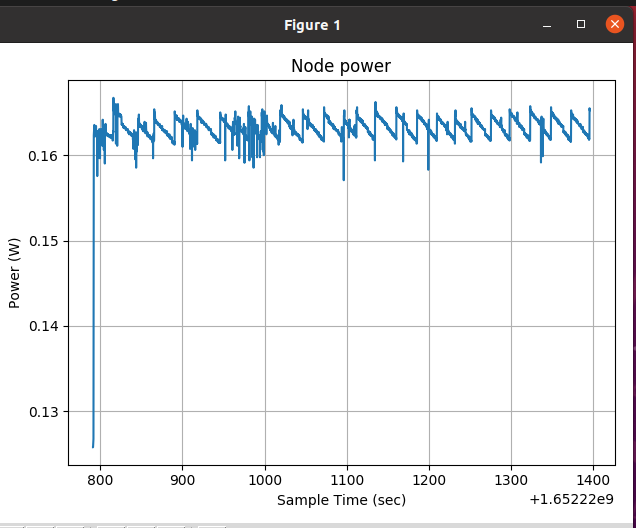
****

**Create the experiment by flashing with malicious nodes with flooding attack**

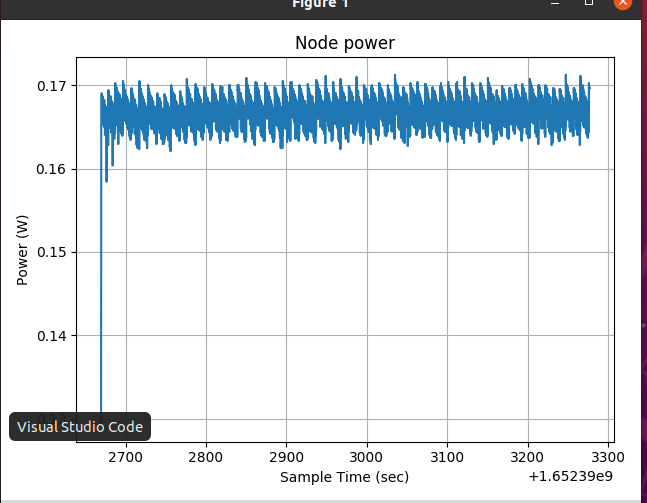
****

**RESULTS**

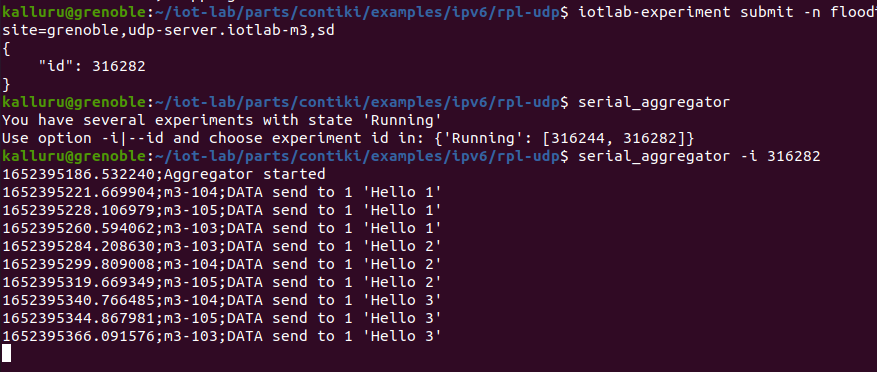
**Before Intrusion power consumed by node:**

****

**When the graph is plotted for one of the nodes we observe that power consumption has been increased**

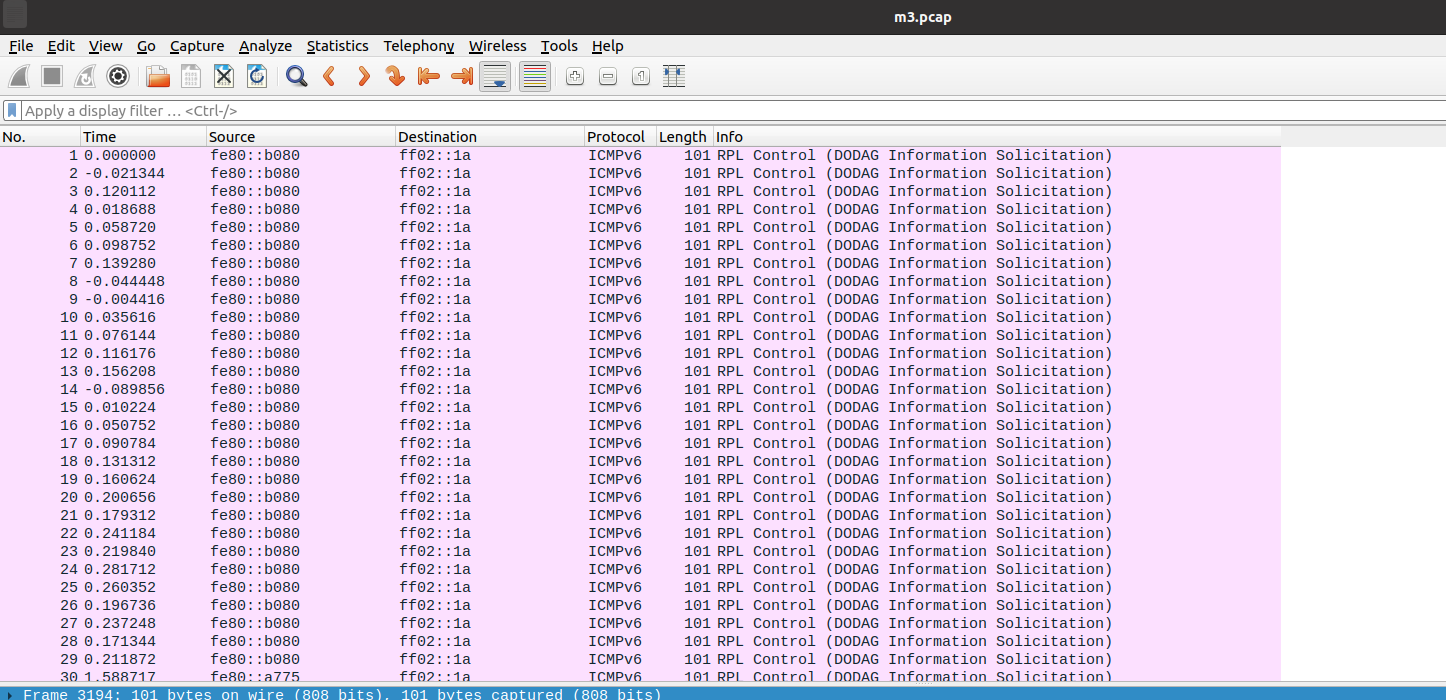
****

**The below output shows the various datapackets captured by monitoring**

****

**Wireshark capture is shown below .**

**It indicates the various dodag information solicitation**

****

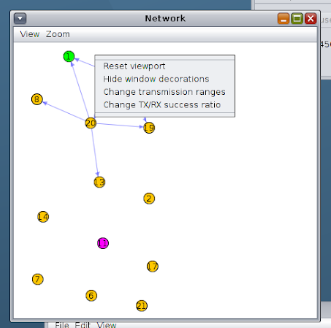
**2. DoDAG/Blackhole attack using Contiki and cooja simulator(by modifying ranks and flow of direction of packets)**

**Process:**

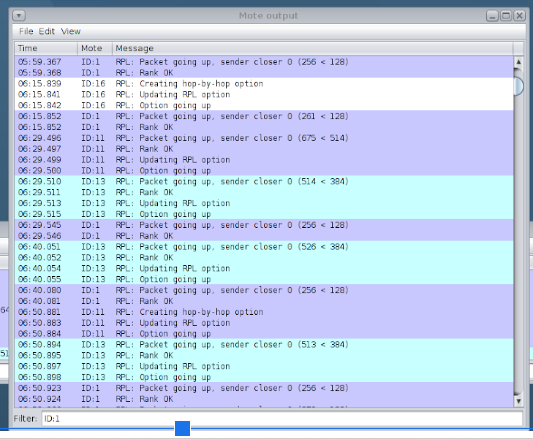
:

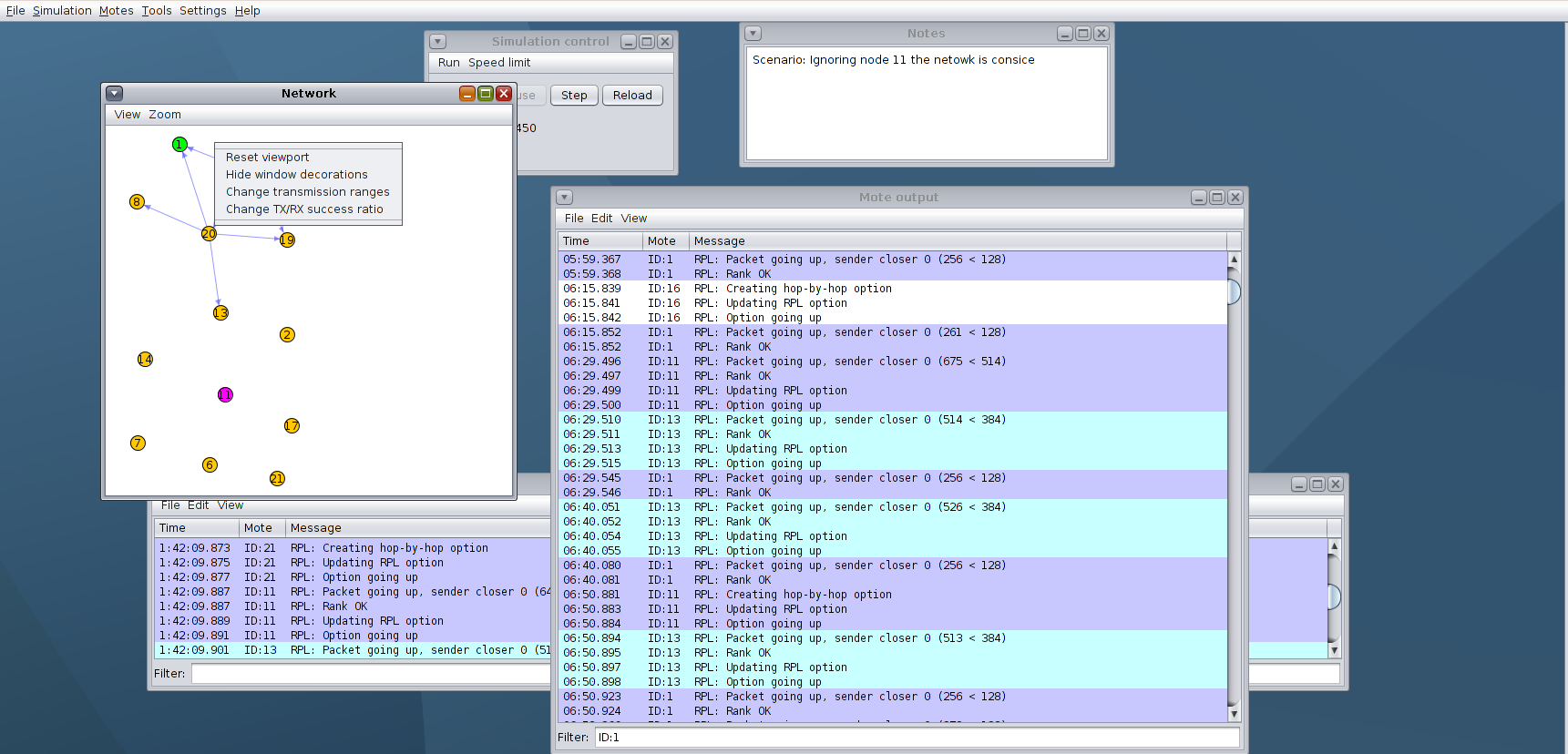
We created a network of 11 motes out of which one is an intrusion udp server and others are udp clients. In this particular scenario , the intrusion node tries to attack the parent by transmitting packets that have both the O and R flag set. When this happens , the parent node drops the packet which leads to trickle timer being reset. Due to this , control messages are broad casted more commonly which causes loss of packets .All this can be observed in the packets captured

**Blue node indicates the sink node and 1 is the intrusion node**



**Packets captured are shown below**

****

****

**DEFENCE:**

Coming to mitigation we had used a algorithm where whenever we are receiving the packet with inconsistency with o set and R set , the packet is dropped ,but the trickle timer is not reset . But we set up a threshold to prevent the trickle timer from resetting.

if (O = 1 and ri < rj ) or (O = 0 and ri > rj ) then if R = 1

then count + +

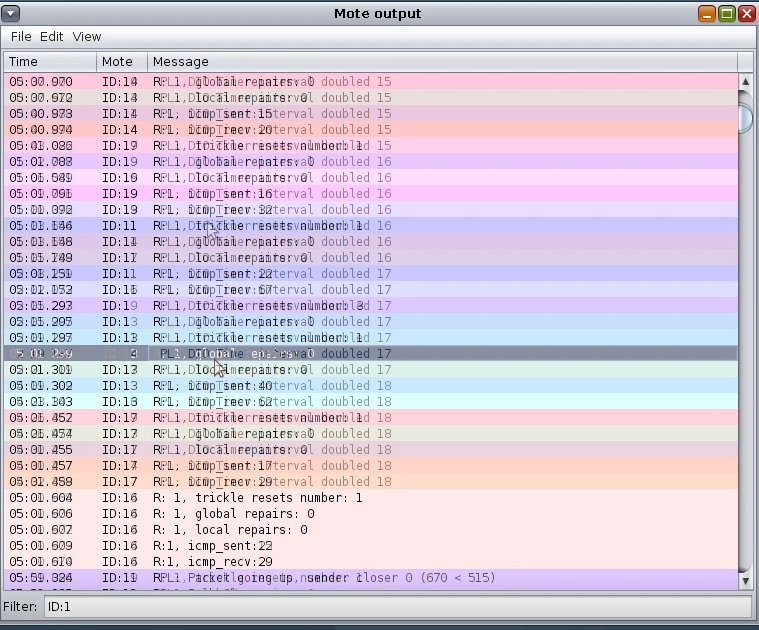
drop(packet)

if count < threshold then

Reset timer

In future,

We can use a controller to verify the nodes for a lot of attacks. One scenario is that , the intrusion node is detected using kMeans algorithm.The node network is organised in such a way that the intrusion nodes donot have child nodes.

****

**Challenges and Interesting aspects**

We had a interesting experience when searching for the attacks. The fit iot lab provides an enriched way of accessing various parameters. Using instant contiki we had a lot of dependency issues when running programs . What we find interesting is that the references in paper are easily available but the actual baseline implementations are very scarce from our perspective.The topic related to rpl routing is vey interesting.

**References:**

**[1]** [**https://github.com/iot-lab/iot-lab/wiki/Control-Node-Sniffer**](https://github.com/iot-lab/iot-lab/wiki/Control-Node-Sniffer)

**[2] https://iot-lab.github.io/docs/tools/radio-monitoring/**

**[3] https://www.iot-lab.info/legacy/dev-center/index.html**

**[4] https://www.iot-lab.info/legacy/tutorials/contiki-public-ipv6-m3/index.html**

**[5] https://www.iot-lab.info/legacy/tutorials/contiki-coap-m3/index.html**

**[6] https://www.iot-lab.info/legacy/tutorials/contiki-private-ipv6-m3/index.html**

**[7] https://anrg.usc.edu/contiki/index.php/Network\_Stack**

**[8] Solapure, Sharwari & Kenchannavar, Harish. (2019). RPL And COAP Protocols,**

**Experimental Analysis for IOT: A Case Study. International Journal of Ad hoc, Sensor**

**& Ubiquitous Computing. 10. 01-15. 10.5121/ijasuc.2019.10201.**

**[9] https://github.com/contiki-ng/contiki-ng**

**[10] https://github.com/contiki-ng/contiki-ng/tree/develop/test**

**[11]** [**RPL\_DODAG\_Visualization\_v13.1/Documentation at main · NetSim-TETCOS/RPL\_DODAG\_Visualization\_v13.1 (github.com)**](https://github.com/NetSim-TETCOS/RPL_DODAG_Visualization_v13.1/tree/main/Documentation)

**[12]** [**Softwarized & Wireless Networks Research Group (github.com)**](https://github.com/SWNRG)

: