# DIS Flooding Attack in IOT Networks Running RPL

**Software Recommended**: NetSim Standard v13.2, Visual Studio 2022

**Reference**: Y. Mai, F. M. Rodriguez and N. Wang, "CC-ADOV: An effective multiple paths congestion control AODV," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 1000-1004.

**Secure URL for the GitHub repository:**
https://github.com/NetSim-TETCOS/RPL_DIS_Flooding_v13.2/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and setup the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

**Introduction:**

In RPL, DIS messages are used by nodes to join the network. A node sends a DIS message to its neighbour nodes to request the routing information so that it may join the existing DODAG. Thus, a new node continuously transmits DIS messages with a fixed interval until it receives a DIO message from any neighbour node. Once a node receives a DIO message, it stops transmitting DIS messages and joins the network by sending DAO to the solicited node.

A malicious node can utilize this feature to degrade the network performance by choosing different DIS transmission interval for periodically transmitting DIS messages to its neighbouring nodes; this is called a DIS flooding attack. This leads to an increase in the network's control packet overhead and power consumption.

**Implementation in RPL (for 1 sink)**

- In RPL the transmitter broadcasts the DIO during DODAG formation.
- The receiver on receiving the DIO from the transmitter updates its parent list, sibling list, rank and sends a DAO message with route information.
- Malicious node upon receiving the DIO message instead of joining existing DODAG it just Drops DIO and frequently transmits DIS messages. Which forces normal nodes to reset their trickle timers and flood the network with DIO messages.

A file Malicious.c is added to the RPL project.

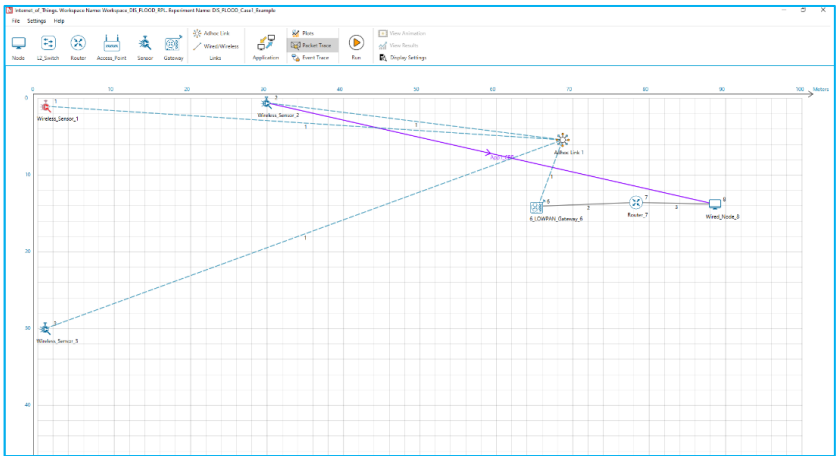
The file contains the following functions

1. **fn_NetSim_RPL_MaliciousNode( );/**/This function is used to identify whether a current device is malicious or not in-order to establish malicious behaviour.
2. **rpl_drop_msg( );**//This function is used to drop the DIO messages received by the malicious nodes instead of replying with a DAO message.

You can set any sensor as malicious, and you can have more than one malicious node in a scenario. Device id's of malicious nodes can be set inside the fn_NetSim_RPL_MaliciosNode() function.

**Example**

- The Workspace_DIS_FLOOD_RPL comes with a sample network configuration that is already saved.
- To open this example, go to Your work in the Home screen of NetSim and click on the DIS_FLOOD_Case1_Example from the list of experiments.
- The saved network scenario consists of
  - 3 Sensor
  - 1 6 _LOWPAN Gateway
  - 1 Routers
  - 1 wired node



- Application Properties

| Application Properties | |
|---|---|
| **Source ID** | 2 |
| **Destination ID** | 8 |
| **Wireless Sensor(Malicious Node)** | 1 |
| **Transport Protocol** | UDP |
| **Other** | Default |

- Link Properties (Adhoc Link1):
  - Channel Characteristics – Pathloss Only
  - Pathloss Model – Log Distance
  - Pathloss Exponent – 3.5
- Set Network Layer routing Protocol to RPL in both sensor and 6_LOWPAN_Gateway
- Device Properties: Go to Sensor Properties -> Network Layer -> DIS_Interval -> 10ms.
- Run the Simulation for 100 seconds.

**Results and Discussion :**

1. View the packet animation. You will find that the malicious node (Device id 1) even after receiving DIO from neighbor nodes instead of joining existing DODAG it just Drops DIO and frequently transmits DIS messages.
2. This will have a direct impact on the Application Throughput and Delay which can be observed

in the Application Metrics table present in NetSim Simulation Results window.

### Case 1: Application throughput Vs. Application generation rate

We fix the DIS interval to 10 milli seconds and vary the application generation rate to see the impact of DIS flooding on the network performance.

- For **With_DIS,** Run the simulation of the imported workspace.
- For **Without_DIS,** Reset the binaries of the imported workspace and run the simulation.
- To reset the binaries, go to your Work -> Source Code -> Reset Binaries
- To recheck the impact of the network performance **With_DIS,** Rebuild the **RPL** project in source code, Go to your Work -> Source Code -> Open Source Code

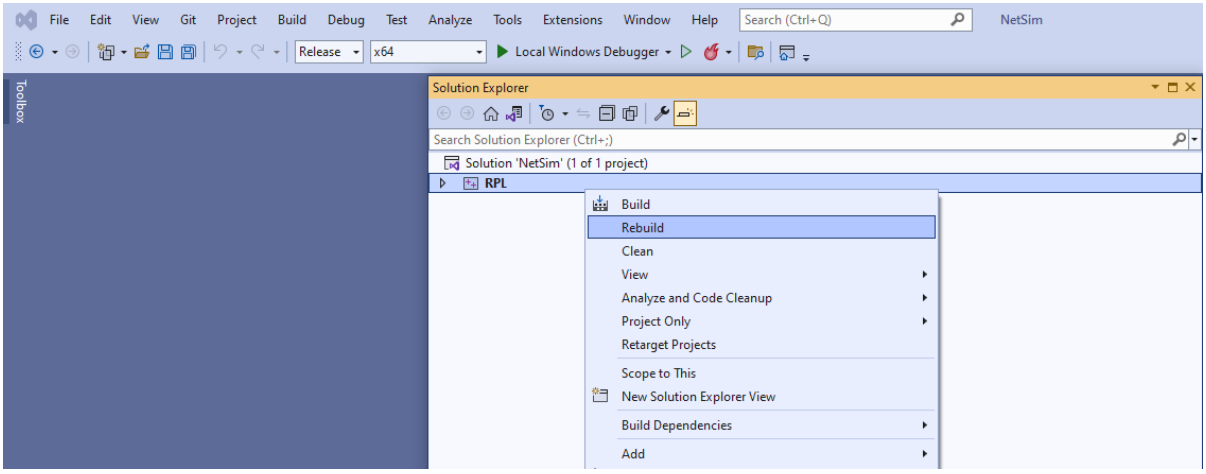
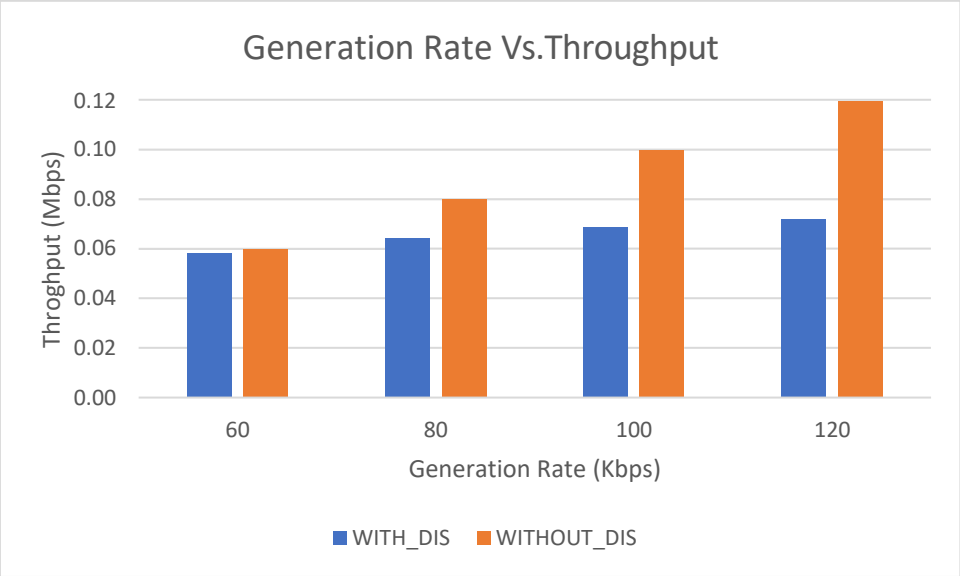
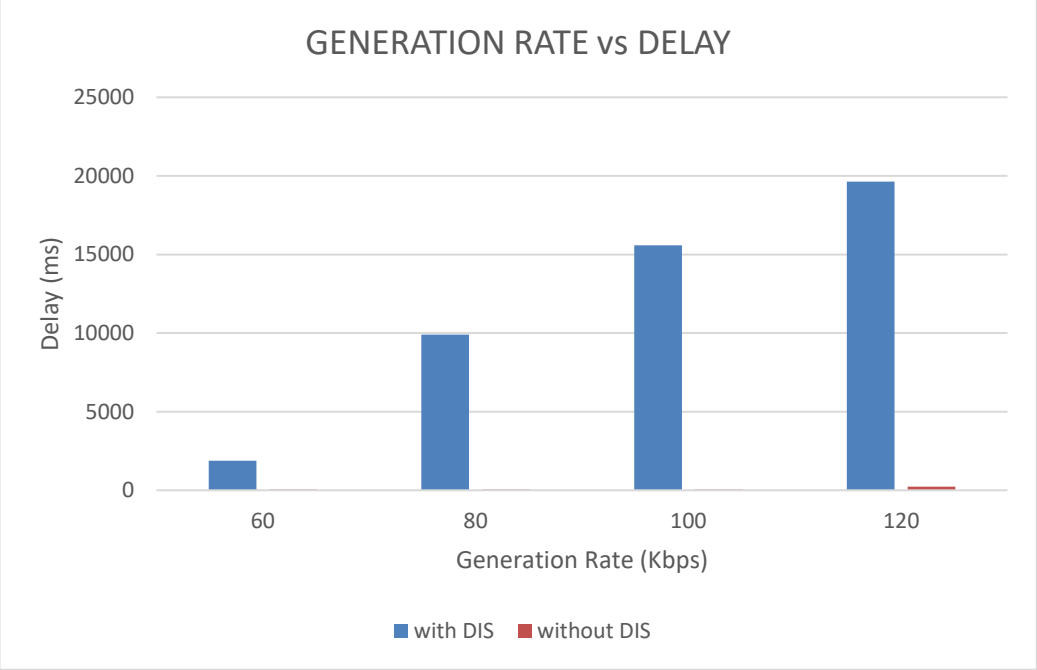



**Figure 1**: Source code of current Workspace

| | Throughput (Mbps) | | Delay (ms) | |
|---|---|---|---|---|
| **Generation Rate (Kbps)** | **With_DIS** | **Without_DIS** | **With_DIS** | **Without_DIS** |
| **60** | 0.06 | 0.06 | 1880.97 | 51.80 |
| **80** | 0.06 | 0.08 | 9899.92 | 51.75 |
| **100** | 0.07 | 0.10 | 15577.11 | 51.76 |
| **120** | 0.07 | 0.12 | 19645.16 | 239.45 |

This can be further understood with the help of following plots:



We can observe that the application throughput decreases in case of DIS flooding when compared with the usual simulations for various application traffic generation rates.

Delay is comparatively high in case of DIS flooding and increases with the increase in generation rate. This is because the nodes are busy receiving and responding to DIS messages from malicious node frequently. The nodes that receive DIS messages are forced to reset their trickle timers and flood the network with DIO messages.

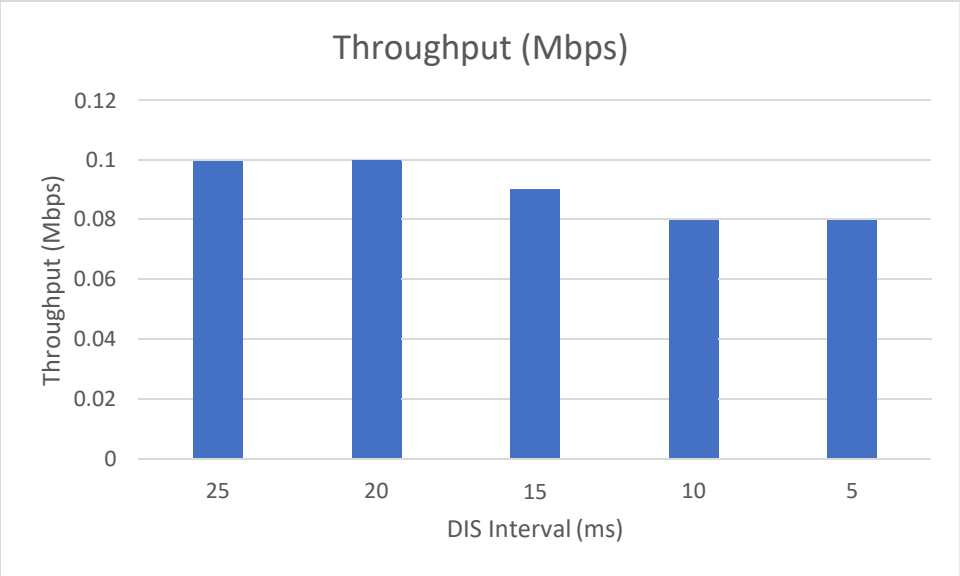
**Case 2: Application throughput Vs. DIS Interval Time**

We fix the application generation rate to 250 Kbps and vary the DIS interval to see the impact of DIS flooding on the network performance.

To change the DIS Interval parameter, go to Sensor Properties -> Network_Layer -> DIS_Interval -> 25ms.

| DIS Interval (ms) | Throughput (Mbps) |
|---|---|
| **25** | 0.10 |
| **20** | 0.10 |
| **15** | 0.09 |
| **10** | 0.08 |
| **5** | 0.08 |

This can be further understood with the help of following plots:




We can observe that the application throughput decreases as we decrease DIS Interval time. Upon decreasing the DIS interval, more DIS messages will be sent by the malicious nodes more frequently. Legitimate sensors spend more time in processing and responding to DIS messages than sending the data packets.

DIS flooding severely degrades the performance of Low Power and Lossy Networks (LLNs) because of the increase in control packet overhead.

**Appendix: NetSim source code modifications**

Set malicious node id. , in malicious.c file, within RPL project

```
/* User can set the MALICIOUS node ID*/
#include "main.h"
#include "RPL.h"
#include "RPL_enum.h"
#define MALICIOUS_NODE1 1
int fn_NetSim_RPL_MaliciousNode(NetSim_EVENTDETAILS*);
void rpl_drop_msg();
int fn_NetSim_RPL_MaliciousNode(NetSim_EVENTDETAILS* pstruEventDetails)
{
if (pstruEventDetails->nDeviceId == MALICIOUS_NODE1)
{ /*For multiple malicious nodes use if(pstruEventDetails->nDeviceId == MALICIOUS_NODE1 ||
pstruEventDetails->nDeviceId == MALICIOUS_NODE2)*/
return 1;
}
return 0;
}
```

**Changes to rpl_process_ctrl_msg(), in RPL_Message.c file, within RPL project**

```
void rpl_process_ctrl_msg()
{
switch (pstruEventDetails->pPacket->nControlDataType % 100)
{
case DODAG_Information_Object:
if (fn_NetSim_RPL_MaliciousNode(pstruEventDetails))
rpl_drop_msg();
else
rpl_process_dio_msg();
break;
case Destination_Advertisement_Object:
rpl_process_dao_msg();
break;
case DODAG_Information_Solicitation:
rpl_process_dis_msg();
break;
default:
fnNetSimError("Unknown rpl ctrl msg %d in %s",pstruEventDetails->pPacket->nControlDataType,
                                                                    __FUNCTION__);
break;
}  }
```