Secure AODV in MANET

Software: NetSim Standard v13.3, Microsoft Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Secure-AODV-v13.3/archive/refs/heads/main.zip

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

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

Introduction:

SAODV is an extension of the AODV routing protocol that can be used to protect the route discovery mechanism by providing security features like integrity and authentication. The reason only route discovery is secured by AODV is that data messages can be protected using a point-to-point security protocol like IPSec. SAODV uses a key management system, and each node maintains public keys, encryption keys, and decryption keys.

To implement SAODV, we have added **Secure AODV.c, RSA.c, and Malicious.c** files in the AODV project. RSA.c file is used to generate keys, encrypt, and decrypt the data. Users can implement their own encryption algorithms by changing the RSA.c file. malicious.c file is used to identify malicious nodes present in the network.

Example:

1. The **Secure_AODV_Workspace** 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 **Secure_AODV_Example** from the list of experiments.
2. After running the simulation, a **Secure_AODV.log** file gets created in the temp path inside the log folder.

Secure AODV implementation:

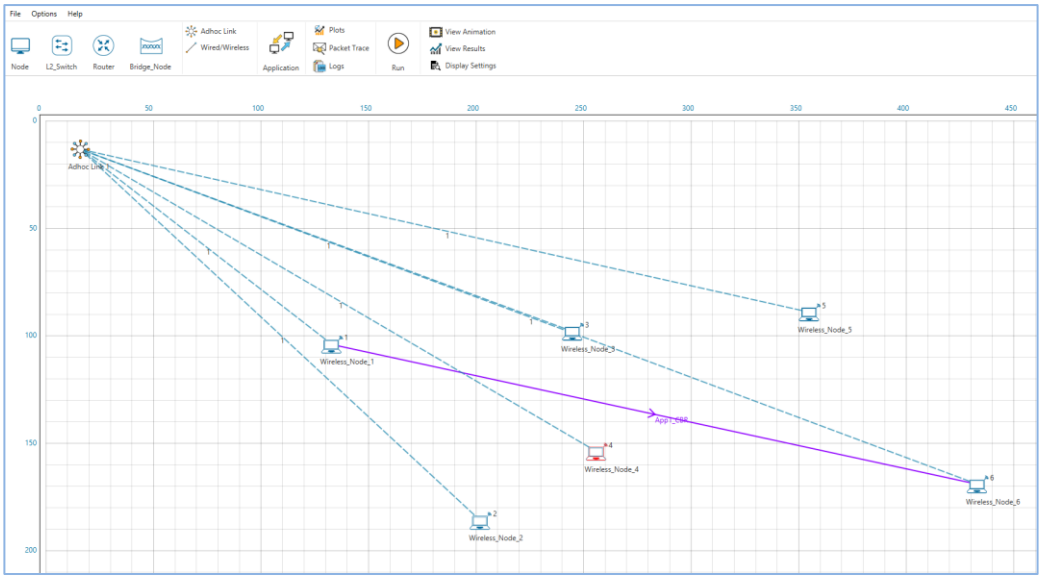

Figure 1: Network setup for Secure AODV

- Open the Source code in Visual Studio by going to Your work -> Source Code and Open code in the NetSim Home Screen window.
- Expand the AODV project.

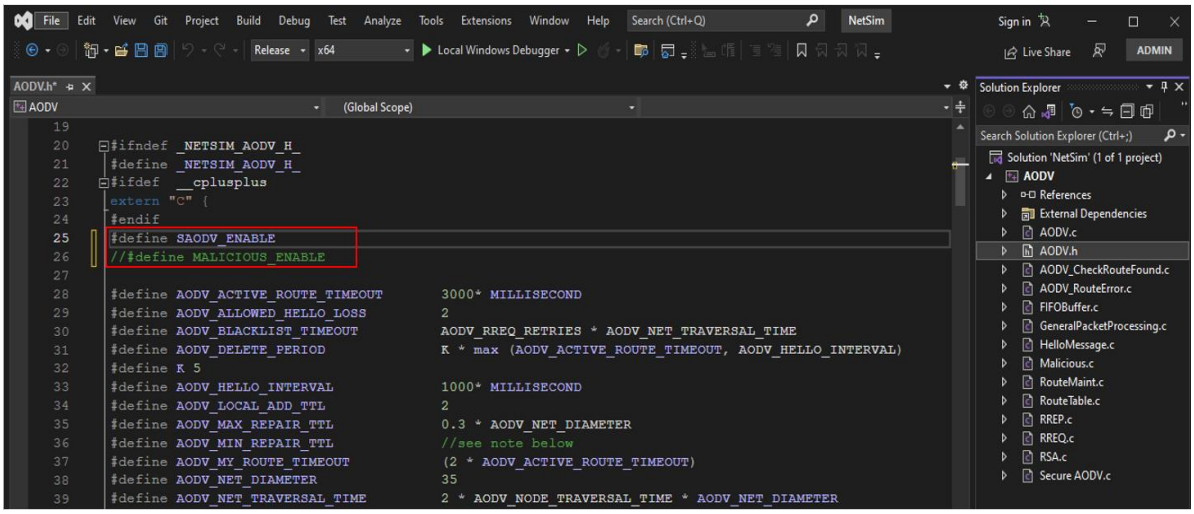

Figure 2: Screenshot Solution Explorer of AODV project

- Here users can enable Secure AODV (Open **AODV.h** file).

  Uncomment the line **#define SAODV_ENABLE**,

  Comment the line **//#define MALICIOUS_ENABLE** present in **AODV.h** file.

Rebuild the solution and run the simulation.

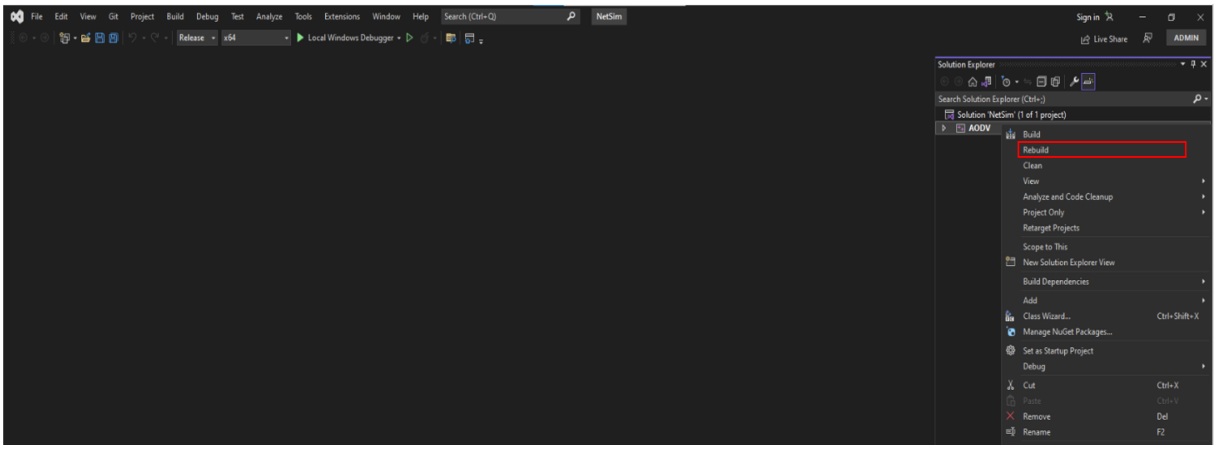

Figure 3: Screenshot of NetSim project Source Code in Visual Studio

A **Secure_AODV.c** file is added to the AODV project which contains the following important functions:

- **saodv_encrypt_packet();** //This function is used to encrypt the control packet data
- **saodv_decrypt_packet();** //This function is used to decrypt the control packet data
- **get_rrep_str_data();** //This function is used to get the route reply data from AODV_RREP control packet
- **get_rreq_str_data();** //This function is used to get the route request data from AODV_RREQ control packet
- **get_saodv_ctrl_packet_type();** //This function is used to change the control packet type from AODV (AODV_RREQ, AODV_RREP) to SAODV (SAODV_RREQ, SAODV_RREP)
- **get_saodv_ctrl_packet();** //This function is called whenever a new control packet is generated
- **get_aodv_ctrl_packet();** //This function is called while processing the control packets

Results and discussion:

After the simulation of the given Configuration file, open packet animation. In the packet animation, users can notice **SAODV_RREQ** and **SAODV_RREP** control packets.

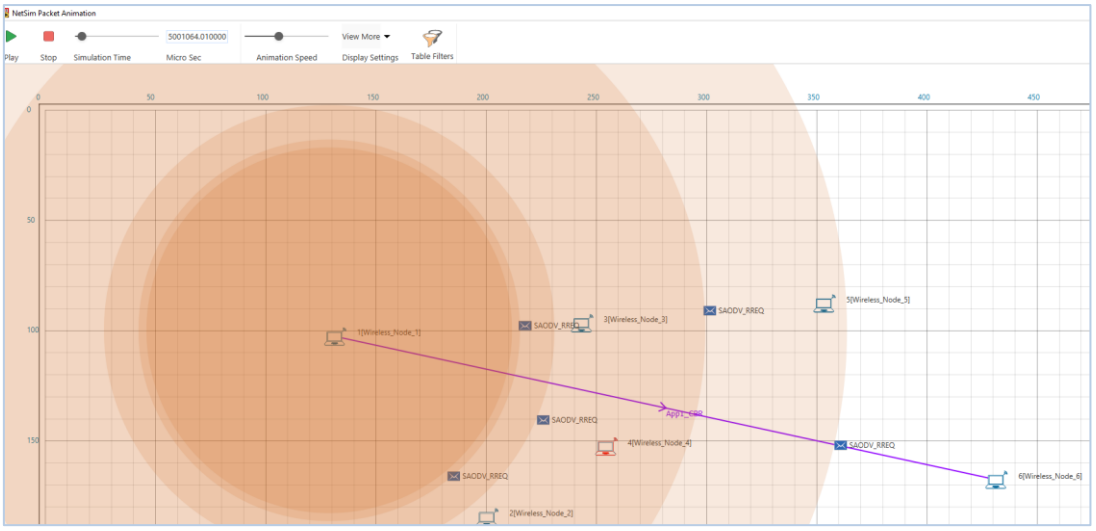

Figure 4: NetSim Animation Window

The SAODV codes also logs certain details in SAODV.log (i.e. present in the temp path inside the log folder).

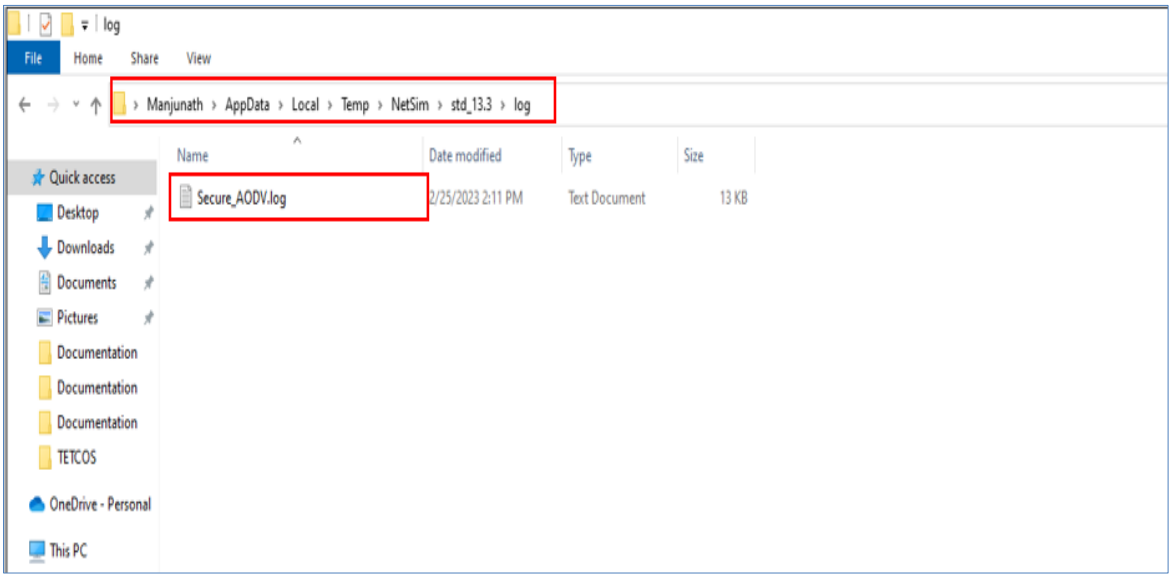

Figure 5: Secure_AODV log file path.

The format of the log file is such that each control packet is logged. The first line represents the packet type and the numbering used in a NetSim internal numbering system where **30701 is RREQ and 30702 is RREP**. The second line is the message which is encrypted. The third line contains the encrypted message after running the RSA encryption algorithm. The fourth line is after decryption and if everything is OK, the 2nd and 4th lines must match.

.....

Packet Type = 30701

Org Data = 1,0,1,11.1.1.6,0,11.1.1.1,1

Encrypted Data = *-Ÿ-*-¡*¡*¡-Ÿ-**¡*¡*¡****

Decrypted Data = 1,0,1,11.1.1.6,0,11.1.1.1,1

.....

Malicious node implementation:

Here users can enable code to malicious node problems. Enable #define MALICIOUS_ENABLE and comment //#define SAODV_ENABLE that are present inside AODV.h file and Rebuild the Solution.

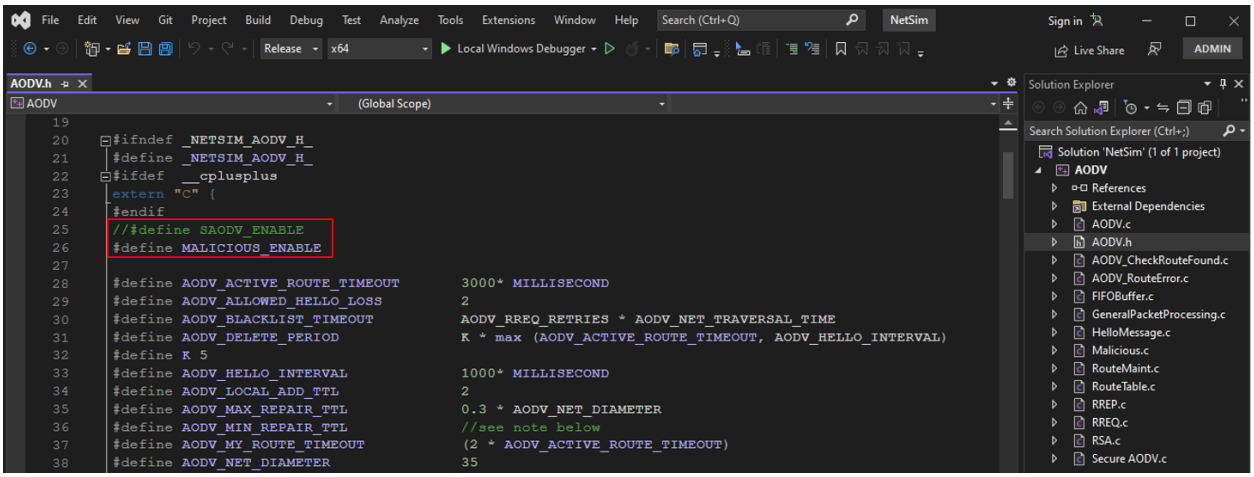

Figure 6: Commit and Uncommit for SAODV and Malicious code

A malicious node advertises wrong routing information to produce itself as a specific node and receives whole network traffic.

After receiving the whole network traffic, it can either modify the packet information or drop them to make the network complicated.

In packet animation, users can notice that malicious nodes will take all the packets and drops without forwarding them to the destination.

A file **malicious.c** is added to the AODV project which contains the following functions:

- **IsMaliciousNode();** //This function is used to identify whether a current device is malicious or not in-order to establish malicious behavior.
- **fn_NetSim_AODV_MaliciousRouteAddToTable();** //This function is used to add a fake route entry into the route table of the malicious device with its next hop as the destination.
- **fn_NetSim_AODV_MaliciousProcessSourceRouteOption();** //This function is used to drop the received packets if the device is malicious, instead of forwarding the packet to the next hop

Rebuild the solution and Run the simulation.

- Results and discussion:
- You can set any device as a malicious node, and you can have more than one malicious node in a scenario.
- Device IDs of malicious nodes can be set using the malicious_node [ ] array present in malicious.c file. Comment the line #define SAODV_ENABLE present in AODV.h file.
- Rebuild the solution and run the simulation.
- If we run the simulation without SAODV, we will get zero throughputs because the malicious node gets all the packets and drops without forwarding them to the destination. You can notice this in the NetSim packet animation.

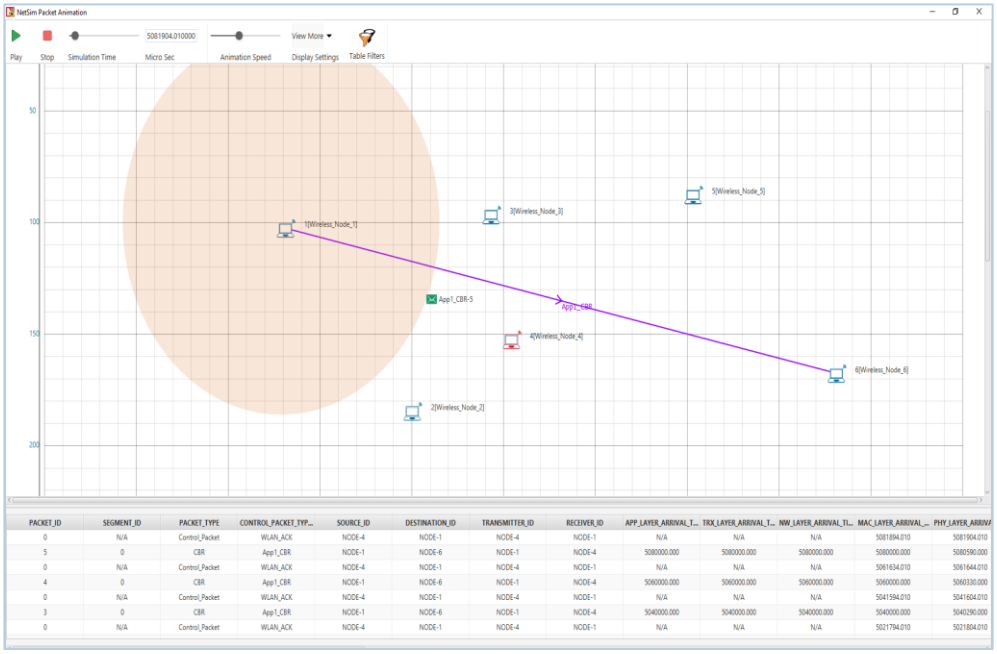

Figure 7: NetSim Animation Window

Both Secure AODV and Malicious node implementation:

Enable (Uncomment) the below mentioned lines of code present in AODV.h file.

#define SAODV_ENABLE

#define MALICIOUS_ENABLE

Rebuild the solution and run the simulation.

Results and discussion

Packets will be transmitted to the destination since SAODV helps in overcoming the Malicious Node problem. Route reply RREP from malicious node 4 will not be accepted by Node 1. It takes the Route reply from node 2 and forms the route. Users can observe the packet flow in the packet animation window.

The SAODV logs certain details in **Secure_AODV.log**.

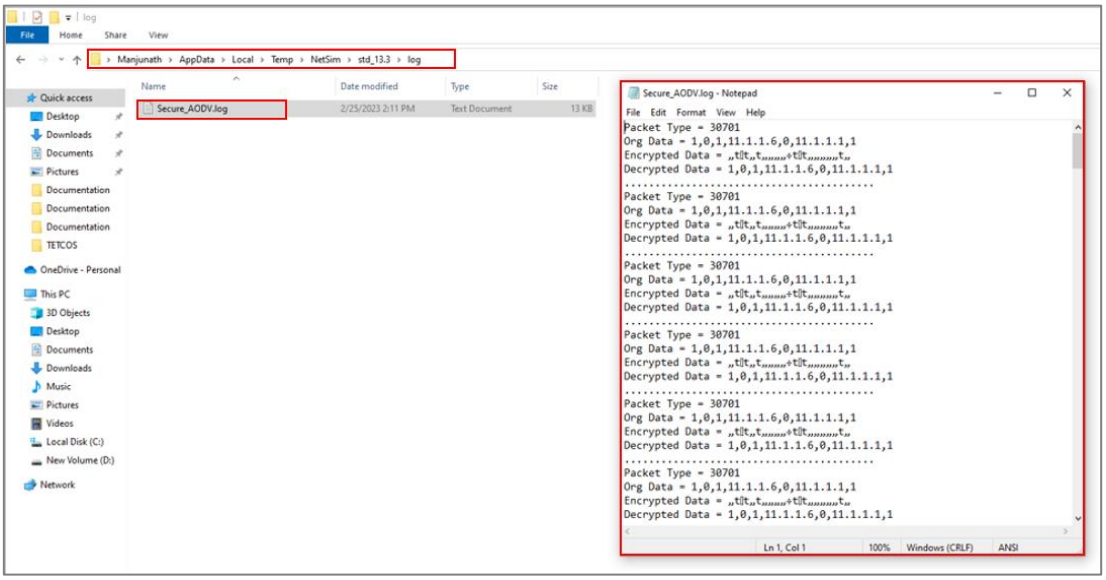

Figure 8: Secure AODV log file

The first line represents the packet type 30701 = RREQ. The second line is the message logged by SAODV when malicious node tries to decrypt the message.

.....

Packet Type = 30702

Encryption and decryption fail. This could be a malicious node.

.....

Packet Type = 30702

Encryption and decryption fail. This could be a malicious node.

.....

Appendix: NetSim source code modifications

We have added Secure_AODV.c, RSA.c and Malicious.c files, we have added the following macros code in AODV.h file within AODV project.

#define SAODV_ENABLE

#define MALICIOUS_ENABLE

Then we have added the following lines of code in enum_AODV_Ctrl_Packet in AODV.h file

```
//#ifdef SAODV_ENABLE
 SAODV_RREQ,
 SAODV_RREP,
 SAODV_RERR,
//#endif
```

We have added the following function prototypes in AODV.h file, within AODV project.

```
#ifdef SAODV_ENABLE
        void get_saodv_ctrl_packet(NetSim_PACKET* packet);
        void get_aodv_ctrl_packet(NetSim_PACKET* packet);
        void saodv_copy_packet(NetSim_PACKET* dest, NetSim_PACKET* src);
        void saodv_free_packet(NetSim_PACKET* packet);
        void remove_from_mapper(void* ptr, bool isfree);
#endif // SAODV_ENABLE
        bool IsMaliciousNode(NETSIM_ID devId);
```

We have added the following function prototypes in AODV.c file

```
int fn_NetSim_AODV_MaliciousRouteAddToTable(NetSim_EVENTDETAILS*);
int fn_NetSim_AODV_MaliciousProcessSourceRouteOption(NetSim_EVENTDETAILS* );
```

Changes to NETWORK_IN event in fn_NetSim_AODV_Run() function in AODV.c file, within AODV project

```
 #ifdef SAODV_ENABLE
        switch (pstruEventDetails->pPacket->nControlDataType)
        {
                case SAODV_RREQ:
```

```
		case SAODV_RREP:
		case SAODV_RERR:
		get_aodv_ctrl_packet(pstruEventDetails->pPacket);
		break;
	}
	if (pstruEventDetails->pPacket == NULL)
	{
		return -1; //Decryption fail.
	}
#endif // SAODV_ENABLE
```

We have added the following lines of code in AODVctrlPacket_RREQ and default cases in NETWORK_IN event to check the current node is malicious or not.

```
if (IsMaliciousNode(pstruEventDetails->nDeviceId))
	fn_NetSim_AODV_MaliciousRouteAddToTable(pstruEventDetails);
```

Changes code in fn_NetSim_AODV_CopyPacket () function, in AODV.c file, within AODV project

---

```
#ifdef SAODV_ENABLE
switch(srcPacket->nControlDataType)
{
case  SAODV_RERR:
case  SAODV_RREQ:
case  SAODV_RREP:
saodv_copy_packet(destPacket,srcPacket);
return 0;
break;
```

```
default:
#endif
        return  fn_NetSim_AODV_CopyPacket_F(destPacket,srcPacket);
#ifdef SAODV_ENABLE
        break;
        }
#endif
```

Changes code in int fn_NetSim_AODV_FreePacket () present in the AODV.c file, within AODV project

---

```
#ifdef SAODV_ENABLE
    switch (packet->nControlDataType)
    {
    case SAODV_RERR:
    case SAODV_RREQ:
    case SAODV_RREP:
    saodv_free_packet(packet);
    return 0;
    break;
    default:
        remove_from_mapper(packet->pstruNetworkData->Packet_RoutingProtocol, true);
        return 0;
        break;
    }
#endif // SAODV_ENABLE
```

Changes code in fn_NetSim_AODV_GenerateRREQ (), fn_NetSim_AODV_RetryRREQ () and fn_NetSim_AODV_ForwardRREQ () functions present in RREQ.c file, within AODV project

---

```
#ifdef SAODV_ENABLE
        get_saodv_ctrl_packet(packet);
#endif
```

Changes code in fn_NetSim_AODV_GenerateRREP(), fn_NetSim_AODV_ForwardRREP () and fn_NetSim_AODV_GenerateRREPByIntermediate () functions present in RREP.c file, within AODV project

---

```
#ifdef SAODV_ENABLE
        get_saodv_ctrl_packet(packet);
#endif
```