MATLAB INTERFACE FOR RPL DODAG VISUALIZATION

Software: NetSim Standard v14.0, Visual Studio 2022, MATLAB R2019 and higher

Project Download Link:
https://github.com/NetSim-TETCOS/RPL-DODAG-Visualization-v14.0/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:

RPL (Routing Protocol for Low-Power and Lossy Networks) is a routing protocol for wireless networks with low power consumption and generally susceptible to packet loss. It is a proactive protocol based on distance vectors and operates on IEEE 802.15.4, optimized for multi-hop and many-to-one communication, but also supports one-to-one messages.
This protocol is specified in RFC 6550 with special applications in RFCs 5867, 5826, 5673 and 5548. RPL can support a wide variety of link layers, including those with limitations, with potential losses or that are used in devices with limited resources. This protocol can quickly create network routes, share routing knowledge and adapt the topology in an efficient way.
RPL creates a topology similar to a tree (DAG or directed acyclic graph). Each node within the network has an assigned rank (Rank), which increases as the teams move away from the root node (DODAG). The nodes resend packets using the lowest range as the route selection criteria.
The DODAG formed as part of RPL protocol in NetSim can be visualized by interfacing NetSim with MATLAB.

DODAG Visualization in NetSim with MATLAB Interfacing:

DODAG Visualization is implemented in NetSim by Interfacing with MATLAB for the purpose of Visualising the Directed Acyclic Graph(DAG).

The codes required for the DODAG Visualization done in MATLAB are written to a PlotDAG.m file and this file is available in the MATLAB folder under bin_x64 of **RPL_DODAG_Visualization in_IoT_Workspace**.

Steps to simulate:

1. Add the following MATLAB install directory path in the Environment PATH variable
   <MATLAB_INSTALL_DIRECTORY>\bin\win64
   For eg: C:\Program Files\MATLAB\R2020b\bin\win64

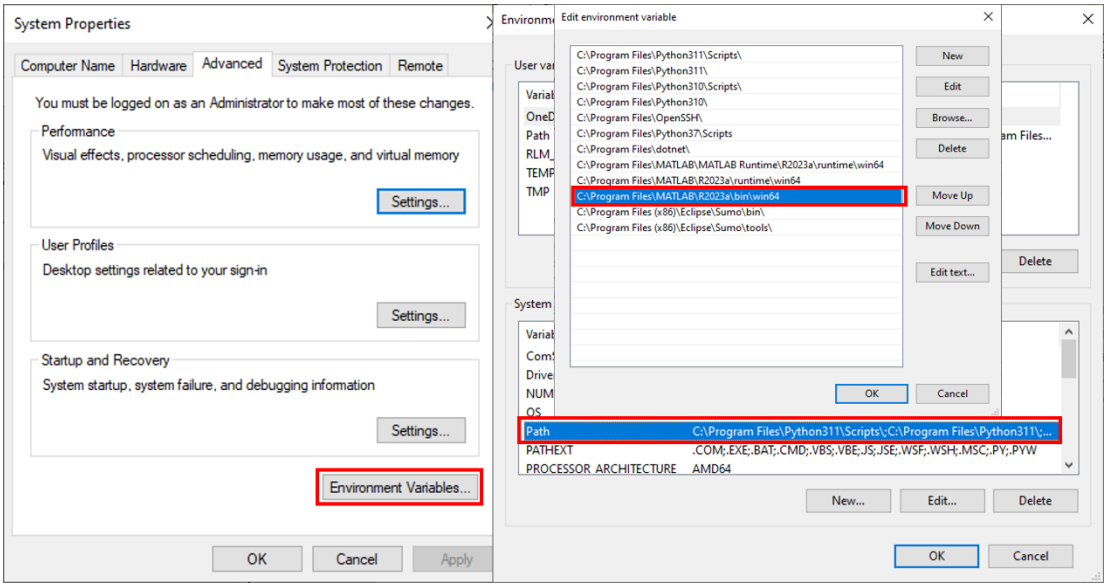

Figure 1: Environment variable PATH

Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

2. Open Command prompt as admin and execute the command “matlab -regserver”. This will register MATLAB as a COM automation server and is required for NetSim to start MATLAB automation server during runtime.

Example:

- The **RPL_DODAG_Visualization_in_IoT_Workspace** comes with a sample network configuration that are already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **RPL_DODAG_Visualization_in_IoT_Example** from the list of experiments.

- The saved network scenario consists of
  a. 5 Wireless Sensor Nodes
  b. 1 LOWPAN Gateway
  c. 1 Router
  d. 1 wired node

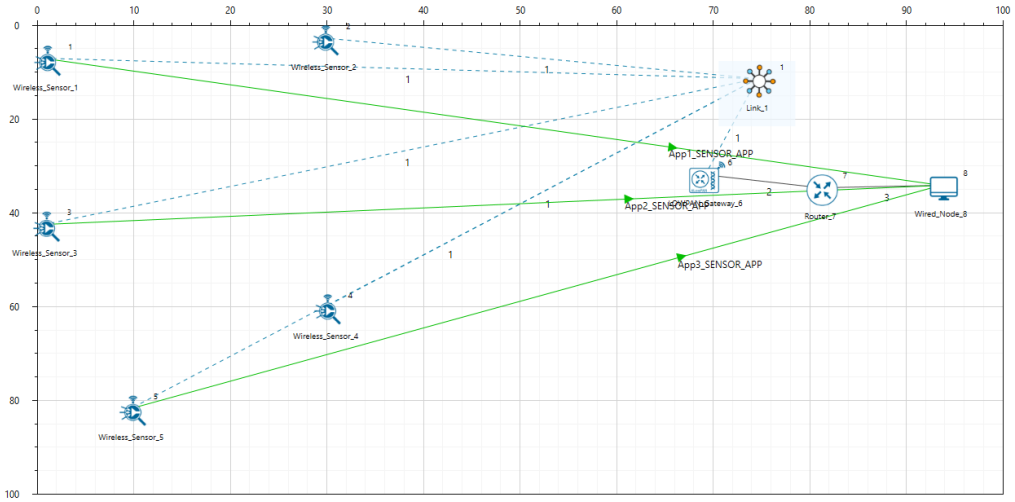

Figure 2: Network Scenario

- Run Simulation.

Results and discussion

A plot will open, showing the DODAG when the simulation is started, and the first route is formed between LOWPAN Gateway and the sensors. And the DODAG will be dynamically updated. The DoDAG gets plotted in MATLAB as shown below:

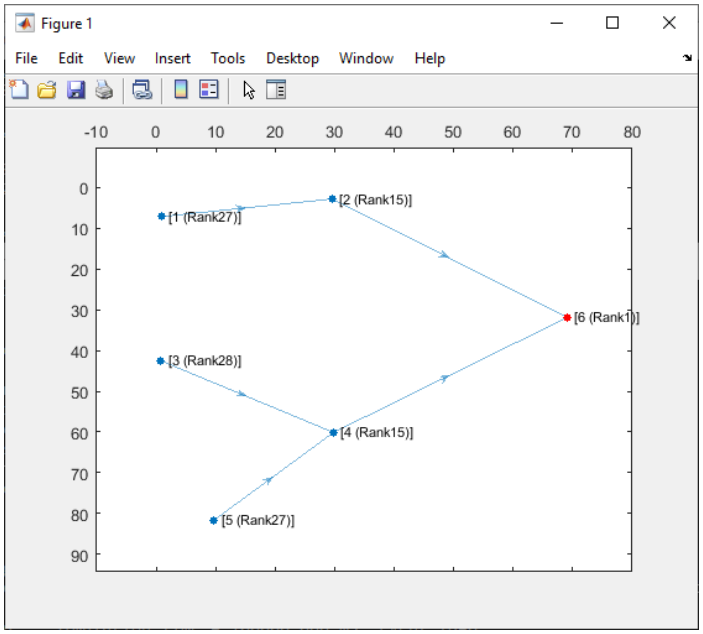

Figure 3: Plot of DODAG Formation in MATLAB

Appendix: NetSim source code modifications

Added MATLAB Functions (), in DoDAG_Plot.c file, within RPL project

/* Which takes care of interactions with MATLAB* /

```
double fn_netsim_matlab_DODAG_run()
{
char theArray[BUFSIZ * 2], Xc[BUFSIZ * 2], Yc[BUFSIZ * 2];
char H[BUFSIZ * 2], nID[BUFSIZ * 2], nRank[BUFSIZ * 2];
char command[BUFSIZ], output[BUFSIZ];
unsigned int i, j = 0;
i = 0;
dcounti = 0;
sprintf(Xc, "Xc=[");
sprintf(Yc, "Yc=[");

sprintf(theArray, "theArray=[");
sprintf(nID, "nID=[");
sprintf(nRank, "nRank=[");
while (dcounti < NETWORK->nDeviceCount)
{
if (!(isSENSOR(dcounti) || isSINKNODE(dcounti)))
{
dcounti++;
continue;
}
j = 0;
dcountj = 0;
while (dcountj < NETWORK->nDeviceCount)
{
if (!(isSENSOR(dcountj) || isSINKNODE(dcountj)))
{
dcountj++;
continue;
}

if  (is_prefparent(NETWORK->ppstruDeviceList[dcounti]->nDeviceId,  NETWORK->ppstruDeviceList[dcountj]-
>nDeviceId))
{
sprintf(theArray, "%s%d,", theArray, 1);
}
else
{
sprintf(theArray, "%s%d,", theArray, 0);
}

j++;
dcountj++;
}
PRPL_NODE rpl = GET_RPL_NODE(NETWORK->ppstruDeviceList[dcounti]->nDeviceId);
if (isSINKNODE(dcounti))
{
sprintf(nRank, "%s%d,", nRank, 1);
}
else if (rpl->joined_dodag != NULL)
{
sprintf(nRank, "%s%d,", nRank, rpl->joined_dodag->rank);
```

```
}
else
{
sprintf(nRank, "%s%d,", nRank, 0);
}

sprintf(Xc, "%s%lf,", Xc, DEVICE_POSITION(NETWORK->ppstruDeviceList[dcounti]->nDeviceId)->X);
sprintf(Yc, "%s%lf,", Yc, DEVICE_POSITION(NETWORK->ppstruDeviceList[dcounti]>nDeviceId)->Y);
sprintf(nID, "%s%d,", nID, NETWORK->ppstruDeviceList[dcounti]->nConfigDeviceId);


if (isSINKNODE(dcounti))
{
sprintf(H, "H=%d;", i + 1);
}
sprintf(theArray, "%s;", theArray);
i++;
dcounti++;

}
sprintf(Xc, "%s];", Xc);
sprintf(Yc, "%s];", Yc);
sprintf(nID, "%s];", nID);
sprintf(nRank, "%s];", nRank);
sprintf(theArray, "%s];", theArray);

netsim_matlab_send_ascii_command(Xc);
netsim_matlab_send_ascii_command(Yc);
netsim_matlab_send_ascii_command(nID);
netsim_matlab_send_ascii_command(nRank);
netsim_matlab_send_ascii_command(theArray);
netsim_matlab_send_ascii_command(H);

sprintf(command, "output=PlotDAG(theArray,Xc,Yc,H,nID,nRank)");
netsim_matlab_send_ascii_command(command);

netsim_matlab_get_value(output, BUFSIZ, "output", "double");

return 0;
}
double fn_netsim_matlab_finish()
{
        fprintf(stderr, "\nPress any key to terminate MATLAB process...\n");
        _getch();
        netsim_matlab_interface_close();
        return 0;
}
```

Changes code in fn_NetSim_RPL_init() and fn_NetSim_RPL_Finish (), in RPL.c, within RPL project

---

```
void netsim_matlab_interface_start()
{
        char exeMATLAB[BUFSIZ];
        sprintf(exeMATLAB, "start %s%s%s", pszAppPath, pathSeperator, "MatlabInterface.exe");
        (void)system(exeMATLAB); // starts MATLAB socket interfacing engine
}
/**
RPL Init function initializes the RPL parameters.
*/
```

```
_declspec (dllexport) int fn_NetSim_RPL_Init(struct stru_NetSim_Network *NETWORK_Formal,

NetSim_EVENTDETAILS *pstruEventDetails_Formal, char *pszAppPath_Formal,
 char *pszWritePath_Formal, int nVersion_Type, void **fnPointer)
{
        netsim_matlab_interface_start();
        netsim_matlab_interface_configure(pszAppPath);
        fn_netsim_matlab_DODAG_run();
        return fn_NetSim_RPL_Init_F();
}


/*This function is called by NetworkStack.dll, once simulation end to free the allocated memory for the network*/

_declspec(dllexport) int fn_NetSim_RPL_Finish()
{
        fn_netsim_matlab_finish();
        return fn_NetSim_RPL_Finish_F();
}
```

Changes code to rpl_add_route_to_parent(), in Neighbor.c file. within RPL project

```
rpl_add_route_to_parent(d, dodag->pref_parent->nodeId);
fn_netsim_matlab_DODAG_run();
for (i = 0; i < rpl->neighbor_count; i++)
{
        PRPL_NEIGHBOR neighbor = rpl->neighbor_list[i];

        if (matching_ranks[i] >= INFINITE_RANK)
        {
        if (neighbor->lastDIOMSG != NULL)
        { /* forget messages from other DODAG iterations */
        rpl_dio_pdu_free(neighbor->lastDIOMSG);
        neighbor->lastDIOMSG = NULL;
        }
        continue;
}
```