**Problem2**

For the part 1 which involves Wi-Fi Network, we need to establish wireless local area network with Ad-hoc mode. Nodes move by following a 2D random walk in a rectangular area defined by the low-left corner (x = -90, y = -90) and the upper-right corner (x = 90, y = 90).

First task requires us to establish the Wireless Local Area Network in Ad-hoc mode. The first thing we need to task consideration is the header files. Clearly, there is no need to build the CSMA network and point-to-point network which means these network’s head files are not needed in the code. Owing to the fact, the head files we used in the part is just shown as the following:

屏幕上有字

描述已自动生成

For this part, we need to make specific illustration to the head file of ssid. Actually, its full name is “Service Set Identifier”. Such a technology can divide one wireless local area network into serval subnets which requires different identification. That is to say, only the people who pass the specific identification can get access to the corresponding subnet.

Next step we just gift the namespace with the name we wanted. The code is just shown as the following:

文本

描述已自动生成

In the next part, we are going to claim the log component whose name is “ThirdScriptExample”, we can turn on or turn off the output of the log component with the method of quoting the name, the code is just shown as the following:

手机屏幕的截图

描述已自动生成

After the above preparatory work done, we will begin to deal with the main function.

Just like previous files, we need to add some parameters of command lines to turn on or turn off some log components or just change the numbers of device we need to use.

文本

描述已自动生成

Just like the code shown above, you can see one of the variables is called “useRts”. That means we want to use RTS/CTS method to deal with the issue of hidden stations. While it is true, then the real exchange of data only occurs after the signal of RTS/CTS protocol exchanged successfully. Then for the rest two variables, the variable whose name is verbose is just used to determine whether to open the two UdpApplications’ logging components, if it is true, then the components are opened, while for the other variable nWifi, we just use it to define how many station nodes we need in the wifi network. Since it is 5, then there are 5 station nodes in the wifi network we built.

For the next part, what we are going to do is just to create the object of CommandLine, we just name it as cmd. Then we will add corresponding parameters to it.

文本

描述已自动生成

And for the next part, we are going to make some restriction to wifi network’s parameters. In the following code, you can clearly see that we set the parameter as 18. That’ s because of the configuration of the grid position allocator. If it is larger than 18, then the grid will exceed boundary.

Afterwards, what we are going to do is just set the verbose parameter and useRts as True, so that we can track the file and work out the hidden station issue. The corresponding code is shown as the following:

文本

描述已自动生成

This part we can get involved the work of network topology construction. First of all, we can construct the wifi network. What we need to do at the first step is just that we can create the wifi network nodes. We need to create nodes of two different types. First type of node is STA nodes while the other is Ap node. The code used for creating these nodes are just shown as the following:

文本

描述已自动生成

After nodes established, we can get down to channel setting as well as Wifiphy setting. Here we will use two helper classes: YansWifiChannelHelper and YansWifiPhyHelper. We need to make extra instruction to the definition of “phy”. Actually, that means Port Physical Layer, and phy is just the simplified name like OSI.

文本

描述已自动生成

With these codes above, we can set three models, fist code can make us establish two models, they are constant speed propagation delay model and logdistance propagation loss model. With the code of second line, we can set NistErrorRate model. Finally, we just use the default configuration for PHY as well as default channel model.

After that, we can set WifiMAc and install Netdevice in the nodes. There are two helper classes we will use here, they are WifiHelper and WifiMacHelper. Here is the code we used for the part:

文本

描述已自动生成

In this part, the method of SetRemoteStationManager can tell the helper to use which kind of algorithm. In the part1, we will use AARF algorithm. Then for the WifiMacHelper part, we are going to create the object of the class Ssid so that we can use it to set the value of SSID in the MAC layer. Finally, we will install the ad-hoc nodes in the Wifi network.

And this part we are going to build mobile model. The program will use Cartesian coordinate system to label the location of each node. The helper class we will use for here is MobilityHelper. We will make illustration to the following code:

文本

描述已自动生成

While it comes to the model setting for moving nodes, we can divide the model into two parts. The first part is the original distribution of nodes while the second one is the moving trace model for the nodes. Clearly in the previous parts we have already determined to use gird pattern. That is the reason why we set the GridPositionAllocator as the allocator we used. After that, we can put our nodes into the 2-dimensional Cartesian coordinate system. DeltaX and DeltaY is the distance between each node in x-axis and y-axis. Then the “GridWidth” means the maximum number of nodes in each row.

After the grid position allocator set, we can construct movement trajectory model in the part. As you see, the model we use is “RandomWalk2dMobilityModel”. The default range of the speed for the nodes is between 2 m/s and 4m/s. And you can see that the parameter set in the instruction of SetMobilityModel is just to set the boundary for the region of nodes moving. And according to the issue, that should be -90, 90, -90 and 90. For the last step, we are going to install nodes in the model with the instruction “mobility.Install(wifiAdhocNodes)”. Just like shown above.

文本

描述已自动生成

Here we will install the TCP/IP protocol stack, the helper class we will use here is InternetStackHelper. First of all, we will create one object for the class. Then we will use the ‘Install’ instruction to install the nodes we created into the stack. Then we create one object for the class Ipv4AddressHelper with the name ipv4. What we will do here is to use the object to allocate IP address for the network devices. Since there are two types of nodes, we need to repeat the step of address assigning twice.

图片包含 文本, 信件

描述已自动生成

And according to the requirement of the issue, the address we will allocate to the wifi network is 192.168.1.0 and 255.255.255.0. And we will complete the part with the instruction “Assign”.

文本, 信件

描述已自动生成

After that we will build the application layer to meet the requirement of the issue. And we know there are two clients and one server in the whole network. The code we used to deal with this part is shown as following:

文本

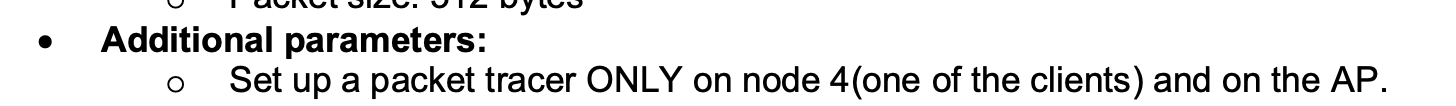
描述已自动生成

First of all, we create the echoServer who is the object of UdpEchoServer. The parameter is the index of port where it listens to the response. According to the issue, it is set as 21.

After the server program set, we need to install it into one specific node. Since the issue asks us to install it at Node 0. We have already known the total number of nodes nWifi is 5. Then we can write the index as 0 or nWifi-5 just like it shown above.

The part we need to be cautious is that they have two clients, so we need to use UdpEchoClientHelper for twice. According to the issue, the clients are at node 4 and node 3. To make the input more convenient, we just set index of 4 and 3 as nWifi - 1 and nWifi - 2.

Furthermore, you can see that two clients will both send two packages. Client 1 will send its package at 1s and 2s, while the client 2 will send its packages at 2s and 4s. So our server will begin to listen at 1s to avoid package loss.



文本

描述已自动生成

First of all, we can see the class we used for the first line is Ipv4GlobalRoutingHelper. We create one object of that class to help us to create routing table for each node based on the link-state advertisement of each node.

One thing that we need to notice is that the simulation we created will never naturally stop. That is because we asked the wireless access points to generate beacons. It will generate beacons forever and this will result in simulator events being scheduled into the future indefinitely. What does that mean? We have to tell the simulator to stop even though it might have beacon generation events scheduled. That second line of code shown above will help us to tell the simulator to stop so that we don’t simulate beacons forever and enter what is essentially an endless loop.

Then as you can see above, we need to only trace the package on node 4. And the index of node begins with 0. That means the parameter for EnablePcap is just 1 like we did above.

Finally, the last three lines of code can make us run the simulation, clean up and then exit the program.