Problem1:

The requirement of the issue is shown as the following:

文本, 信件, 电子邮件

描述已自动生成

It is obvious that we are going to establish the point-to-point network in the issue. First of all, we will construct the network topology we are going to achieve, which is shown as the following:

192.168.2.0/24

N0 ------------------------- N1

This part, we are going to modify and explain the code:

First part, we will take various of head files, by the way, we will make the claim to the name space:

文本

描述已自动生成

In the next part, we are going to claim the log component whose name is FirstScriptExample, 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

The major part of the construction of the network is the main function, we will make the illustration in sequence.

There are two lines of the code to make the two log components valid, these two components are established within the application of Echo Client and Echo Server. Furthermore, we will set the log component with the level of LOG\_LEVEL\_INFO. This part is shown as the following:

文本

描述已自动生成

After that, we will focus ourselves on the network topology implement. First of all, we will create the object of ns3 node, they are on behalf of the computer in the simulating process. We establish the Nodecontainer with the name of nodes. As the image of topology shown before, there are two nodes in the whole network, that means the value of the parameter of the Create method is 2. The code is shown as the following:

图形用户界面, 文本

描述已自动生成

After the nodes are established, we are going to create the connection between the points, we will use PointToPointHelper to help us to deploy and connect the PointToPointNetDevice and the object of PointToPointChannel. According to the issue, the transmission rate is 10 Mbps while the delay is 2ms. The code is just shown as the following:

文本

描述已自动生成

After the nodes are connected, we will install the protocol stack for the PointToPointHelper object and the point-to-point net device. Finally, we will arrange the IP address for the device of the node. The last part we will use the Ipv4AddressHelper to combine the IP addresses with the corresponding device. The code of the part is shown as the following:

文本

描述已自动生成

The next part we are going to work out is the application program. We will set the application of UdpEchoServer on our nodes, the parameter’s value is the port number. Then we use the method of Start and Stop to set the time interval. According to the issue, the port number is 63, so the code we modified is shown as the following:

图形用户界面, 文本

描述已自动生成

After the set of the UdpEchoServer, we begin to deal with the UdpEchoClient. We will use UdpEchoClientHelper to help us to set the application of client.

We firstly create the object of client, telling it that set its remote address as the server node’s IP address. Meanwhile we make it send the package to the port 9. For the part of attribute setting, three attributes we will set, they are the maximized number of packages we are going to deliver, the time interval between two packages and the size of data contained by each package. We also use the Start and Stop instruction to tell the client when to begin and when to stop. The code we modified is shown as the following:

文本

描述已自动生成

Finally, we run the simulator and destroy all the objects created to free the space. To enable to trace the package, we will use the promiscuous mode to trace the data. The code is shown as the following:

图片包含 文本

描述已自动生成

Result:

文本

描述已自动生成

There are two methods of tracing the message, first method is that we set the trace file for each device and just store the package delivered or received by the device, the second method is that we will choose one of the network devices, setting it as the promiscuous mode, so the single device can get all the data packages, storing them into a single pcap file. The method we take is the second one.

Then we will use wireshark to trace the package and make the analysis.

First-0-0-pcap

图片包含 图形用户界面

描述已自动生成

First-1-0-pcap

图形用户界面, 文本, 应用程序, 电子邮件

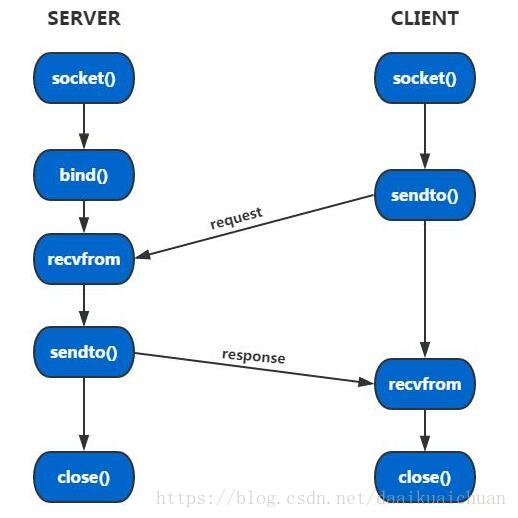
描述已自动生成

First of all, we need to make the illustration to the name of the pcap file. The name can be understood from the following formula:

Example-<Node Index>-<Net Device Index>.pcap. From the two files above, we can clearly see that the tracing work for two nodes are executed by the same net device. Then, if we look at the communication between the two nodes, we can find several interesting things. First, we can see the communicating process， first of all, the size of package delivered is not equal to the size of the data, since Udp transmission needs to capsulate the data. First reason is that it will add the IP header which is 22 bytes. Second reason is it will add a Udp header whose size is 8 bytes. Furthermore, since the point-to-point protocol also requires one header, that makes the package delivered has a bigger size.

Furthermore, we can see the mode of the UDP transmission, let us look at the image shown above.

It is obvious that we set the node 0 as the client and the node 1 as the server, and the time interval between two messages node 0 received is larger than the time interval node 1 has. Since the server requires some time to deal with the requirement, and the frame of the Udp transmission is shown as the following:



Let us make the deeper analysis to the file:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 电子邮件

描述已自动生成

We can find that checksum is validation disabled, the reason is that the receiver needs to re-compute the checksum and make the comparation to see if they are equal, so here the checksum should be ignored.

Problem2:

The real image of the network we are going to achieve is shown as the following:

图示, 日程表

描述已自动生成

The network contains:

1. 3 nodes in the first shared but operating under CSMA
2. 3 nodes in the second shared but operating under CSMA
3. 2 nodes in the point to point link, and furthermore, node2 and node3 have two network interfaces, one for each link they are connected.

The application running in the network are:

1. UDP Echo Server at Node 1: Listening on port 21.
2. UDP Echo Client at Node 5: Sends 2 UDP Echo packets to the server at times 4s and 7s.

We are required to use wireshark to capture and analyse the packet traces.

To implement the goal of the laboratory assignment, we will first design the network is shown as the following:

192.168.3.0

N0 N1 N2 ---------------N3 N4 N5

| | | | | |

= = = = = = = = = = = = = =

LAN 192.168.1.0. LAN 192.168.2.0

As the topology image shown above, what we need to do at the issue is build the CSMA network as well as point-to-point network, we will use second.cc file as the basement and just make some adjustment to make it adopted to the issue, next part we will explain the code in sequence just in order to make it clear. Finally, we will use show the packet trace and make the analysis.

Code Explanation:

As usual, we will firstly begin the code with various head files as well as defining the log component, just shown as the following:

文本

描述已自动生成

After that, in the main function, we will use verbose flag to judge whether the log components of UdpEchoClientApplication and UdpEchoSeeverApplication are enabled to use, if it is true, then both of them can be used, so we will set the verbose flag as True. Afterwards, since the property of CSMA network is that there must be two more nodes in each cluster, the value of nCsma should not less than 1. In the issue, we can clearly see that there are 3 nodes in each CSMA cluster, and that means the value of nCsma is 3 minus 1 which equals to 2. Then we can create the object of the class CommandLine, then add the attribute of nCsma and verbose to its parameter system which is shown as the following:

文本

描述已自动生成

To ensure that the two values of nCsma we set is bigger than zero, we use the two line of the code just in case.

图形用户界面

描述已自动生成

The next part we are about to create the object of NodeContainer p2pNodes and csmaNodes. As we have already known, point-to-point network in the issue contains the node 2 and node 3, that means, the parameter in the instruction of p2pNode.Create() is 2. Then we create the CSMA network clusters. Since there are two clusters according to the issue, and the index of client and server in each cluster is 1 and 2, the code we corrected is shown as the following:

文本

描述已自动生成

Afterwards, we establish the channel for each network and add the corresponding attribute of data transmission rate and time delay towards the nodes just shown as the following:

文本

描述已自动生成

Next part we are going to deal with the protocol stack, we will combine the network of point-to-point with the network of CSMA, then we will arrange the IP address for the networks just like the issue’s requirement.

The code for the part is just shown as the following:

文本

描述已自动生成

Finally, we begin to establish the transmission layer, this problem asks us to use the protocol of UDP, according to the issue’s requirement, the UDPEchoServer do the listening work on port 21, so we will arrange the port 21 to the server, and we will also add the corresponding attribute to the client and server:

文本

描述已自动生成

Finally, we will create the routing table for each node of the network, we will create pcap tracing file to trace point-to-point network, by the way, we will trace CSMA network in promiscuous mode. We will run the simulator for the last step.

文本

描述已自动生成

Result:

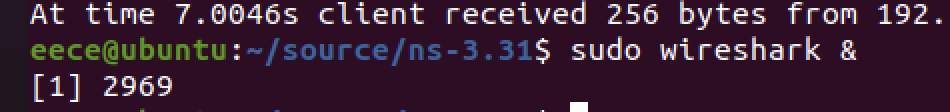
We run the program on Ubuntu system this part, and we finally get the following results:

文本

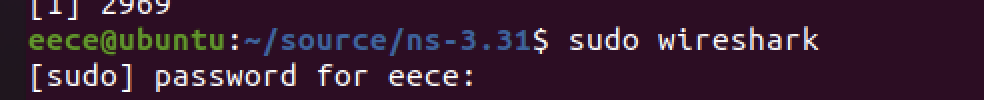
描述已自动生成

We can clearly see that the server we set on 192.168.1.2 and the client we set on192.168.2.3 have the interaction in the whole process just like the image shown above.

To make the analysis, we will use wireshark to trace and capture the package we need to analysis. Since there is problem when we use the following instruction:



We will use the following instruction to operate the wireshark:



CSMA means that carrier sense multiple access, that is to say, there are multiple terminals on one shared medium, and each terminal has one network device connected. There are two methods of tracing the message, first method is that we set the trace file for each device and just store the package delivered or received by the device, the second method is that we will choose one of the network devices, setting it as the promiscuous mode, so the single device can get all the data packages, storing them into a single pcap file. The method we take is the second one.

Then we get the following results:

Second-0-0.pcap

表格

描述已自动生成

As we can see above, the file above is the point-to-point network establishment, the source address is the 192.168.2.3 while the destination addressed the 192.168.1.2 whose port number is 21. That means, the file tracing the UDP Echo packages delivery.

Second-2-0-pcap

表格

描述已自动生成

This part we can see there is the new protocaol, ARP(Address Resolution Protocol). It is a TCP/IP protocol used for getting the physical address. The host broadcast the ARP request containing IP address of the destination to all the devices of the network and receive the returned message as well just so that it can enusre the goal address. It will also store the IP address and the physical address to the ARP buffer for a while just in order to save the time and resource.

Second-5-0-pcap

表格

描述已自动生成