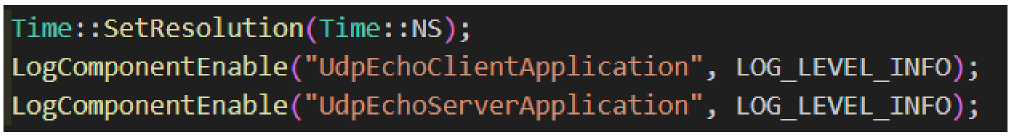
Computer networks
Lab2: Network Simulator with Ns-3

- Program:
  - Time Resolution and Logging:

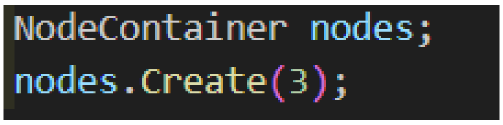
    Sets the time resolution to nanoseconds for the simulation and enables informational logging for the UDP echo applications.



```
Time::SetResolution(Time::NS);
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

  - Node Creation:

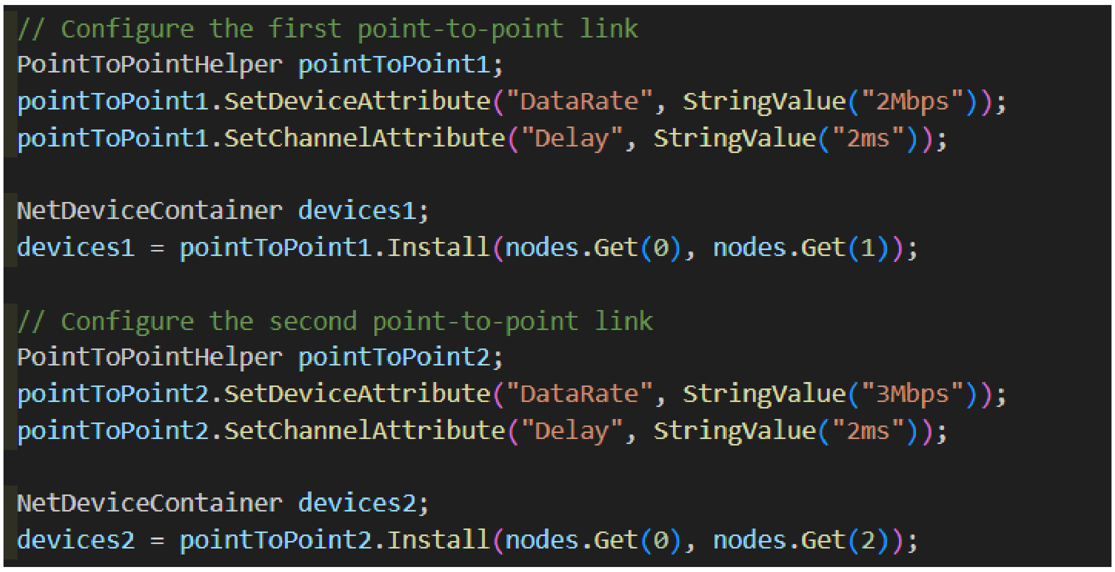
    Creates a container for nodes and instantiates three nodes within it.



```
NodeContainer nodes;
nodes.Create(3);
```

- Point-to-Point Helpers and Device Containers:

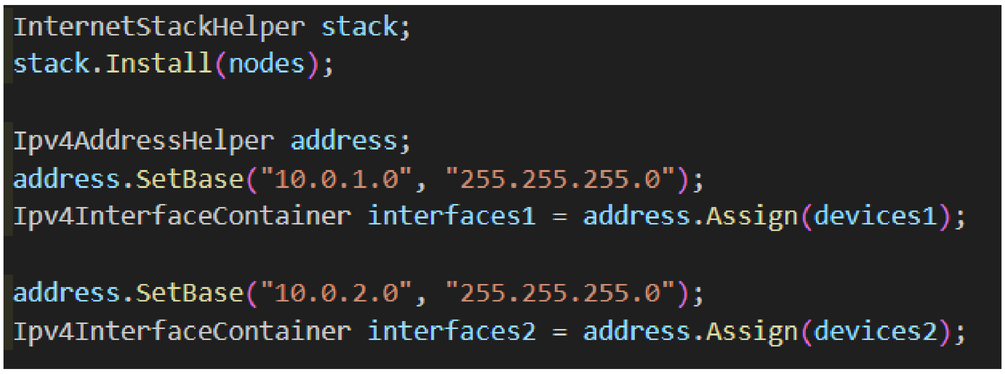
  Sets up two point-to-point links with specified data rates and delays. Each link connects Node 0 with one of the server nodes.



```
// Configure the first point-to-point link
PointToPointHelper pointToPoint1;
pointToPoint1.SetDeviceAttribute("DataRate", StringValue("2Mbps"));
pointToPoint1.SetChannelAttribute("Delay", StringValue("2ms"));

NetDeviceContainer devices1;
devices1 = pointToPoint1.Install(nodes.Get(0), nodes.Get(1));

// Configure the second point-to-point link
PointToPointHelper pointToPoint2;
pointToPoint2.SetDeviceAttribute("DataRate", StringValue("3Mbps"));
pointToPoint2.SetChannelAttribute("Delay", StringValue("2ms"));

NetDeviceContainer devices2;
devices2 = pointToPoint2.Install(nodes.Get(0), nodes.Get(2));
```

- IP Addressing:

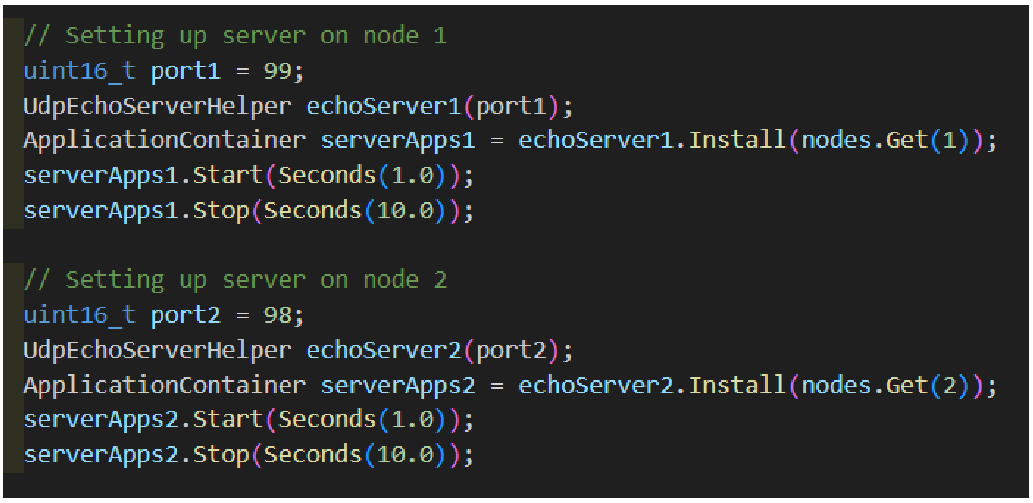
  Installs the internet stack on all nodes and assigns IP addresses to the devices on the point-to-point links.



```
InternetStackHelper stack;
stack.Install(nodes);

Ipv4AddressHelper address;
address.SetBase("10.0.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces1 = address.Assign(devices1);

address.SetBase("10.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces2 = address.Assign(devices2);
```

- Server Setup:

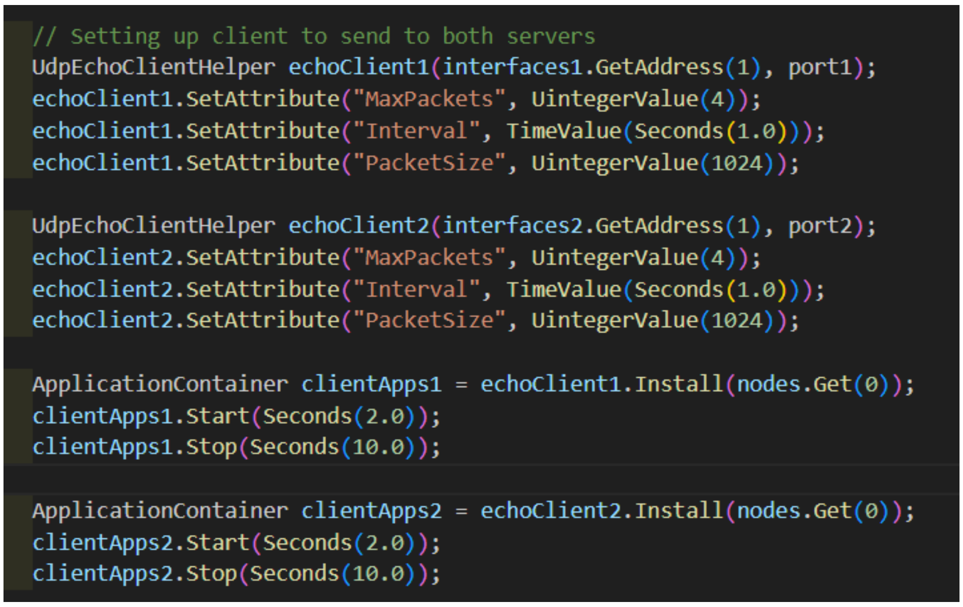
  Defines and starts UDP echo servers on Nodes 1 and 2, listening on ports 99 and 98, respectively.



```
// Setting up server on node 1
uint16_t port1 = 99;
UdpEchoServerHelper echoServer1(port1);
ApplicationContainer serverApps1 = echoServer1.Install(nodes.Get(1));
serverApps1.Start(Seconds(1.0));
serverApps1.Stop(Seconds(10.0));

// Setting up server on node 2
uint16_t port2 = 98;
UdpEchoServerHelper echoServer2(port2);
ApplicationContainer serverApps2 = echoServer2.Install(nodes.Get(2));
serverApps2.Start(Seconds(1.0));
serverApps2.Stop(Seconds(10.0));
```

- Client Setup:

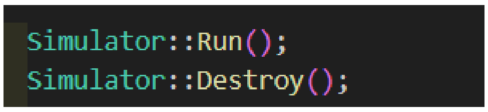
  Defines and starts UDP echo clients on Node 0 that will send packets to both servers. Each client is set to send four packets, as per the requirement.



```
// Setting up client to send to both servers
UdpEchoClientHelper echoClient1(interfaces1.GetAddress(1), port1);
echoClient1.SetAttribute("MaxPackets", UintegerValue(4));
echoClient1.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient1.SetAttribute("PacketSize", UintegerValue(1024));

UdpEchoClientHelper echoClient2(interfaces2.GetAddress(1), port2);
echoClient2.SetAttribute("MaxPackets", UintegerValue(4));
echoClient2.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient2.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps1 = echoClient1.Install(nodes.Get(0));
clientApps1.Start(Seconds(2.0));
clientApps1.Stop(Seconds(10.0));

ApplicationContainer clientApps2 = echoClient2.Install(nodes.Get(0));
clientApps2.Start(Seconds(2.0));
clientApps2.Stop(Seconds(10.0));
```

- Simulation Execution:

  Runs the simulation and then destroys it after completion.



```
Simulator::Run();
Simulator::Destroy();
```

- Questions:

1. What is the different between network simulation and emulation?

   Ans:

   The network simulation is more about theoretical modeling and analysis, being used for research and educational purposes. The network emulation, which pretend to be entire physical networks for testing actual servers and client configurations, making it suitable for final stages of network design and testing.

2. Generally, in NS-3, if you don’t change the code, the output will be always the same every time you run, even if you set some probabilistic parameter like error rate, why?

   Ans:

   In the NS-3 simulation environment, uses pseudo-random number generators (PRNGs) for probabilistic processes such as error rates. These PRNGs generate sequences of numbers that appear random but are actually determined by an initial value called a seed. If the seed value is not changed between runs, the sequence of 'random' numbers generated by the PRNG will be the same each time, leading to identical outputs for each simulation.

3. Following the previous question, how to deal with this problem?

   Ans:

   Use a different seed each time. To ensure different outcomes in each simulation run, you should set a different seed value for the random number generators. In NS-3, this can typically be done in the simulation script.

Bonus

1. What have you learned from this lab?

   Ans:

   I learned the whole steps of creating the new network flow.

2. What difficulty have you met in this lab?

   Ans:

   The difficulty I met in this lab is trying to figure out the NS-3 environment and set up the NS-3 environment.