

EXPERIMENT 4

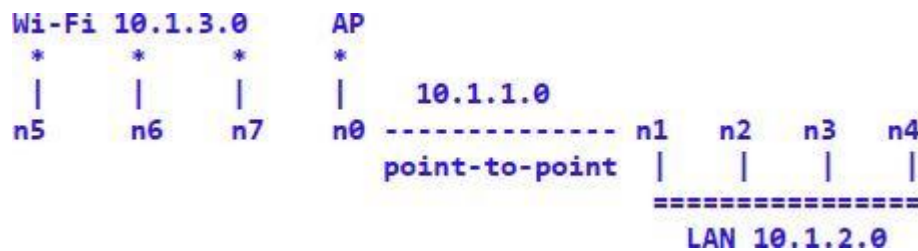
BUILDING TOPOLOGIES IN NS-3 (CSMA, WI-FI)

OBJECTIVE:

- Build and analyze the network using CSMA channel

BACKGROUND:

You have implemented simple (point to point topology) topology in lab 2 and 3. In today's lab we are going to implement point to point, CSMA (LAN) and Wi-Fi Network as shown in figure 4.1.



Step 1

Figure 4.1

First of all, you have to create simple point to point topology between two nodes, which you have already implemented in last two labs.

Step 2

Now we are going to extend our point-to-point example (the link between the nodes n0 and n1 below) by hanging a bus network off of the right side. This can be done by implement CSMA channel. Our task is to create four nodes on LAN (CSMA channel) – one required node and three extra nodes.

Step 3

You can see that we are adding a new network device to the node on the left side of the point-topoint link that becomes the access point for the wireless network. A number of wireless STA nodes are created to fill out the new 10.1.3.0 network as shown on the left side of the illustration.

Visit the following link in order to understand the code. It also provides you with the line by line explanation of the code given below: <https://www.nsnam.org/docs/release/3.40/tutorial/html/building-topologies.html>

Exercise:

Modify above code to implement the following topology shown in figure 4.2. Paste your code in the space provided below.

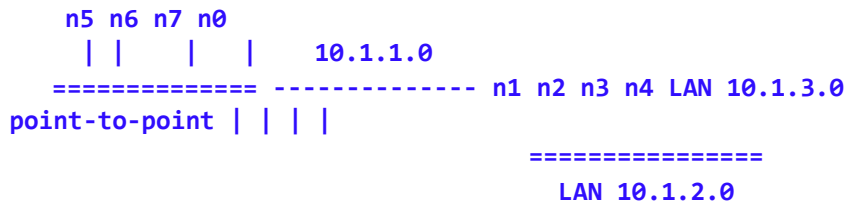


Figure 4.2

1. Set node 3 as 'UdpEchoServer' port set to 93.
2. Set node 6 as 'UdpEchoClient'.

```

1. #include "ns3/core-module.h"
   #include "ns3/network-module.h"
   #include "ns3/csma-module.h"
   #include "ns3/internet-module.h"
   #include "ns3/point-to-point-module.h"
   #include "ns3/applications-module.h"
   #include "ns3/ipv4-global-routing-helper.h"

   using namespace ns3;

   NS_LOG_COMPONENT_DEFINE ("TwoLanExample");

   int main (int argc, char *argv[])
   {
       bool verbose = true;

       CommandLine cmd;
       cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
       cmd.Parse (argc, argv);

       if (verbose)
       {
           LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
           LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
       }

       // ----- Node Creation -----
       NodeContainer p2pNodes;

       NodeContainer csmaNodes;          // right LAN 10.1.2.0 (n1, n2, n3, n4)
       csmaNodes.Create (4);

       NodeContainer csmaNodes1;        // left LAN 10.1.3.0 (n0, n5, n6, n7)
       csmaNodes1.Create (4);
       p2pNodes.Add(csmaNodes.Get(3), csmaNodes1.Get(0));

       // ----- Point-to-Point -----
       PointToPointHelper pointToPoint;
       pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
       pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

       NetDeviceContainer p2pDevices;
       p2pDevices = pointToPoint.Install (p2pNodes);

       // ----- CSMA LANs -----
       CsmHelper csma;
  
```

```

csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);           //right LAN

CsmaHelper csma1;
csma1.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma1.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices1;
csmaDevices1 = csma1.Install (csmaNodes1);       //left LAN


// ----- Internet Stack -----
InternetStackHelper stack;
stack.Install (csmaNodes);   // right LAN 10.1.2.0
stack.Install (csmaNodes1);  // left LAN 10.1.3.0


// ----- IP Addressing -----
Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");           // right LAN
Ipv4InterfaceContainer csmaInterfaces = address.Assign (csmaDevices);

address.SetBase ("10.1.3.0", "255.255.255.0");           // left LAN
Ipv4InterfaceContainer csmaInterfaces1 = address.Assign (csmaDevices1);


// ----- Applications -----
// Server on n3 (index 2 in csmaNodes)

UdpEchoServerHelper echoServer (93);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (2));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));


// Client on n6 (index 2 in csmaNodes1), targeting n3

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (2), 93);
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (csmaNodes1.Get (1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));


// ----- Routing -----
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();


// ----- Tracing -----
pointToPoint.EnablePcapAll ("two-lan");
csma.EnablePcap ("two-lan", csmaDevices.Get (1), true);
csma1.EnablePcap ("two-lan", csmaDevices1.Get (1), true);


// ----- Run Simulation -----
Simulator::Run ();
Simulator::Destroy ();
return 0;
}

```

```

linuxuban@linuxuban-VMware-Virtual-Platform: ~/Desktop/ns-allinone-3.40/ns-3.40$
./ns3 run scratch/lab4t1.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable /home/lin...40/build/scratch/ns3.40-lab4t1-default
At time +2s client sent 1024 bytes to 10.1.2.3 port 93
At time +2.01292s server received 1024 bytes from 10.1.3.2 port 49153
At time +2.01292s server sent 1024 bytes to 10.1.3.2 port 49153
At time +2.02484s client received 1024 bytes from 10.1.2.3 port 93
linuxuban@linuxuban-VMware-Virtual-Platform:~/Desktop/ns-allinone-3.40/ns-3.40$

```

POST LAB QUESTIONS:

Modify above code to implement the following topology shown in figure 4.3 and paste the modified code in the space provided below.

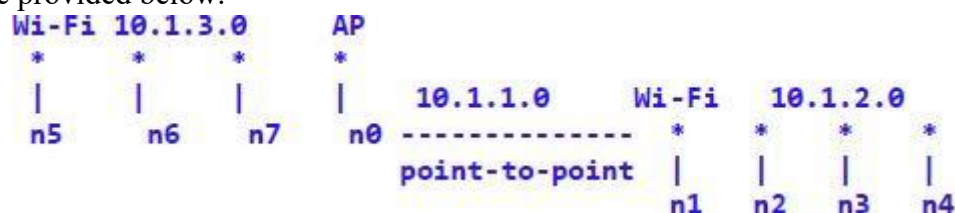


Figure 4.3

1. Set node 3 as 'UdpEchoServer' port set to 93.
2. Set node 6 as 'UdpEchoClient'.

```
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("WifiToWifiExample");

int main (int argc, char *argv[])
{
    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

    // ----- Nodes -----
    NodeContainer p2pNodes;
    p2pNodes.Create (2); // n0 (AP left), n1 (AP right)

    NodeContainer wifiStaLeft;
    wifiStaLeft.Create (3); // n5, n6, n7

    NodeContainer wifiStaRight;
    wifiStaRight.Create (3); // n2, n3, n4

    // ----- Point-to-Point -----
    PointToPointHelper p2p;
    p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
    NetDeviceContainer p2pDevices = p2p.Install (p2pNodes);

    // ----- WiFi Left -----

    WifiHelper wifiLeft;
    wifiLeft.SetRemoteStationManager ("ns3::MinstrelHtWifiManager");

    YansWifiChannelHelper channelLeft = YansWifiChannelHelper::Default ();
    YansWifiPhyHelper phyLeft;
    phyLeft.SetChannel (channelLeft.Create ());

    WifiMacHelper macLeft;
    Ssid ssidLeft = Ssid ("wifi-left");

    macLeft.SetType ("ns3::StaWifiMac",
                    "Ssid", SsidValue (ssidLeft),
                    "ActiveProbing", BooleanValue (false));

    NetDeviceContainer staDevicesLeft = wifiLeft.Install (phyLeft, macLeft, wifiStaLeft);

    macLeft.SetType ("ns3::ApWifiMac",
                    "Ssid", SsidValue (ssidLeft));

    NetDeviceContainer apDeviceLeft = wifiLeft.Install (phyLeft, macLeft, p2pNodes.Get (0));

    // ----- WiFi Right -----

    WifiHelper wifiRight;
    wifiRight.SetRemoteStationManager ("ns3::MinstrelHtWifiManager");

    YansWifiChannelHelper channelRight = YansWifiChannelHelper::Default ();

    YansWifiPhyHelper phyRight;
    phyRight.SetChannel (channelRight.Create ());

    WifiMacHelper macRight;
    Ssid ssidRight = Ssid ("wifi-right");
```

```

macRight.SetType ("ns3::StaWifiMac",
                  "Ssid", SsidValue (ssidRight),
                  "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevicesRight = wifiRight.Install (phyRight, macRight, wifiStaRight);

macRight.SetType ("ns3::ApWifiMac",
                  "Ssid", SsidValue (ssidRight));
NetDeviceContainer apDeviceRight = wifiRight.Install (phyRight, macRight, p2pNodes.Get (1));


// ----- Mobility -----

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                              "MinX", DoubleValue (0.0),
                              "MinY", DoubleValue (0.0),
                              "DeltaX", DoubleValue (5.0),
                              "DeltaY", DoubleValue (10.0),
                              "GridWidth", UIntegerValue (3),
                              "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));


mobility.Install (wifiStaLeft);
mobility.Install (wifiStaRight);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (p2pNodes);


// ----- Internet Stack -----

InternetStackHelper stack;
stack.Install (wifiStaLeft);
stack.Install (wifiStaRight);
stack.Install (p2pNodes);


// ----- IP Addressing -----

Ipv4AddressHelper address;

// P2P (between APs)

address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces = address.Assign (p2pDevices);

// Left WiFi

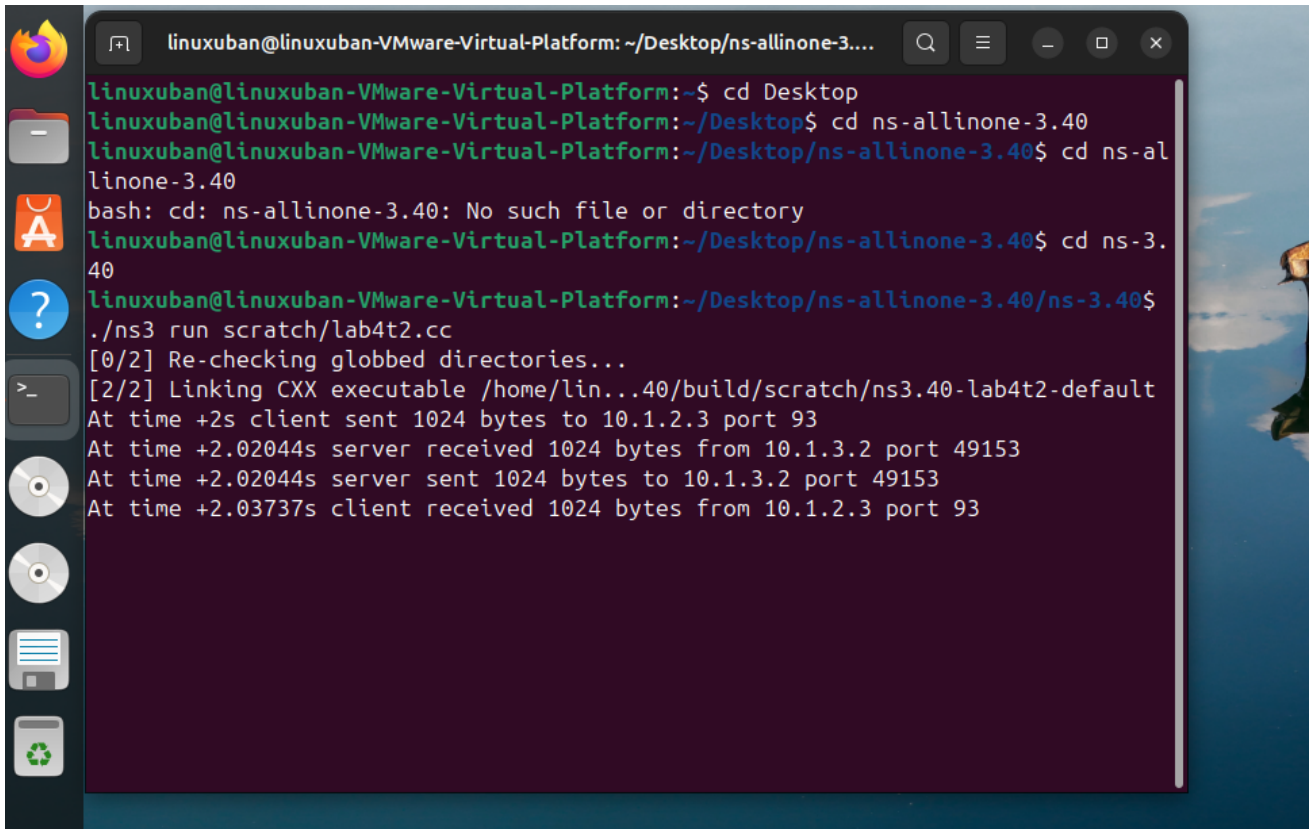
address.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer staIfLeft = address.Assign (staDevicesLeft);
Ipv4InterfaceContainer apIfLeft = address.Assign (apDeviceLeft);

// Right WiFi

address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer staIfRight = address.Assign (staDevicesRight);
Ipv4InterfaceContainer apIfRight = address.Assign (apDeviceRight);

```

```
// ----- Applications -----  
// Server on n3 (STA of right WiFi)  
UdpEchoServerHelper echoServer (93);  
ApplicationContainer serverApps = echoServer.Install (wifiStaRight.Get (2)); // n3  
serverApps.Start (Seconds (1.0));  
serverApps.Stop (Seconds (10.0));  
  
// Client on n6 (STA of left WiFi) → server n3  
UdpEchoClientHelper echoClient (staIfRight.GetAddress (2), 93);  
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));  
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));  
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));  
  
ApplicationContainer clientApps = echoClient.Install (wifiStaLeft.Get (1)); // n6  
clientApps.Start (Seconds (2.0));  
clientApps.Stop (Seconds (10.0));  
  
// ----- Routing -----  
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();  
  
// ----- Run Simulation -----  
  
Simulator::Run ();  
Simulator::Destroy ();  
return 0;  
}
```

A terminal window titled 'linuxuban@linuxuban-VMware-Virtual-Platform: ~/Desktop/ns-allinone-3....' is shown. The terminal output displays a series of directory changes and a network test execution. The test results show a successful data transfer of 1024 bytes between a client and a server on different ports (93 and 49153) at specific timestamps.

```
linuxuban@linuxuban-VMware-Virtual-Platform:~$ cd Desktop
linuxuban@linuxuban-VMware-Virtual-Platform:~/Desktop$ cd ns-allinone-3.40
linuxuban@linuxuban-VMware-Virtual-Platform:~/Desktop/ns-allinone-3.40$ cd ns-allinone-3.40
bash: cd: ns-allinone-3.40: No such file or directory
linuxuban@linuxuban-VMware-Virtual-Platform:~/Desktop/ns-allinone-3.40$ cd ns-3.40
linuxuban@linuxuban-VMware-Virtual-Platform:~/Desktop/ns-allinone-3.40/ns-3.40$ ./ns3 run scratch/lab4t2.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable /home/lin...40/build/scratch/ns3.40-lab4t2-default
At time +2s client sent 1024 bytes to 10.1.2.3 port 93
At time +2.02044s server received 1024 bytes from 10.1.3.2 port 49153
At time +2.02044s server sent 1024 bytes to 10.1.3.2 port 49153
At time +2.03737s client received 1024 bytes from 10.1.2.3 port 93
```