**Lab2 Building Topologies**

1. In this section we are going to add a bus network to the point-to-point network we built in myfirst.cc, so the network topology will look like the following.

// Default Network Topology

//

// 10.1.1.0

// n0 ------------------------ n1 n2 n3 n4

// point-to-point | | | |

// ================

// LAN 10.1.2.0

Create an empty file mysecond.cc in scratch directory.

1.1 Add the following include files and namespace to mysecond.cc.

#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 ("SecondScriptExample");

1.2 The main program starts with the following. This code will allow you to turn off logging and change the number of devices on the CSMA network via command line argument.

**int**

**main (int argc, char \*argv[])**

**{**

**bool** verbose = true;

**uint32\_t** nCsma = 3;

CommandLine cmd;

cmd.AddValue ("nCsma", "Number of **\"**extra**\"** CSMA nodes/devices", nCsma);

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);

}

nCsma = nCsma == 0 ? 1 : nCsma; // makes sure you have at least one “extra” node.

* 1. The next step is to create network node including two nodes that we will connect via the point-to-point link, and CSMA nodes on the bus network.

NodeContainer p2pNodes;

p2pNodes.Create (2);

NodeContainer csmaNodes;

csmaNodes.Add (p2pNodes.Get (1));

csmaNodes.Create (nCsma);

The code “csmaNodes.Add (p2pNodes.Get (1));” Gets the second node (having an index of one) from the point-to-point node container and adds it to the container of nodes that will get CSMA devices. The node in question is going to end up with a point-to-point device and a CSMA device.

* 1. Next install network devices (and channels) to p2p nodes and CSMA nodes, respectively

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));

csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

* 1. The next step is to install internet protocol stack and assign IP addresses

InternetStackHelper stack;

stack.Install (p2pNodes.Get (0));

stack.Install (csmaNodes);

Recall that we took one of the nodes from the p2pNodes container and added it to the csmaNodes container. Thus we only need to install the stacks on the remaining p2pNodes node, and all of the nodes in the csmaNodes container to cover all of the nodes in the simulation.

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer p2pInterfaces;

p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer csmaInterfaces;

csmaInterfaces = address.Assign (csmaDevices);

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Note that these are two different networks, so it is necessary to turn on global routing.

* 1. Now it is time to create and install applications.

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

* 1. Next we enable pcap tracing

pointToPoint.EnablePcapAll ("second");

csma.EnablePcap ("second", csmaDevices.Get (1), true);

The first line enable pcap tracing on all PoitToPoint nodes. The second line enables Pcap tracing on the second device (with index 1) in the csmaDevices container. Setting the third parameter to true enables promiscuous captures (sniffing).

* 1. The last section of code just runs and cleans up the simulation just like the first.cc example.

Simulator::Run ();

Simulator::Destroy ();

return 0;

}

* 1. Compile and run the script

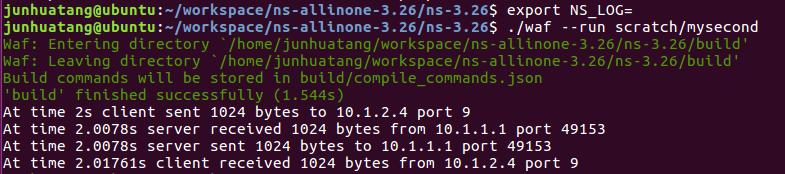
$./waf

$export NS\_LOG= //clear NS\_LOG setting

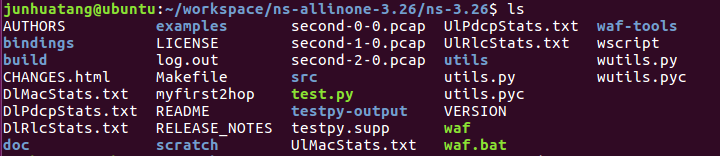
$rm \*.tr //delete all the .tr file in the current directory

$rm \*.pcap //delete all the .pcap file in the current directory

$./waf --run scratch/mysecond

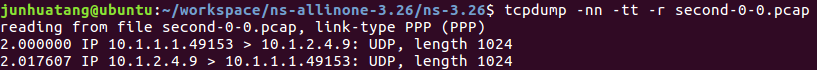


$ls //list the files in the current directory

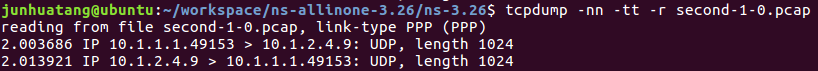


Note that second-0-0.pcap and second-1-0.pcap are produced by pointToPoint.EnablePcapAll ("second"); and second-2-0.pcap is produced by csma.EnablePcap ("second", csmaDevices.Get (1), true);

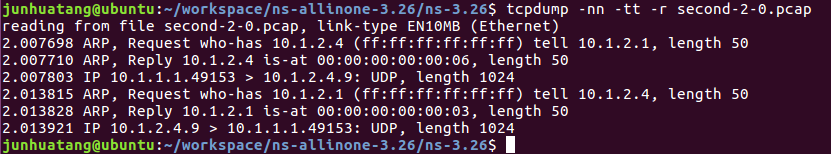
$ tcpdump -nn -tt -r second-0-0.pcap



$ tcpdump -nn -tt -r second-1-0.pcap



$ tcpdump -nn -tt -r second-2-0.pcap



1. Build a 2-hop Point-to-Point network as illustrated below,

192.168.10.0 192.168.50.0

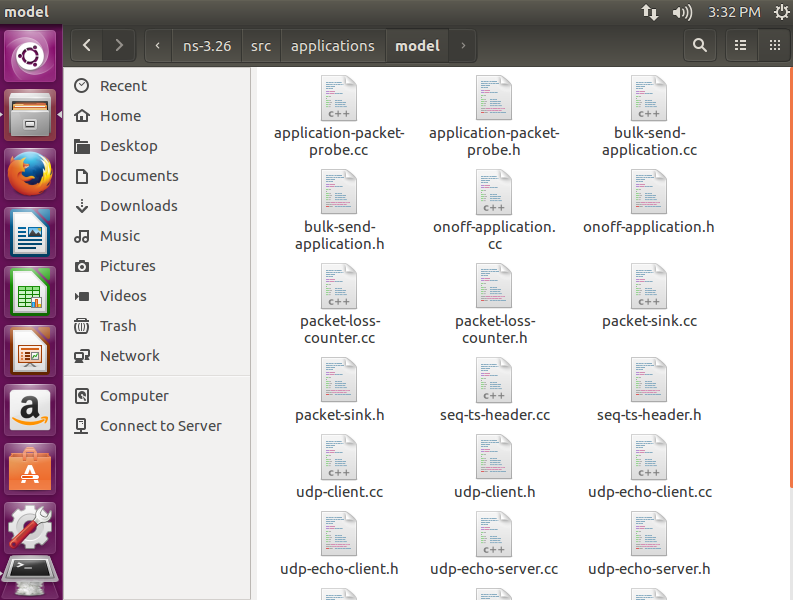
n0 ------------------------ n1-------------------------------- n2

point-to-point point-to-point

5Mbps, 2ms 1Mbps, 2ms

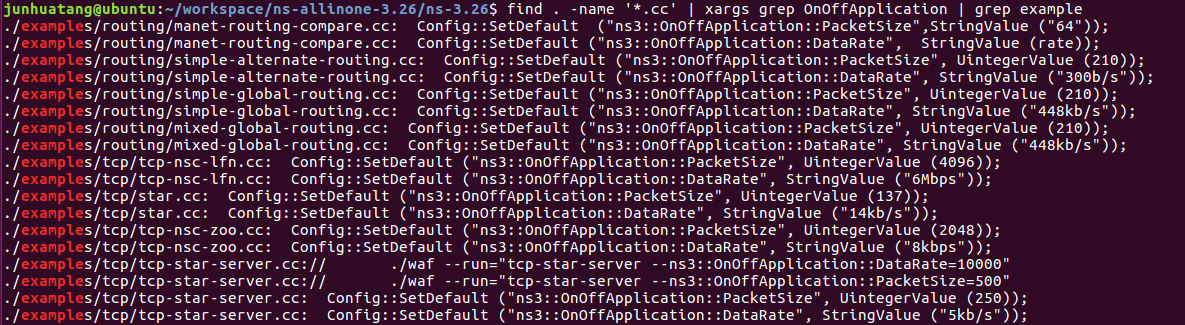
* n0 sends a total number of 2000 bytes to n2.
* Use onoff-application (TCP) on n0 with packet size 512, data rate 50kb/s, set the OnTime random variable to 1 and OffTime random variable to 0.
* Use PacketSink (TCP) application on n2 to receive the packets.
* Enable NS\_LOG on both onoff-application and PacketSink, turn on pacp tracing on all nodes.
* Use filename: lab2.cc

Hints: For onoff-application and packet-sink application, the source codes are in /home/workspace/ns-allinone-3.28/ns-3.28/src/applications/model. Or go to <https://www.nsnam.org/doxygen/index.html>, click Modules→Applications→OnOffApplication to see a detailed description of the application.



Find an example of using OnOffApplication.

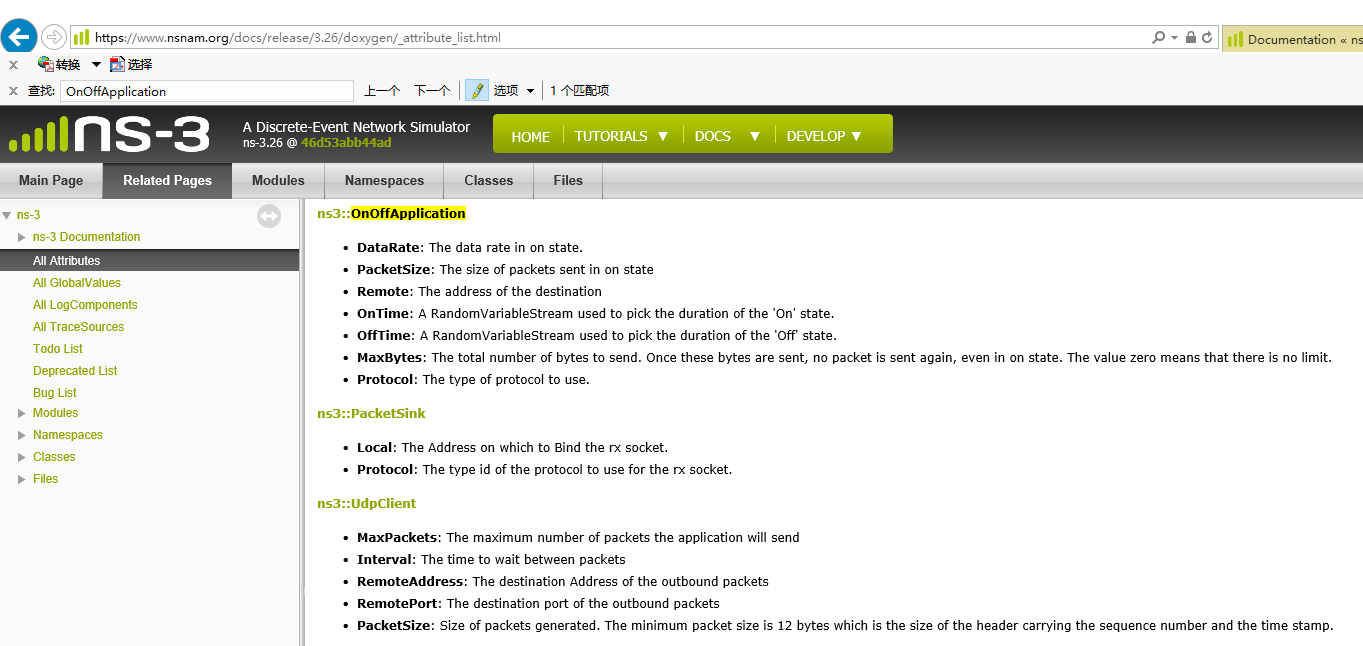
$find . -name '\*.cc' | xargs grep OnOffApplication | grep example



Open a file from the list (e.g. ./examples/tcp/star.cc) and see how OnOffApplication is created and installed.

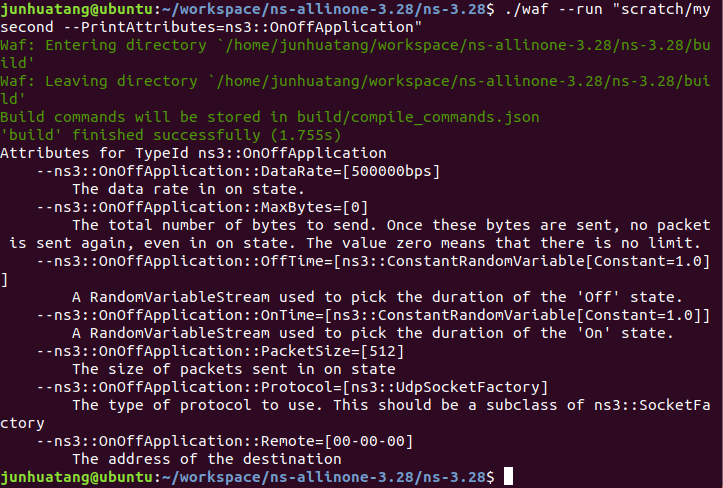
To check the attributes of OnOffApplication:

<https://www.nsnam.org/docs/release/3.28/doxygen/_attribute_list.html>

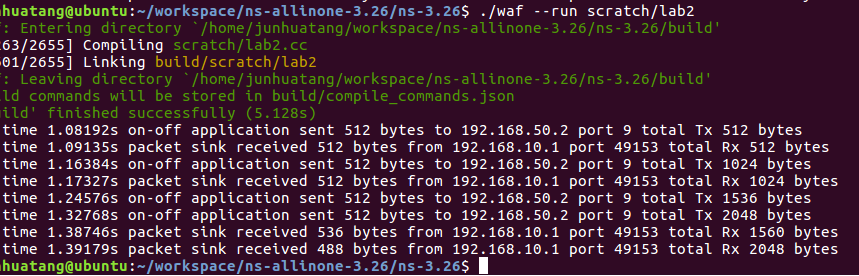


To check the default values of the attributes:

$./waf --run "scratch/mysecond --PrintAttributes=ns3::OnOffApplication"



$./waf --run scratch/lab2



**Lab 2 turn in：**

1. lab2.cc
2. lab2-report.doc (lab2-report.pdf)
   1. screenshot of simulation output
   2. pcap file contents to show that packets are delivered to the destination