Computer Networks Lab 07

Course: Computer Networks (CL3001) **Semester:** Spring 2024
Instructor: Sameer Faisal **T.A:** N/A

Note:

- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Lab Objective

- Introduction to NS3 network simulator.
- Simple network simulation in NS3.
- Implementation of TCP congestion control & simulation in NS3.

NS3

1. Introduction

NS3 is a discrete event simulator targeted at networking research. NS3 provides substantial support for simulation of TCP, UDP, routing & multicast protocols over wired & wireless (local & satellite) networks. This simulator is primarily UNIX based & the NS commands can either be entered via UNIX command prompt or by running a scripting language file, but scripting language is preferred way. NS3 uses Tcl (pronounced as 'tickle') as it's scripting language, which is edited using a text editor & in this example we will use vi, gedit or nano.

2. Introduction to vi

vi is a text editor that is available in all UNIX systems, other editors such pico & emacs can also be used instead. Before you start for the first time, you must note that vi is modal editor. A mode is like an environment. Different modes in vi interpret the same key differently. For example, if you're in insert mode, typing a adds an a to the text, whereas in command mode, typing a put you in insert mode because a is the key abbreviation for the append command. If you get confused about what mode you're in, press the Escape key. Pressing Escape key always returns you to the command mode & if you are already in command mode, it simply beeps to remind you of the fact.

When you are in command mode, you can manage your document; this includes the capability to change text, rearrange it, & delete it. Insert mode is when you are maneuver the text area using the arrows keys.

For starting a new document, simply type vi after the command prompt to the start the vi editor. The cursor will be located in the top left corner & each of the following lines will start with a tilde (~) denoting empty lines. Note that the vi editor is currently in command mode. In order to edit an existing file, type the name

of the file along with the extension after typing vi & a space in the UNIX command prompt. To enter into insert mode simply press a on the keyboard. Then to save the edited document, type a colon (:) & the cursor will be located in the bottom left corner after the colon, followed by a w & then press enter. In order to quit vi editor type a colon followed by a q (:q), this will take you back to the UNIX prompt. You can also choose to quit editing without saving by typing :q!, & if you forget the exclamation mark (!) the system will issue a warning.

Note: direct link for NS 3 image:
https://www.nsnam.com/2020/04/download-vm-image-of-ns3-and-contiking.html

3. Simple Simulation in NS3

First.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int main (int argc, char *argv[])
{
CommandLine cmd;

cmd.Parse (argc, argv);
```

```
Time::SetResolution (Time::NS);
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0))
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
```

```
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Simulator::Run ();

Simulator::Destroy ();

return 0;
}
```

Second.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#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"
#include "ns3/netanim-module.h"
```

```
// Default Network Topology
//
// 10.1.1.0
// n0 ---------- n1 n2 n3 n4
// point-to-point | | | |
// ================
// LAN 10.1.2.0
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

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;
 NodeContainer p2pNodes;
 p2pNodes.Create (2);
 NodeContainer csmaNodes;
 csmaNodes.Add (p2pNodes.Get (1));
 csmaNodes.Create (nCsma);
 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);

InternetStackHelper stack;
 stack.Install (p2pNodes.Get (0));
 stack.Install (csmaNodes);
 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);
 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));
 Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
 pointToPoint.EnablePcapAll ("second");
 csma.EnablePcap ("second", csmaDevices.Get (1), true);

AnimationInterface anim ("second.xml");
 Simulator::Run ();
 Simulator::Destroy ();
 return 0;
}
```

4. Code Explanation

The following is the explanation of the script above. In general, an NS script starts with making a Simulator object instance.

- set ns [new Simulator]: generates an NS simulator object instance, and assigns it to variable ns (italics is used for variables and values in this section).
  What this line does is the following: Initialize the packet format (ignore this for now).
  - Create a scheduler (default is calendar scheduler).
  - Select the default address format (ignore this for now)

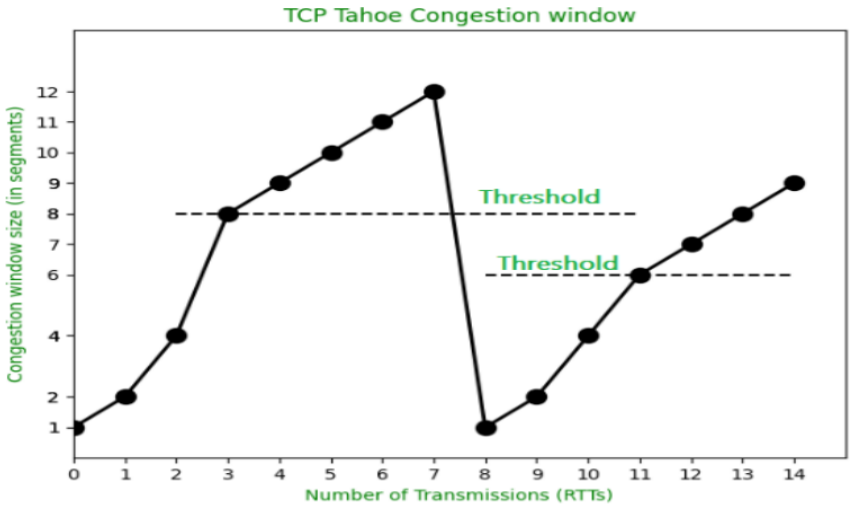
- The "Simulator" object has member functions that do the following:
  - Create compound objects such as nodes and links (described later).
  - Connect network component objects created (ex. attach-agent).
  - Set network component parameters (mostly for compound objects).
  - Create connections between agents (ex. make connection between a "tcp" and "sink").
  - Specify NAM display options.

Most of member functions are for simulation setup (referred to as plumbing functions) and scheduling, however some of them are for the NAM display.
The "Simulator" object memberfunction implementations are located in the "ns-2/tcl/lib/nslib.tcl" file.

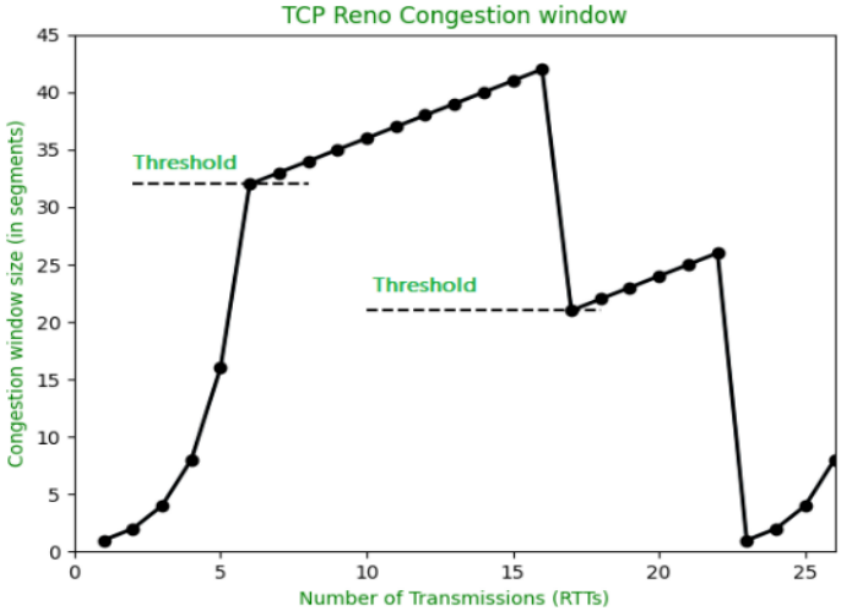
TCP Congestion Control Simulation

In this example we will implement & simulate congestion control mechanism of TCP between two nodes. It consists of three parts which are as follows:

1. Slow start threshold (exponential increase of packets) ssthresh.
2. Congestion avoidance (additive increase).
3. Congestion detection (multiplicative decrease).



TCP Tahoe



TCP Reno

Lab Exercise

1. Apply TCP Congestion control mechanism in NS3 & plot the congestion window graph.

Helpful link for TCP congestion Task:
https://www.youtube.com/watch?v=9rkN3FtOkaQ&t=885s