**Program to simulate UDP Client Server**

CODE :

/\* -\*- 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

\*/

// Network topology

//

// n0 n1 n2 n3

// | | | |

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

// LAN

//

// - UDP flows from n0 to n1 and back

// - DropTail queues

// - Tracing of queues and packet receptions to file "udp-echo.tr"

#include <fstream>

#include "ns3/core-module.h"

#include "ns3/csma-module.h"

#include "ns3/applications-module.h"

#include "ns3/internet-module.h"

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("UdpEchoExample");

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

{

//

// Users may find it convenient to turn on explicit debugging

// for selected modules; the below lines suggest how to do this

//

#if 0

LogComponentEnable ("UdpEchoExample", LOG\_LEVEL\_INFO);

LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_ALL);

LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_ALL);

#endif

//

// Allow the user to override any of the defaults and the above Bind() at

// run-time, via command-line arguments

//

bool useV6 = false;

Address serverAddress;

CommandLine cmd;

cmd.AddValue ("useIpv6", "Use Ipv6", useV6);

cmd.Parse (argc, argv);

//

// Explicitly create the nodes required by the topology (shown above).

//

NS\_LOG\_INFO ("Create nodes.");

NodeContainer n;

n.Create (4);

InternetStackHelper internet;

internet.Install (n);

NS\_LOG\_INFO ("Create channels.");

//

// Explicitly create the channels required by the topology (shown above).

//

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));

csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));

NetDeviceContainer d = csma.Install (n);

//

// We've got the "hardware" in place. Now we need to add IP addresses.

//

NS\_LOG\_INFO ("Assign IP Addresses.");

if (useV6 == false)

{

Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (d);

serverAddress = Address(i.GetAddress (1));

}

else

{

Ipv6AddressHelper ipv6;

ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));

Ipv6InterfaceContainer i6 = ipv6.Assign (d);

serverAddress = Address(i6.GetAddress (1,1));

}

NS\_LOG\_INFO ("Create Applications.");

//

// Create a UdpEchoServer application on node one.

//

uint16\_t port = 9; // well-known echo port number

UdpEchoServerHelper server (port);

ApplicationContainer apps = server.Install (n.Get (1));

apps.Start (Seconds (1.0));

apps.Stop (Seconds (10.0));

//

// Create a UdpEchoClient application to send UDP datagrams from node zero to

// node one.

//

uint32\_t packetSize = 1024;

uint32\_t maxPacketCount = 1;

Time interPacketInterval = Seconds (1.);

UdpEchoClientHelper client (serverAddress, port);

client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));

client.SetAttribute ("Interval", TimeValue (interPacketInterval));

client.SetAttribute ("PacketSize", UintegerValue (packetSize));

apps = client.Install (n.Get (0));

apps.Start (Seconds (2.0));

apps.Stop (Seconds (10.0));

#if 0

//

// Users may find it convenient to initialize echo packets with actual data;

// the below lines suggest how to do this

//

client.SetFill (apps.Get (0), "Hello World");

client.SetFill (apps.Get (0), 0xa5, 1024);

uint8\_t fill[] = { 0, 1, 2, 3, 4, 5, 6};

client.SetFill (apps.Get (0), fill, sizeof(fill), 1024);

#endif

AsciiTraceHelper ascii;

csma.EnableAsciiAll (ascii.CreateFileStream ("udp-echo.tr"));

csma.EnablePcapAll ("udp-echo", false);

//

// Now, do the actual simulation.

//

NS\_LOG\_INFO ("Run Simulation.");

Simulator::Run ();

Simulator::Destroy ();

NS\_LOG\_INFO ("Done.");

}

OUTPUT/SCREENSHOTS :

