UDP CLIENT SERVER

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2009 INRIA
 *
 * 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
 *
 * Author: Mohamed Amine Ismail <amine.ismail@sophia.inria.fr>
 */

// Network topology
//
//       n0    n1
//       |     |
//       =======
//         LAN (CSMA)
//
// - UDP flow from n0 to n1 of 1024 byte packets at intervals of 50 ms
//   - maximum of 320 packets sent (or limited by simulation duration)
//   - option to use IPv4 or IPv6 addressing
//   - option to disable logging statements

#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample");

int
main (int argc, char *argv[])
{
  // Declare variables used in command-line arguments
  bool useV6 = false;
  bool logging = true;
```

```
Address serverAddress;

CommandLine cmd (__FILE__);
cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
cmd.AddValue ("logging", "Enable logging", logging);
cmd.Parse (argc, argv);

if (logging)
  {
    LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
  }

NS_LOG_INFO ("Create nodes in above topology.");
NodeContainer n;
n.Create (4);

InternetStackHelper internet;
internet.Install (n);

NS_LOG_INFO ("Create channel between the two nodes.");
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
NetDeviceContainer d = csma.Install (n);

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));
  }
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NS_LOG_INFO ("Create UdpServer application on node 1.");
uint16_t port = 4000;
UdpServerHelper server (port);
ApplicationContainer apps = server.Install (n.Get (1));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));
```

```
  NS_LOG_INFO ("Create UdpClient application on node 0 to send to node 1.");
  uint32_t MaxPacketSize = 1024;
  Time interPacketInterval = Seconds (0.05);
  uint32_t maxPacketCount = 320;
  UdpClientHelper client (serverAddress, port);
  client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
  client.SetAttribute ("Interval", TimeValue (interPacketInterval));
  client.SetAttribute ("PacketSize", UintegerValue (MaxPacketSize));
  apps = client.Install (n.Get (0));
  apps.Start (Seconds (2.0));
  apps.Stop (Seconds (10.0));

MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(n);
AnimationInterface anim("udp-client-server.xml");
AnimationInterface::SetConstantPosition(n.Get(0),10,25);
AnimationInterface::SetConstantPosition(n.Get(1),40,35);
AnimationInterface::SetConstantPosition(n.Get(2),60,45);
AnimationInterface::SetConstantPosition(n.Get(3),80,55);
anim.EnablePacketMetadata(true);
pointToPoint.EnablePcapAll("udp-client-server");
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```