Amrit Pandey
207907, MCA Year 2

CCN lab Cycle 1, Assignment 4

Source Code

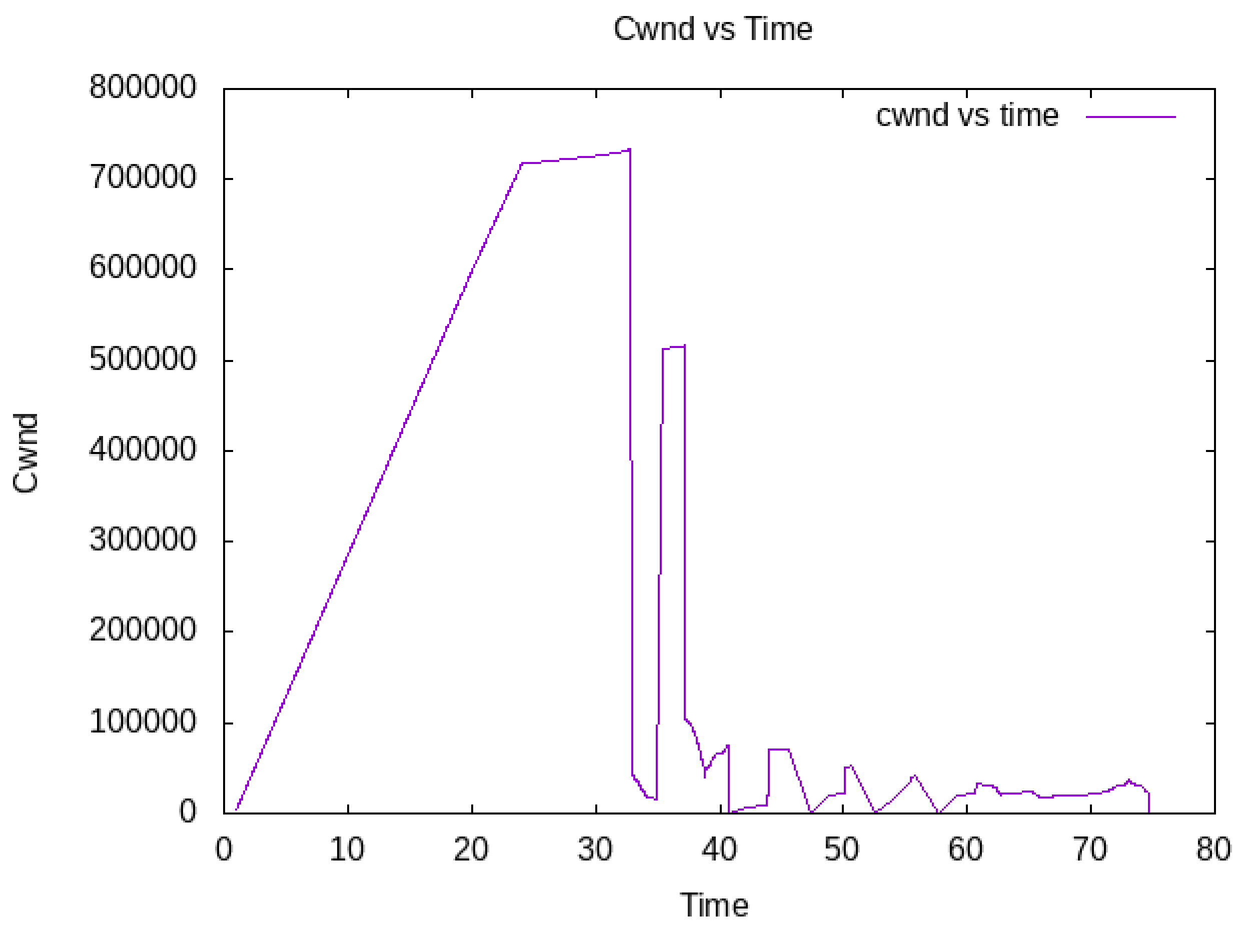
```
#include <fstream>
#include <string>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Lab2");

class MyApp : public Application
{
public:

  MyApp ();
  virtual ~MyApp();

  void Setup (Ptr<Socket> socket, Address address, uint32_t
packetSize, uint32_t nPackets, DataRate dataRate);
  void ChangeRate(DataRate newrate);

private:
  virtual void StartApplication (void);
  virtual void StopApplication (void);

  void ScheduleTx (void);
  void SendPacket (void);

  Ptr<Socket>     m_socket;
  Address         m_peer;
  uint32_t        m_packetSize;
  uint32_t        m_nPackets;
  DataRate        m_dataRate;
  EventId         m_sendEvent;
  bool            m_running;
  uint32_t        m_packetsSent;
};

MyApp::MyApp ()
  : m_socket (0),
    m_peer (),
    m_packetSize (0),
    m_nPackets (0),
    m_dataRate (0),
    m_sendEvent (),
    m_running (false),
    m_packetsSent (0)
{
}

MyApp::~MyApp()
{
  m_socket = 0;
}

void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t
packetSize, uint32_t nPackets, DataRate dataRate)
{
  m_socket = socket;
  m_peer = address;
  m_packetSize = packetSize;
  m_nPackets = nPackets;
  m_dataRate = dataRate;
}

void
MyApp::StartApplication (void)
{
  m_running = true;
  m_packetsSent = 0;
  m_socket->Bind ();
  m_socket->Connect (m_peer);
  SendPacket ();
}

void
MyApp::StopApplication (void)
{
  m_running = false;
  if (m_sendEvent.IsRunning ())
    {
      Simulator::Cancel (m_sendEvent);
    }

  if (m_socket)
    {
      m_socket->Close ();
    }
}

void
MyApp::SendPacket (void)
{
  Ptr<Packet> packet = Create<Packet> (m_packetSize);
  m_socket->Send (packet);

  if (++m_packetsSent < m_nPackets)
    {
      ScheduleTx ();
    }
}

void
MyApp::ScheduleTx (void)
{
  if (m_running)
    {
      Time tNext (Seconds (m_packetSize * 8 / static_cast<double>
(m_dataRate.GetBitRate ())));
      m_sendEvent = Simulator::Schedule (tNext,
&MyApp::SendPacket, this);
    }
}

void
MyApp::ChangeRate(DataRate newrate)
{
   m_dataRate = newrate;
   return;
}

static void
CwndChange (uint32_t oldCwnd, uint32_t newCwnd)
```

```
{
  std::cout << Simulator::Now ().GetSeconds () << "\t" << newCwnd
<<"\n";
}

void
IncRate (Ptr<MyApp> app, DataRate rate)
{
     app->ChangeRate(rate);
    return;
}


int main (int argc, char *argv[])
{
  std::string lat = "2ms";
  std::string rate = "500kb/s"; // P2P link
  bool enableFlowMonitor = false;


  CommandLine cmd;
  cmd.AddValue ("latency", "P2P link Latency in miliseconds",
lat);
  cmd.AddValue ("rate", "P2P data rate in bps", rate);
  cmd.AddValue ("EnableMonitor", "Enable Flow Monitor",
enableFlowMonitor);

  cmd.Parse (argc, argv);

  NS_LOG_INFO ("Create nodes.");
  NodeContainer c;
  c.Create(6);

  NodeContainer n0n4 = NodeContainer (c.Get (0), c.Get (4));
  NodeContainer n1n4 = NodeContainer (c.Get (1), c.Get (4));
  NodeContainer n2n5 = NodeContainer (c.Get (2), c.Get (5));
  NodeContainer n3n5 = NodeContainer (c.Get (3), c.Get (5));
  NodeContainer n4n5 = NodeContainer (c.Get (4), c.Get (5));

  InternetStackHelper internet;
  internet.Install (c);

  NS_LOG_INFO ("Create channels.");
  PointToPointHelper p2p;
  p2p.SetDeviceAttribute ("DataRate", StringValue (rate));
  p2p.SetChannelAttribute ("Delay", StringValue (lat));
  NetDeviceContainer d0d4 = p2p.Install (n0n4);
  NetDeviceContainer d1d4 = p2p.Install (n1n4);
  NetDeviceContainer d4d5 = p2p.Install (n4n5);
  NetDeviceContainer d2d5 = p2p.Install (n2n5);
  NetDeviceContainer d3d5 = p2p.Install (n3n5);

  NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i0i4 = ipv4.Assign (d0d4);

  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer i1i4 = ipv4.Assign (d1d4);

  ipv4.SetBase ("10.1.3.0", "255.255.255.0");
  Ipv4InterfaceContainer i4i5 = ipv4.Assign (d4d5);

  ipv4.SetBase ("10.1.4.0", "255.255.255.0");
  Ipv4InterfaceContainer i2i5 = ipv4.Assign (d2d5);

  ipv4.SetBase ("10.1.5.0", "255.255.255.0");
  Ipv4InterfaceContainer i3i5 = ipv4.Assign (d3d5);

  NS_LOG_INFO ("Enable static global routing.");
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  NS_LOG_INFO ("Create Applications.");

  uint16_t sinkPort = 8080;
  Address sinkAddress (InetSocketAddress (i2i5.GetAddress (0),
sinkPort)); // interface of n2
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps = packetSinkHelper.Install (c.Get
(2)); //n2 as sink
  sinkApps.Start (Seconds (0.));
  sinkApps.Stop (Seconds (100.));

  Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (c.Get (0),
TcpSocketFactory::GetTypeId ()); //source at n0

  // Trace Congestion window
  ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow",
MakeCallback (&CwndChange));
```

```
  // Create TCP application at n0
  Ptr<MyApp> app = CreateObject<MyApp> ();
  app->Setup (ns3TcpSocket, sinkAddress, 1040, 100000, DataRate
("250Kbps"));
  c.Get (0)->AddApplication (app);
  app->SetStartTime (Seconds (1.));
  app->SetStopTime (Seconds (100.));


  // UDP connfection from N1 to N3

  uint16_t sinkPort2 = 6;
  Address sinkAddress2 (InetSocketAddress (i3i5.GetAddress (0),
sinkPort2)); // interface of n3
  PacketSinkHelper packetSinkHelper2 ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort2));
  ApplicationContainer sinkApps2 = packetSinkHelper2.Install
(c.Get (3)); //n3 as sink
  sinkApps2.Start (Seconds (0.));
  sinkApps2.Stop (Seconds (100.));

  Ptr<Socket> ns3UdpSocket = Socket::CreateSocket (c.Get (1),
UdpSocketFactory::GetTypeId ()); //source at n1

  // Create UDP application at n1
  Ptr<MyApp> app2 = CreateObject<MyApp> ();
  app2->Setup (ns3UdpSocket, sinkAddress2, 1040, 100000, DataRate
("250Kbps"));
  c.Get (1)->AddApplication (app2);
  app2->SetStartTime (Seconds (20.));
  app2->SetStopTime (Seconds (100.));

// Increase UDP Rate
  Simulator::Schedule (Seconds(30.0), &IncRate, app2,
DataRate("500kbps"));

  // Flow Monitor
  Ptr<FlowMonitor> flowmon;
  if (enableFlowMonitor)
    {
      FlowMonitorHelper flowmonHelper;
      flowmon = flowmonHelper.InstallAll ();
    }
```

```
//
// Now, do the actual simulation.
//
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds(100.0));
  Simulator::Run ();
  if (enableFlowMonitor)
    {
      flowmon->CheckForLostPackets ();
      flowmon->SerializeToXmlFile("lab-2.flowmon", true, true);
    }
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

<u>Output:</u>


Cwnd vs Time
cwnd vs time
Cwnd
Time