Mawlana Bhashani Science and Technology University

Santosh, Tangail-1902.

Lab Report

Department of Information and Communication Technology

Report No: 02
Report Name: TCP Variants

Course Title: Wireless and Mobile Communication Lab
Course Code: ICT-4202

Submitted By	Submitted To
Name: **Md. Fahim Ferdous Khan** ID: **IT-16018** Session: 2015-16 4th Year 2nd Semester Dept. of Information & Communication Technology, MBSTU.	Nazrul Islam Assistant Professor Dept. of Information & Communication Technology, MBSTU.

Submission Date: 11-09-2020

Objective:

TCP is a widely applied solution as it guarantees the delivery of data by implementing acknowledgement based techniques and used in wired networks as it ensures guard against the common remedy congestion in networks. Deploying TCP over wireless networks, restrictions such as random link errors, random packet loss, node distance, constantly changing topology are taken in account. Many solutions exist for the performance analysis in different environments and picking the suitable one in required environment. The afore-mentioned variants of TCP are revived.

Source 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
 */

#include <fstream>
#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 ("FifthScriptExample");

//
============================================================================
========
//
//        node 0                node 1
//   +----------------+    +----------------+
//   |    ns-3 TCP    |    |    ns-3 TCP    |
```

```
//  +----------------+    +----------------+
//  |    10.1.1.1    |    |    10.1.1.2    |
//  +----------------+    +----------------+
//  | point-to-point |    | point-to-point |
//  +----------------+    +----------------+
//          |                     |
//          +---------------------+
//              5 Mbps, 2 ms
//
//
// We want to look at changes in the ns-3 TCP congestion window.  We need
// to crank up a flow and hook the CongestionWindow attribute on the socket
// of the sender.  Normally one would use an on-off application to generate a
// flow, but this has a couple of problems.  First, the socket of the on-off
// application is not created until Application Start time, so we wouldn't be
// able to hook the socket (now) at configuration time.  Second, even if we
// could arrange a call after start time, the socket is not public so we
// couldn't get at it.
//
// So, we can cook up a simple version of the on-off application that does what
// we want.  On the plus side we don't need all of the complexity of the on-off
// application.  On the minus side, we don't have a helper, so we have to get
```

```
// a little more involved in the details, but this is trivial.
//
// So first, we create a socket and do the trace connect on it; then we pass
// this socket into the constructor of our simple application which we then
// install in the source node.
//
=============================================================================
//
class MyApp : public Application
{
public:

  MyApp ();
  virtual ~MyApp();

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

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);
    }
```

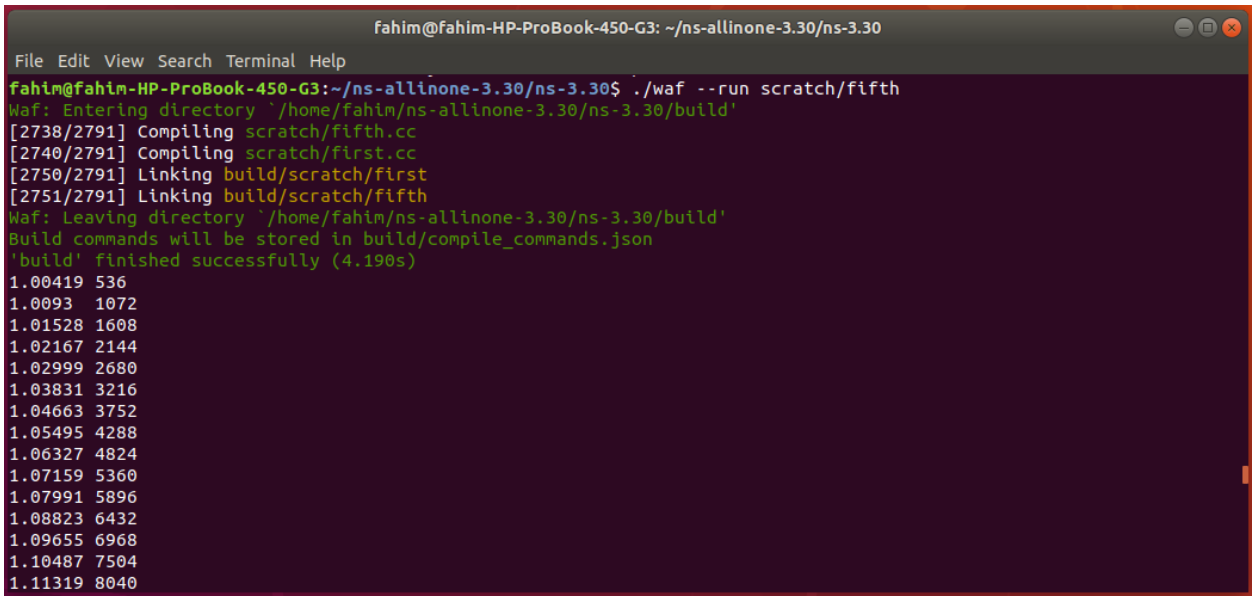
```
}

static void
CwndChange (uint32_t oldCwnd, uint32_t newCwnd)
{
  NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" << newCwnd);
}

static void
RxDrop (Ptr<const Packet> p)
{
  NS_LOG_UNCOND ("RxDrop at " << Simulator::Now ().GetSeconds ());
}

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

  NodeContainer nodes;
  nodes.Create (2);
```

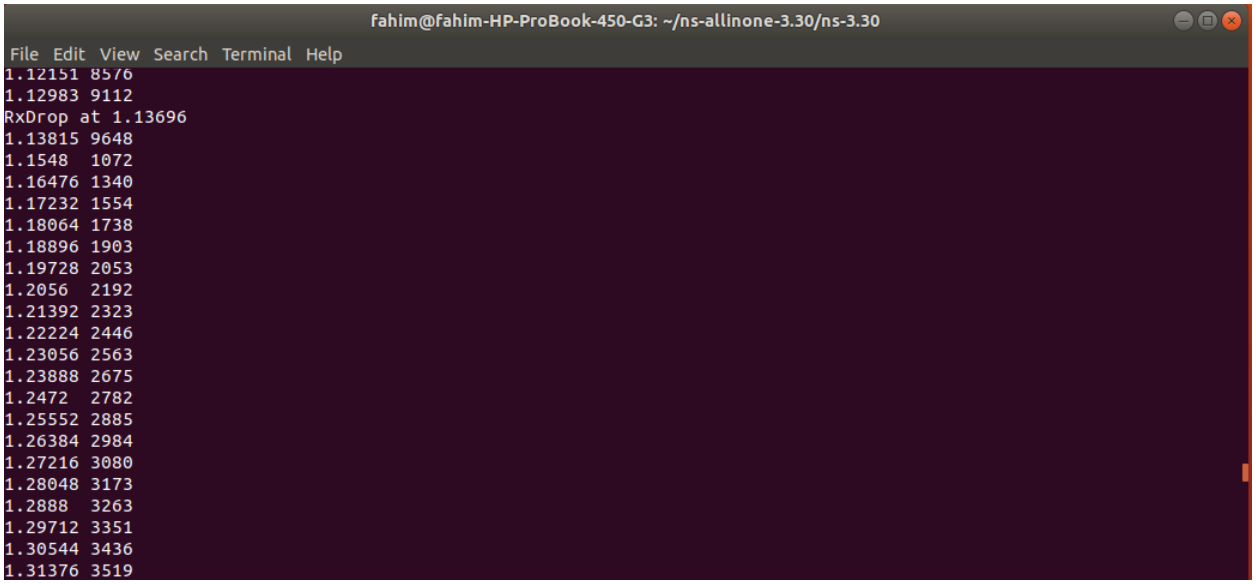
```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);

Ptr<RateErrorModel> em = CreateObject<RateErrorModel> ();
em->SetAttribute ("ErrorRate", DoubleValue (0.00001));
devices.Get (1)->SetAttribute ("ReceiveErrorModel", PointerValue (em));

InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.252");
Ipv4InterfaceContainer interfaces = address.Assign (devices);

uint16_t sinkPort = 8080;
Address sinkAddress (InetSocketAddress (interfaces.GetAddress (1), sinkPort));
```

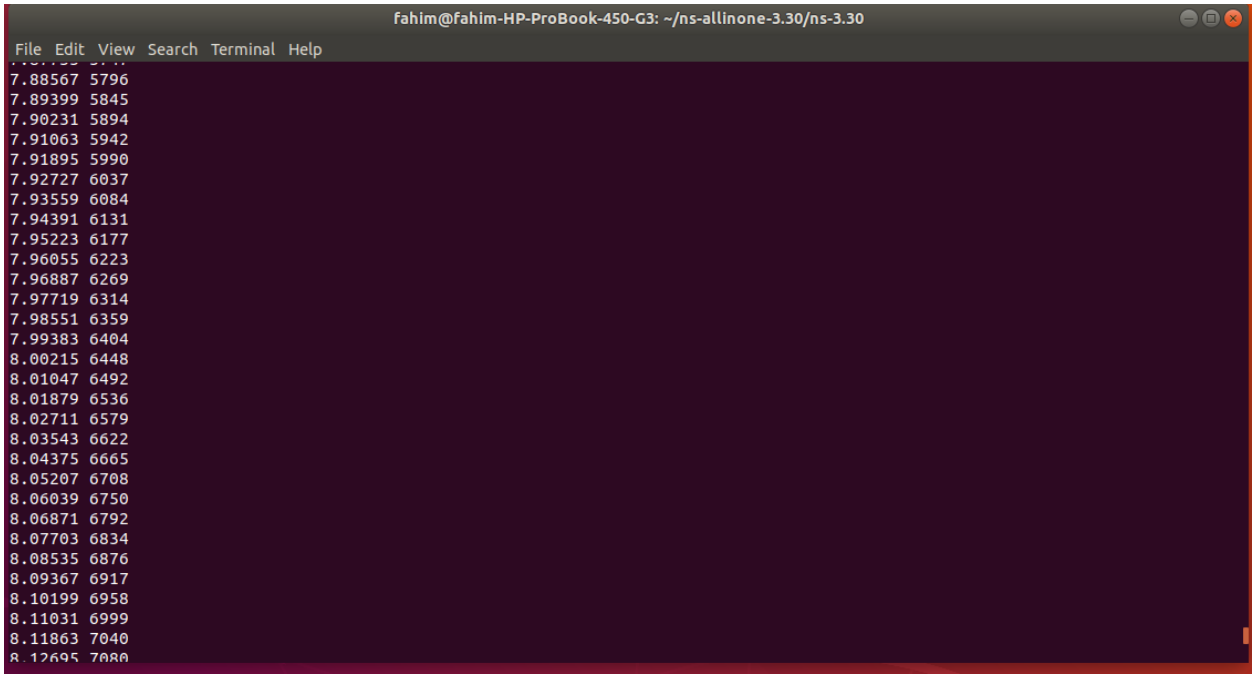
```
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), sinkPort));

  ApplicationContainer sinkApps = packetSinkHelper.Install (nodes.Get (1));

  sinkApps.Start (Seconds (0.));

  sinkApps.Stop (Seconds (20.));


  Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (nodes.Get (0),
TcpSocketFactory::GetTypeId ());

  ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow",
MakeCallback (&CwndChange));


  Ptr<MyApp> app = CreateObject<MyApp> ();

  app->Setup (ns3TcpSocket, sinkAddress, 1040, 1000, DataRate ("1Mbps"));

  nodes.Get (0)->AddApplication (app);

  app->SetStartTime (Seconds (1.));

  app->SetStopTime (Seconds (20.));


  devices.Get (1)->TraceConnectWithoutContext ("PhyRxDrop", MakeCallback
(&RxDrop));


  Simulator::Stop (Seconds (20));

  Simulator::Run ();

  Simulator::Destroy ();
```

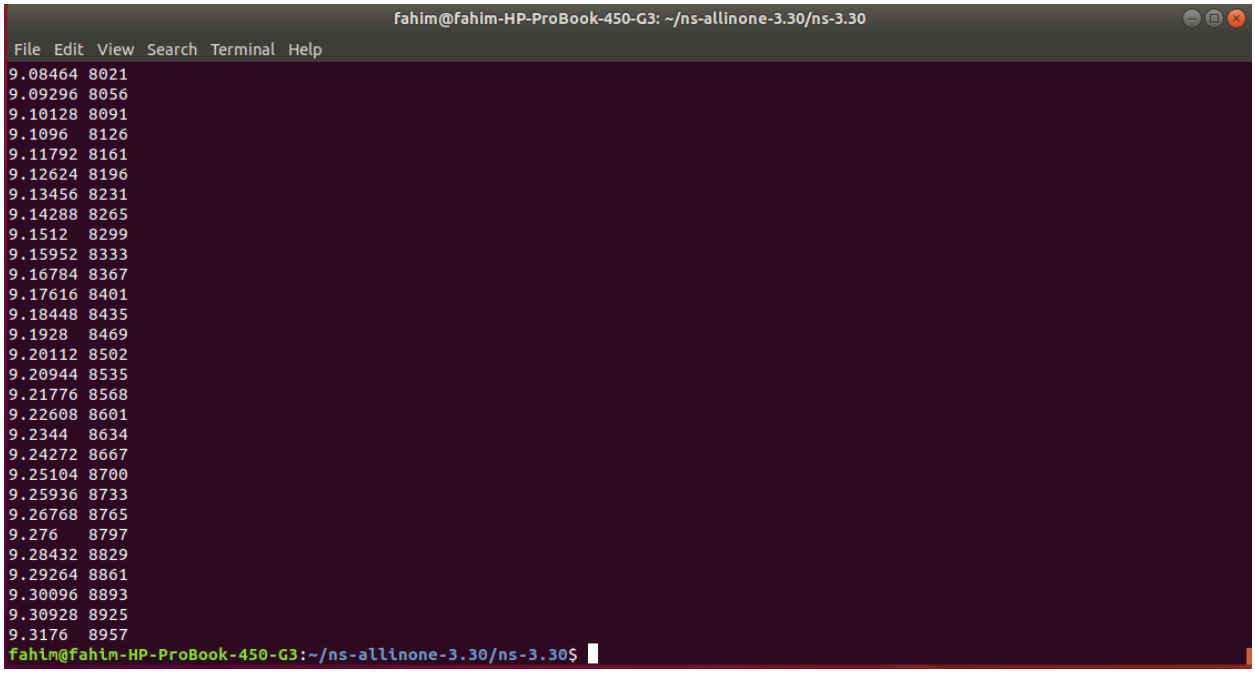
```
  return 0;
}
```

Output:






fahim@fahim-HP-ProBook-450-G3: ~/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help
7.88567 5796
7.89399 5845
7.90231 5894
7.91063 5942
7.91895 5990
7.92727 6037
7.93559 6084
7.94391 6131
7.95223 6177
7.96055 6223
7.96887 6269
7.97719 6314
7.98551 6359
7.99383 6404
8.00215 6448
8.01047 6492
8.01879 6536
8.02711 6579
8.03543 6622
8.04375 6665
8.05207 6708
8.06039 6750
8.06871 6792
8.07703 6834
8.08535 6876
8.09367 6917
8.10199 6958
8.11031 6999
8.11863 7040


fahim@fahim-HP-ProBook-450-G3: ~/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help
9.08464 8021
9.09296 8056
9.10128 8091
9.1096 8126
9.11792 8161
9.12624 8196
9.13456 8231
9.14288 8265
9.1512 8299
9.15952 8333
9.16784 8367
9.17616 8401
9.18448 8435
9.1928 8469
9.20112 8502
9.20944 8535
9.21776 8568
9.22608 8601
9.2344 8634
9.24272 8667
9.25104 8700
9.25936 8733
9.26768 8765
9.276 8797
9.28432 8829
9.29264 8861
9.30096 8893
9.30928 8925
9.3176 8957
fahim@fahim-HP-ProBook-450-G3:~/ns-allinone-3.30/ns-3.30$

Conclusion:

Comparison of TCP variants with respect to other remaining network parameters would be an important future attempt. Such type of analysis is very helpful for selecting the appropriate TCP in a certain platform. alternate congestion-control algorithms would provide little advantage over TCP congestion-control procedures.