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.
Course Code: ICT-4201

Submitted By	Submitted To
Name: **Md Khaled Hasan Manna** ID: **IT-16011** 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: 29-08-2020

Objective:

- Two client Node1 and Node2 on the left side of the dumbbell and server nodes Node3 and Node4 on the right side of the dumbbell. Let Node5 and Node6 form the bridge of the dumbbell. Use point to point links.
- Install a TCP socket instance on Node1 that will connect to Node3 and a UDP socket instance on Node2 that will connect to Node4.
- Start the TCP application at time 1s.UDP application at time 20s at rate Rate1 such that it clogs half the dumbbell bridge's link capacity.
- Increase the UDP application's rate at time 30s to rate Rate2 such that it clogs the whole of the dumbbell bridge's capacity.
- Use the ns-3 tracing mechanism to record changes in congestion window size of the TCP instance over time. Use gnuplot/matplotlib to visualise plots of cwnd vs time.
- Mark points of fast recovery and slow start in the graphs.
- Perform the above experiment for TCP variants Tahoe, Reno and New Reno, all of which are available with ns-3.

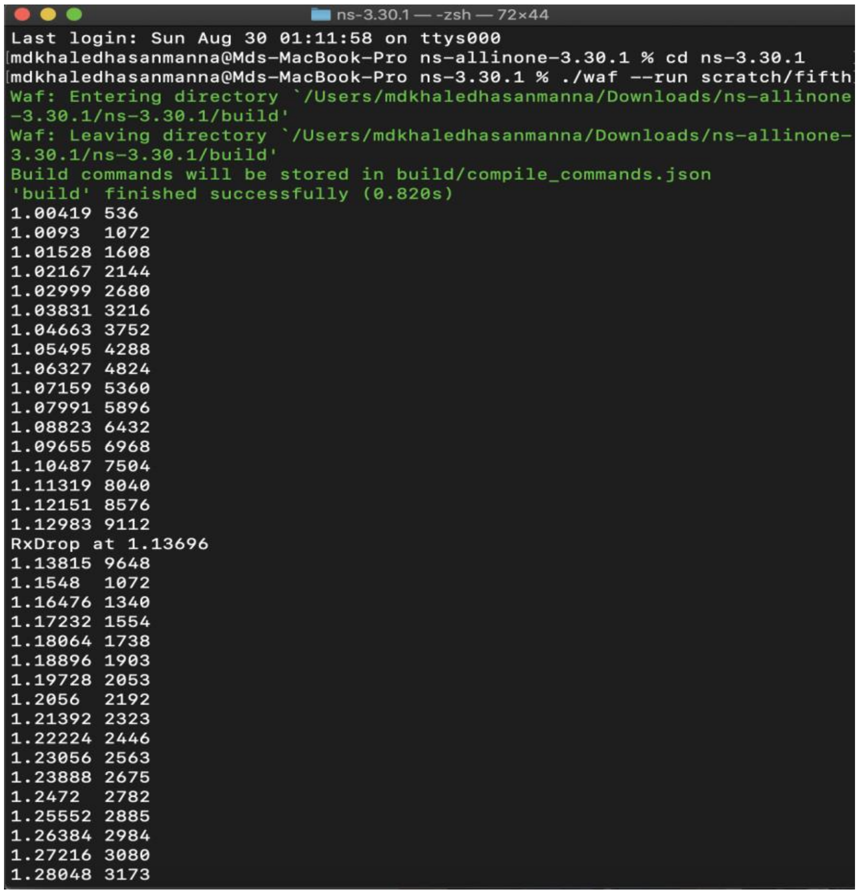
Source Code:

```
#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");
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;
```

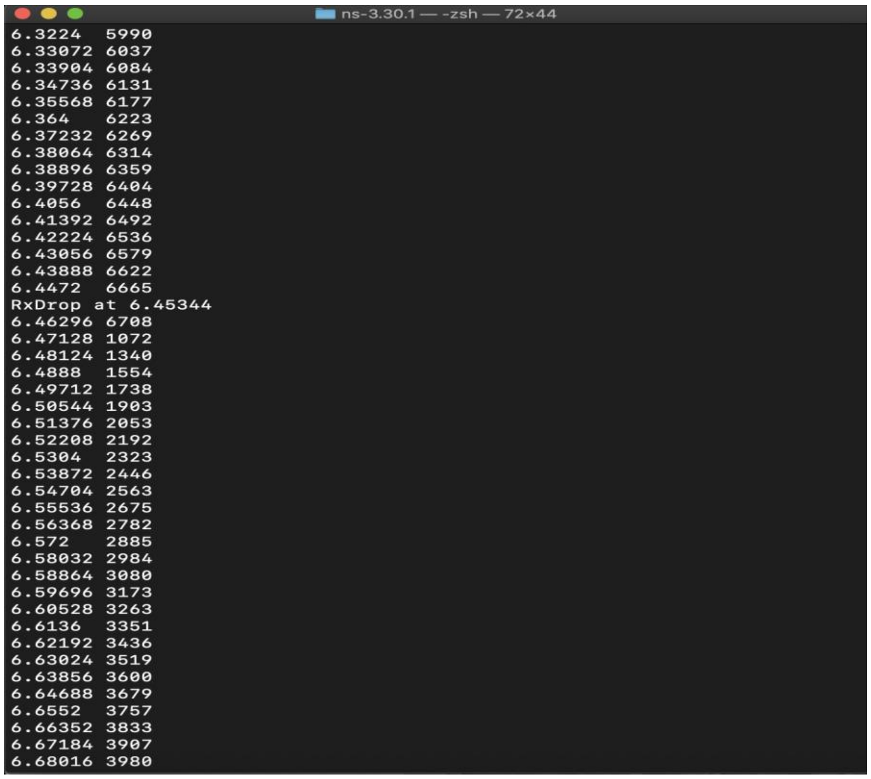
```
}
void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets, DataRate
dataRate)
{
  m_socket = socket;
  m_peer = address;
  m_packetSize = packetSize;
  m_nPackets = nPackets;
  m_dataRate = dataRate;
}
void
MyApp::StartApplication (void)
{
  m_running = true;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

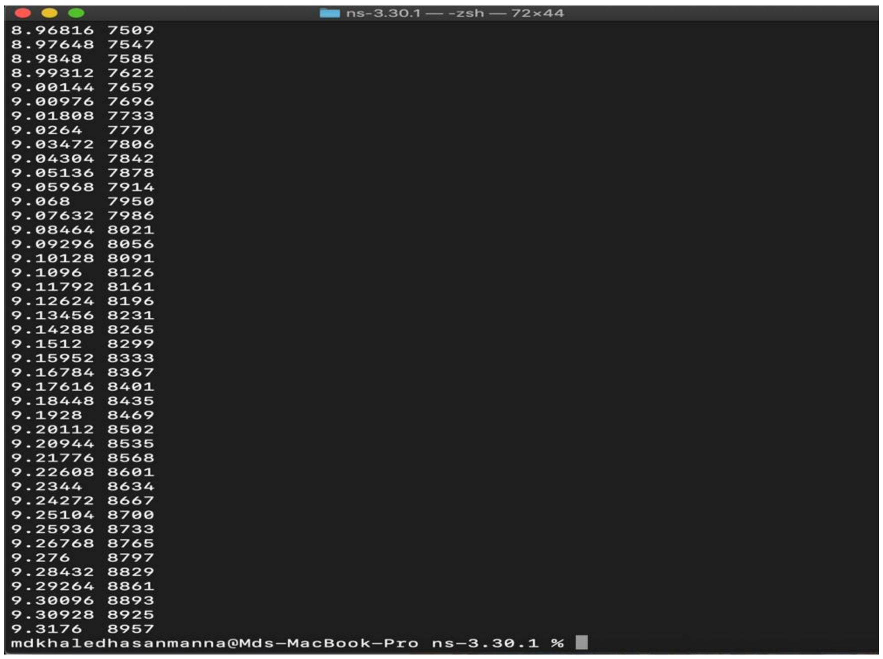
Output:



```
Last login: Sun Aug 30 01:11:58 on ttys000
mdkhaledhasanmanna@Mds-MacBook-Pro ns-allinone-3.30.1 % cd ns-3.30.1
mdkhaledhasanmanna@Mds-MacBook-Pro ns-3.30.1 % ./waf --run scratch/fifth
Waf: Entering directory `/Users/mdkhaledhasanmanna/Downloads/ns-allinone
-3.30.1/ns-3.30.1/build'
Waf: Leaving directory `/Users/mdkhaledhasanmanna/Downloads/ns-allinone-
3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.820s)
1.00419 536
1.0093  1072
1.01528 1608
1.02167 2144
1.02999 2680
1.03831 3216
1.04663 3752
1.05495 4288
1.06327 4824
1.07159 5360
1.07991 5896
1.08823 6432
1.09655 6968
1.10487 7504
1.11319 8040
1.12151 8576
1.12983 9112
RxDrop at 1.13696
1.13815 9648
1.1548  1072
1.16476 1340
1.17232 1554
1.18064 1738
1.18896 1903
1.19728 2053
1.2056  2192
1.21392 2323
1.22224 2446
1.23056 2563
1.23888 2675
1.2472  2782
1.25552 2885
1.26384 2984
1.27216 3080
1.28048 3173
```



```
6.3224  5990
6.33072 6037
6.33904 6084
6.34736 6131
6.35568 6177
6.364   6223
6.37232 6269
6.38064 6314
6.38896 6359
6.39728 6404
6.4056  6448
6.41392 6492
6.42224 6536
6.43056 6579
6.43888 6622
6.4472  6665
RxDrop at 6.45344
6.46296 6708
6.47128 1072
6.48124 1340
6.4888  1554
6.49712 1738
6.50544 1903
6.51376 2053
6.52208 2192
6.5304  2323
6.53872 2446
6.54704 2563
6.55536 2675
6.56368 2782
6.572   2885
6.58032 2984
6.58864 3080
6.59696 3173
6.60528 3263
6.6136  3351
6.62192 3436
6.63024 3519
6.63856 3600
6.64688 3679
6.6552  3757
6.66352 3833
6.67184 3907
6.68016 3980
```



```
8.96816 7509
8.97648 7547
8.9848  7585
8.99312 7622
9.00144 7659
9.00976 7696
9.01808 7733
9.0264  7770
9.03472 7806
9.04304 7842
9.05136 7878
9.05968 7914
9.068   7950
9.07632 7986
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
mdkhaledhasanmanna@Mds-MacBook-Pro ns-3.30.1 %
```

Conclusion:

TCP provides the reliable and connection oriented network.

Standard TCP congestion control is based on the reduction of its congestion window after a packet loss. We describe a variant of TCP (Tahoe, Vegas), TCP is most widely used transport protocol in both wired and wireless networks