LAB NO.:02

Name of Experiment: TCP Variants

Objectives:

1. To create a simple dumbbell topology, 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.
2. To install a TCP socket instance on Node1 that will connect to Node3.
3. To install a UDP socket instance on Node2 that will connect to Node4.
4. To start the TCP application at time 1s.
5. To start the UDP application at time 20s at rate Rate1 such that it clogs half the dumbbell bridge's link capacity.
6. To increase the UDP application's rate at time 30s to rate Rate2 such that it clogs the whole of the dumbbell bridge's capacity.
7. To use the ns-3 tracing mechanism to record changes in congestion window size of the TCP instance over time. Use gnuplot/matplotlib to visualize plots of cwnd vs time.
8. To mark points of fast recovery and slow start in the graphs.
9. To perform the above experiment for TCP variants Tahoe, Reno and New Reno, all of which are available with ns-3.

Source Code:

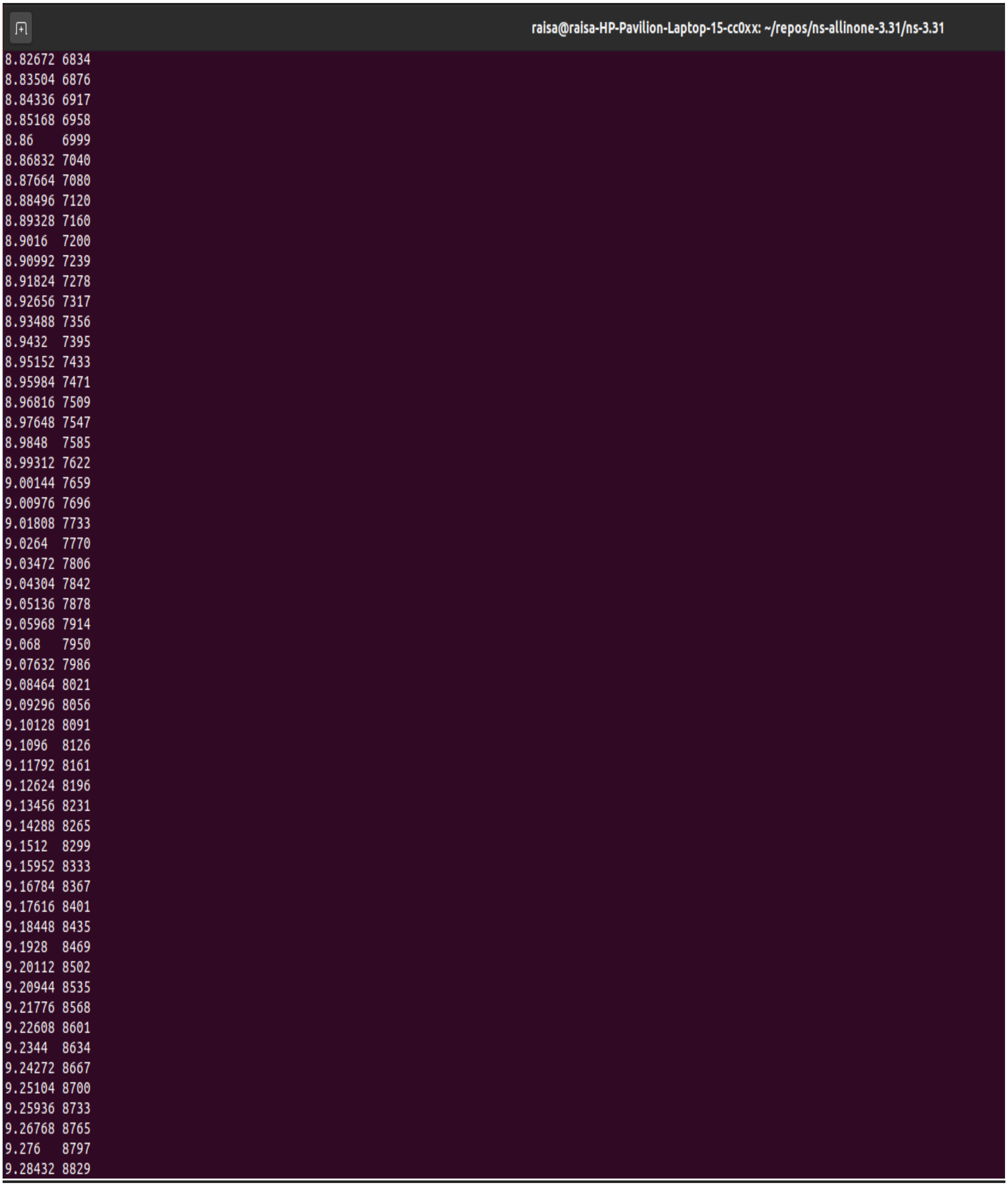
```
#include <fstream>

#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");
```

```
// ===========================================================================
//
//         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;
}
```

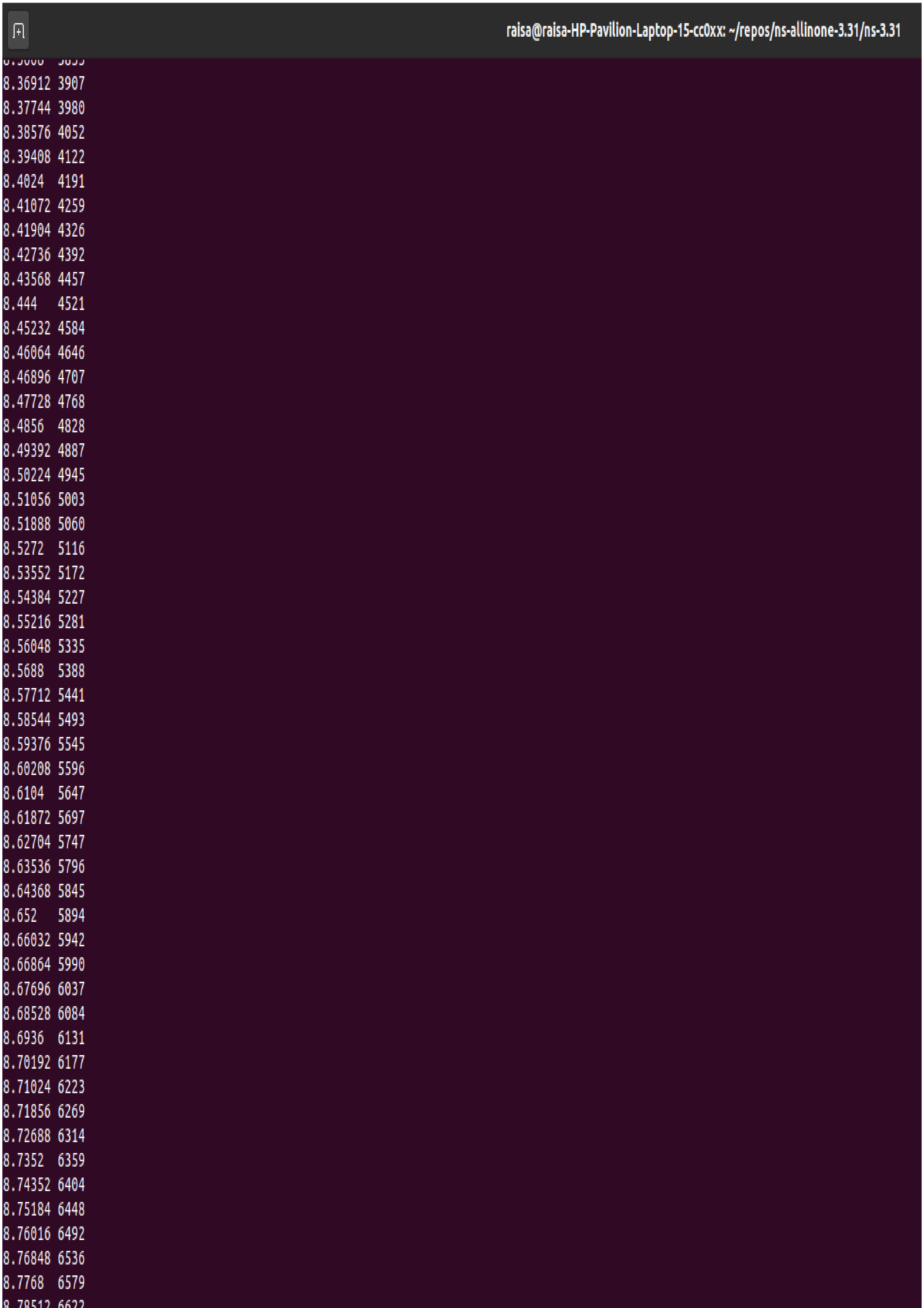
```
void
MyApp::StartApplication (void)
{
  m_running = true;
  m_packetsSent = 0;
  m_socket->Bind ();
  m_socket->Connect (m_peer);
  SendPacket ();
}

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:



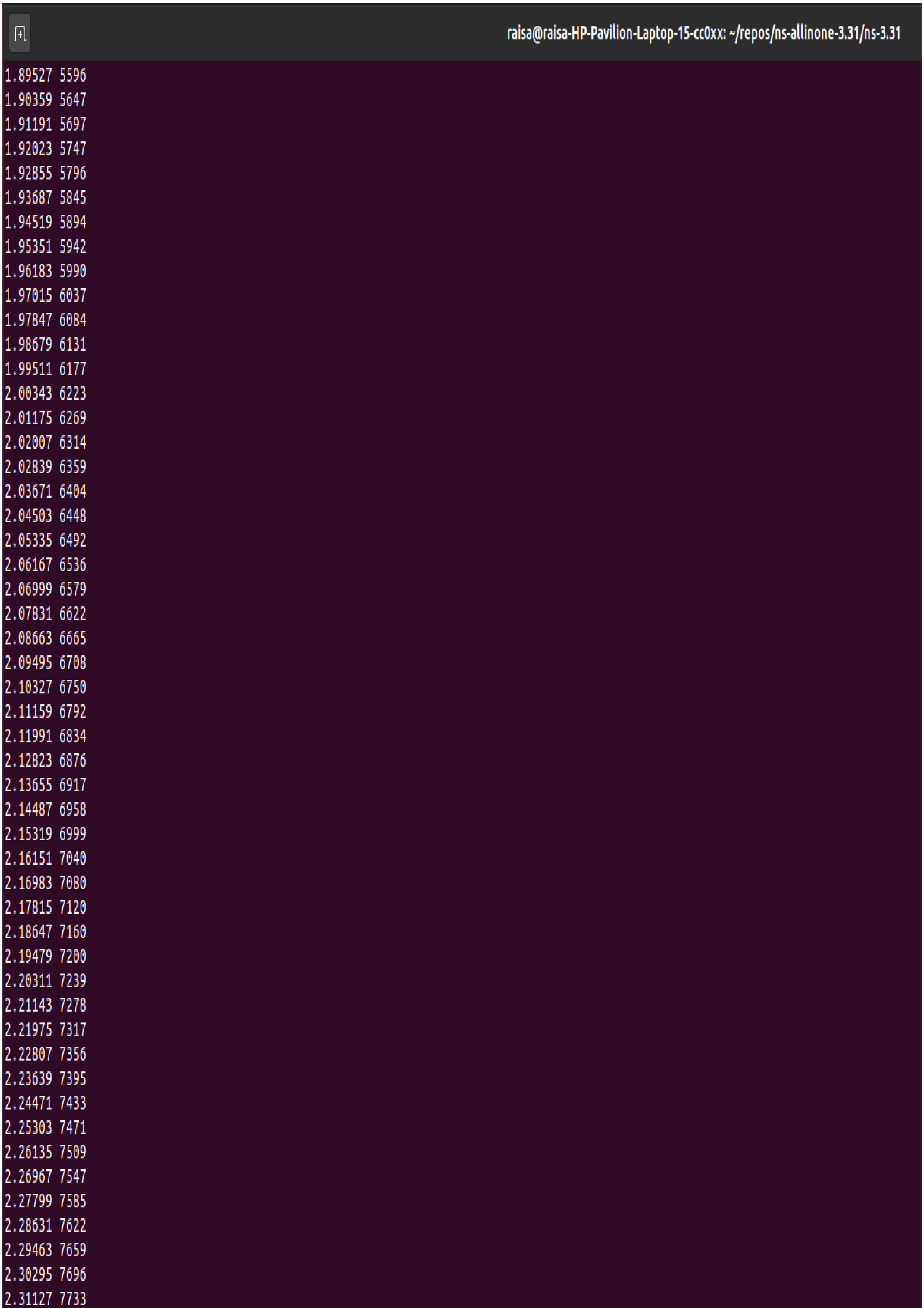
raisa@raisa-HP-Pavilion-Laptop-15-cc0xx: ~/repos/ns-allinone-3.31/ns-3.31

```
8.82672 6834
8.83504 6876
8.84336 6917
8.85168 6958
8.86    6999
8.86832 7040
8.87664 7080
8.88496 7120
8.89328 7160
8.9016  7200
8.90992 7239
8.91824 7278
8.92656 7317
8.93488 7356
8.9432  7395
8.95152 7433
8.95984 7471
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
```



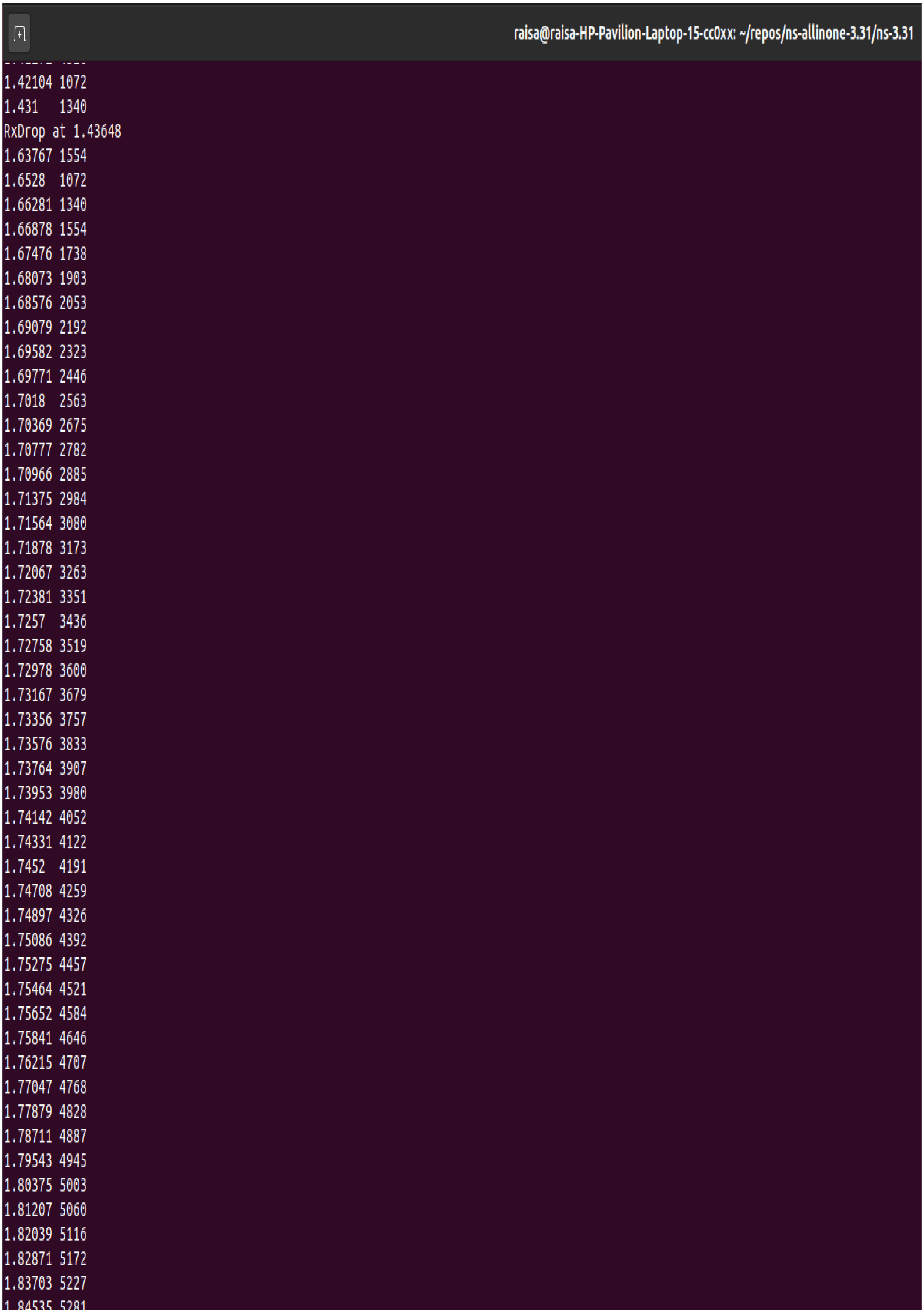
raisa@raisa-HP-Pavilion-Laptop-15-cc0xx: ~/repos/ns-allinone-3.31/ns-3.31

```
8.3608  3833
8.36912 3907
8.37744 3980
8.38576 4052
8.39408 4122
8.4024  4191
8.41072 4259
8.41904 4326
8.42736 4392
8.43568 4457
8.444   4521
8.45232 4584
8.46064 4646
8.46896 4707
8.47728 4768
8.4856  4828
8.49392 4887
8.50224 4945
8.51056 5003
8.51888 5060
8.5272  5116
8.53552 5172
8.54384 5227
8.55216 5281
8.56048 5335
8.5688  5388
8.57712 5441
8.58544 5493
8.59376 5545
8.60208 5596
8.6104  5647
8.61872 5697
8.62704 5747
8.63536 5796
8.64368 5845
8.652   5894
8.66032 5942
8.66864 5990
8.67696 6037
8.68528 6084
8.6936  6131
8.70192 6177
8.71024 6223
8.71856 6269
8.72688 6314
8.7352  6359
8.74352 6404
8.75184 6448
8.76016 6492
8.76848 6536
8.7768  6579
8.78512 6622
```

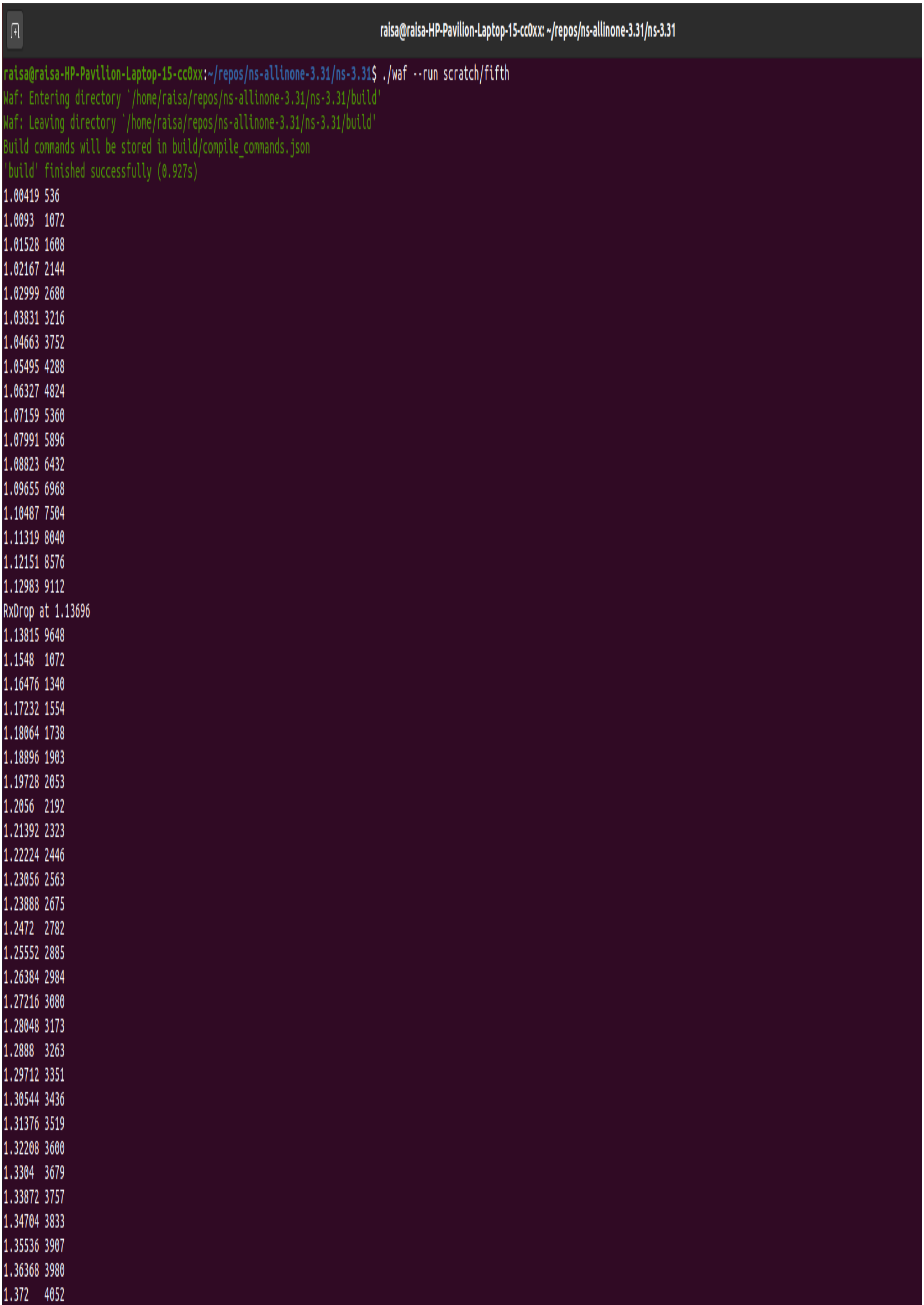


raisa@raisa-HP-Pavilion-Laptop-15-cc0xx: ~/repos/ns-allinone-3.31/ns-3.31

```
1.89527 5596
1.90359 5647
1.91191 5697
1.92023 5747
1.92855 5796
1.93687 5845
1.94519 5894
1.95351 5942
1.96183 5990
1.97015 6037
1.97847 6084
1.98679 6131
1.99511 6177
2.00343 6223
2.01175 6269
2.02007 6314
2.02839 6359
2.03671 6404
2.04503 6448
2.05335 6492
2.06167 6536
2.06999 6579
2.07831 6622
2.08663 6665
2.09495 6708
2.10327 6750
2.11159 6792
2.11991 6834
2.12823 6876
2.13655 6917
2.14487 6958
2.15319 6999
2.16151 7040
2.16983 7080
2.17815 7120
2.18647 7160
2.19479 7200
2.20311 7239
2.21143 7278
2.21975 7317
2.22807 7356
2.23639 7395
2.24471 7433
2.25303 7471
2.26135 7509
2.26967 7547
2.27799 7585
2.28631 7622
2.29463 7659
2.30295 7696
2.31127 7733
```

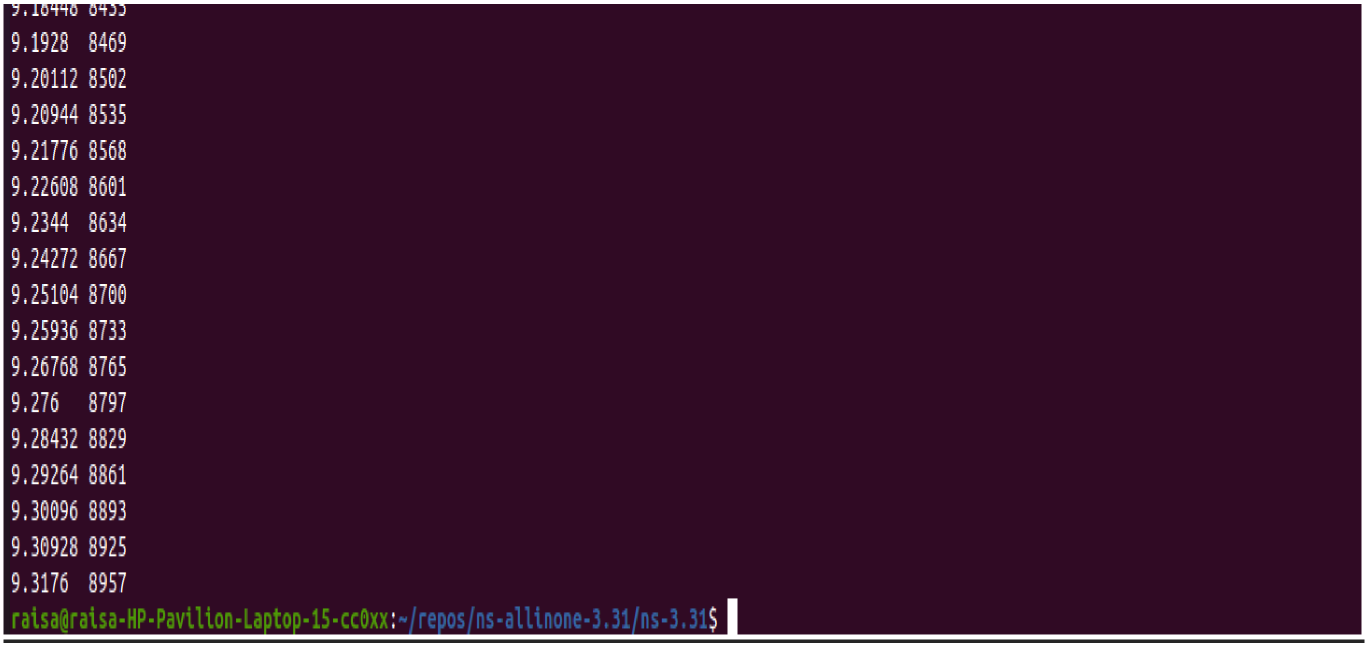

raisa@raisa-HP-Pavilion-Laptop-15-cc0xx: ~/repos/ns-allinone-3.31/ns-3.31

```
1.42104 1072
1.431   1340
RxDrop at 1.43648
1.63767 1554
1.6528  1072
1.66281 1340
1.66878 1554
1.67476 1738
1.68073 1903
1.68576 2053
1.69079 2192
1.69582 2323
1.69771 2446
1.7018  2563
1.70369 2675
1.70777 2782
1.70966 2885
1.71375 2984
1.71564 3080
1.71878 3173
1.72067 3263
1.72381 3351
1.7257  3436
1.72758 3519
1.72978 3600
1.73167 3679
1.73356 3757
1.73576 3833
1.73764 3907
1.73953 3980
1.74142 4052
1.74331 4122
1.7452  4191
1.74708 4259
1.74897 4326
1.75086 4392
1.75275 4457
1.75464 4521
1.75652 4584
1.75841 4646
1.76215 4707
1.77047 4768
1.77879 4828
1.78711 4887
1.79543 4945
1.80375 5003
1.81207 5060
1.82039 5116
1.82871 5172
1.83703 5227
1.84535 5281
```



raisa@raisa-HP-Pavilion-Laptop-15-cc0xx: ~/repos/ns-allinone-3.31/ns-3.31

```
raisa@raisa-HP-Pavilion-Laptop-15-cc0xx:~/repos/ns-allinone-3.31/ns-3.31$ ./waf --run scratch/fifth
Waf: Entering directory `/home/raisa/repos/ns-allinone-3.31/ns-3.31/build'
Waf: Leaving directory `/home/raisa/repos/ns-allinone-3.31/ns-3.31/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.927s)
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
1.2888  3263
1.29712 3351
1.30544 3436
1.31376 3519
1.32208 3600
1.3304  3679
1.33872 3757
1.34704 3833
1.35536 3907
1.36368 3980
1.372   4052
```



```
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
raisa@raisa-HP-Pavilion-Laptop-15-cc0xx:~/repos/ns-allinone-3.31/ns-3.31$
```

Conclusion: Here in this experiment, TCP internals and the difference between each of the variants are found using NS-3 mechanism and it was successfully done.