Name: Wahia Tasnim **ID: IT-16029**

Experiment N0: 02

Name of Experiment : TCP Variants

Objective :

1. 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. Install a TCP socket instance on Node1 that will connect to Node3.
3. Install a UDP socket instance on Node2 that will connect to Node4.
4. Start the TCP application at time 1s.
5. Start the UDP application at time 20s at rate Rate1 such that it clogs half the dumbbell bridge's link capacity.
6. 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. 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.
8. Mark points of fast recovery and slow start in the graphs.
9. Perform the above experiment for TCP variants Tahoe, Reno and New Reno, all of which are available with ns-3.

Source Code:

```
 * Network topology:
*
*   Ap    STA
*   *      *
*   |      |
*   n1     n2
*
* In this example, an HT station sends TCP packets to the access point.
```

```
 * We report the total throughput received during a window of 100ms.
 * The user can specify the application data rate and choose the variant
 * of TCP i.e. congestion control algorithm to use.
 */

#include "ns3/command-line.h"
#include "ns3/config.h"
#include "ns3/string.h"
#include "ns3/log.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/mobility-helper.h"
#include "ns3/on-off-helper.h"
#include "ns3/yans-wifi-channel.h"
#include "ns3/mobility-model.h"
#include "ns3/packet-sink.h"
#include "ns3/packet-sink-helper.h"
#include "ns3/tcp-westwood.h"
#include "ns3/internet-stack-helper.h"
#include "ns3/ipv4-address-helper.h"
#include "ns3/ipv4-global-routing-helper.h"

NS_LOG_COMPONENT_DEFINE ("wifi-tcp");
```

```
using namespace ns3;


Ptr<PacketSink> sink;                     /* Pointer to the packet sink application */

uint64_t lastTotalRx = 0;                 /* The value of the last total received bytes */


void

CalculateThroughput ()

{

  Time now = Simulator::Now ();                                  /* Return the simulator's virtual time.
*/

  double cur = (sink->GetTotalRx () - lastTotalRx) * (double) 8 / 1e5;     /* Convert Application
RX Packets to MBits. */

  std::cout << now.GetSeconds () << "s: \t" << cur << " Mbit/s" << std::endl;

  lastTotalRx = sink->GetTotalRx ();

  Simulator::Schedule (MilliSeconds (100), &CalculateThroughput);

}


int

main (int argc, char *argv[])

{

  uint32_t payloadSize = 1472;                      /* Transport layer payload size in bytes. */

  std::string dataRate = "100Mbps";                  /* Application layer datarate. */

  std::string tcpVariant = "TcpNewReno";             /* TCP variant type. */
  std::string phyRate = "HtMcs7";                    /* Physical layer bitrate. */

  double simulationTime = 10;                        /* Simulation time in seconds. */

  bool pcapTracing = false;                          /* PCAP Tracing is enabled or not. */



  /* Command line argument parser setup. */

  CommandLine cmd;

  cmd.AddValue ("payloadSize", "Payload size in bytes", payloadSize);

  cmd.AddValue ("dataRate", "Application data ate", dataRate);

  cmd.AddValue ("tcpVariant", "Transport protocol to use: TcpNewReno, "

                "TcpHybla, TcpHighSpeed, TcpHtcp, TcpVegas, TcpScalable, TcpVeno, "

                "TcpBic, TcpYeah, TcpIllinois, TcpWestwood, TcpWestwoodPlus, TcpLedbat ",
tcpVariant);

  cmd.AddValue ("phyRate", "Physical layer bitrate", phyRate);

  cmd.AddValue ("simulationTime", "Simulation time in seconds", simulationTime);

  cmd.AddValue ("pcap", "Enable/disable PCAP Tracing", pcapTracing);

  cmd.Parse (argc, argv);



  tcpVariant = std::string ("ns3::") + tcpVariant;

  // Select TCP variant

  if (tcpVariant.compare ("ns3::TcpWestwoodPlus") == 0)

    {

      // TcpWestwoodPlus is not an actual TypeId name; we need TcpWestwood here

      Config::SetDefault ("ns3::TcpL4Protocol::SocketType", TypeIdValue
(TcpWestwood::GetTypeId ()));
```

```
      // the default protocol type in ns3::TcpWestwood is WESTWOOD

      Config::SetDefault ("ns3::TcpWestwood::ProtocolType", EnumValue
(TcpWestwood::WESTWOODPLUS));

    }

  else

    {

      TypeId tcpTid;

      NS_ABORT_MSG_UNLESS (TypeId::LookupByNameFailSafe (tcpVariant, &tcpTid),
"TypeId " << tcpVariant << " not found");

      Config::SetDefault ("ns3::TcpL4Protocol::SocketType", TypeIdValue
(TypeId::LookupByName (tcpVariant)));

    }



  /* Configure TCP Options */

  Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (payloadSize));



  WifiMacHelper wifiMac;

  WifiHelper wifiHelper;

  wifiHelper.SetStandard (WIFI_PHY_STANDARD_80211n_5GHZ);



  /* Set up Legacy Channel */

  YansWifiChannelHelper wifiChannel;

  wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");

  wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel", "Frequency",
DoubleValue (5e9));
```

```
/* Setup Physical Layer */

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

wifiPhy.SetChannel (wifiChannel.Create ());

wifiPhy.SetErrorRateModel ("ns3::YansErrorRateModel");

wifiHelper.SetRemoteStationManager ("ns3::ConstantRateWifiManager",

                                    "DataMode", StringValue (phyRate),

                                    "ControlMode", StringValue ("HtMcs0"));


NodeContainer networkNodes;

networkNodes.Create (2);

Ptr<Node> apWifiNode = networkNodes.Get (0);

Ptr<Node> staWifiNode = networkNodes.Get (1);


/* Configure AP */

Ssid ssid = Ssid ("network");

wifiMac.SetType ("ns3::ApWifiMac",

                 "Ssid", SsidValue (ssid));


NetDeviceContainer apDevice;

apDevice = wifiHelper.Install (wifiPhy, wifiMac, apWifiNode);


/* Configure STA */
```

```
wifiMac.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid));

NetDeviceContainer staDevices;
staDevices = wifiHelper.Install (wifiPhy, wifiMac, staWifiNode);

/* Mobility model */
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (1.0, 1.0, 0.0));

mobility.SetPositionAllocator (positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (apWifiNode);
mobility.Install (staWifiNode);

/* Internet stack */
InternetStackHelper stack;
stack.Install (networkNodes);

Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.255.255.0");
```

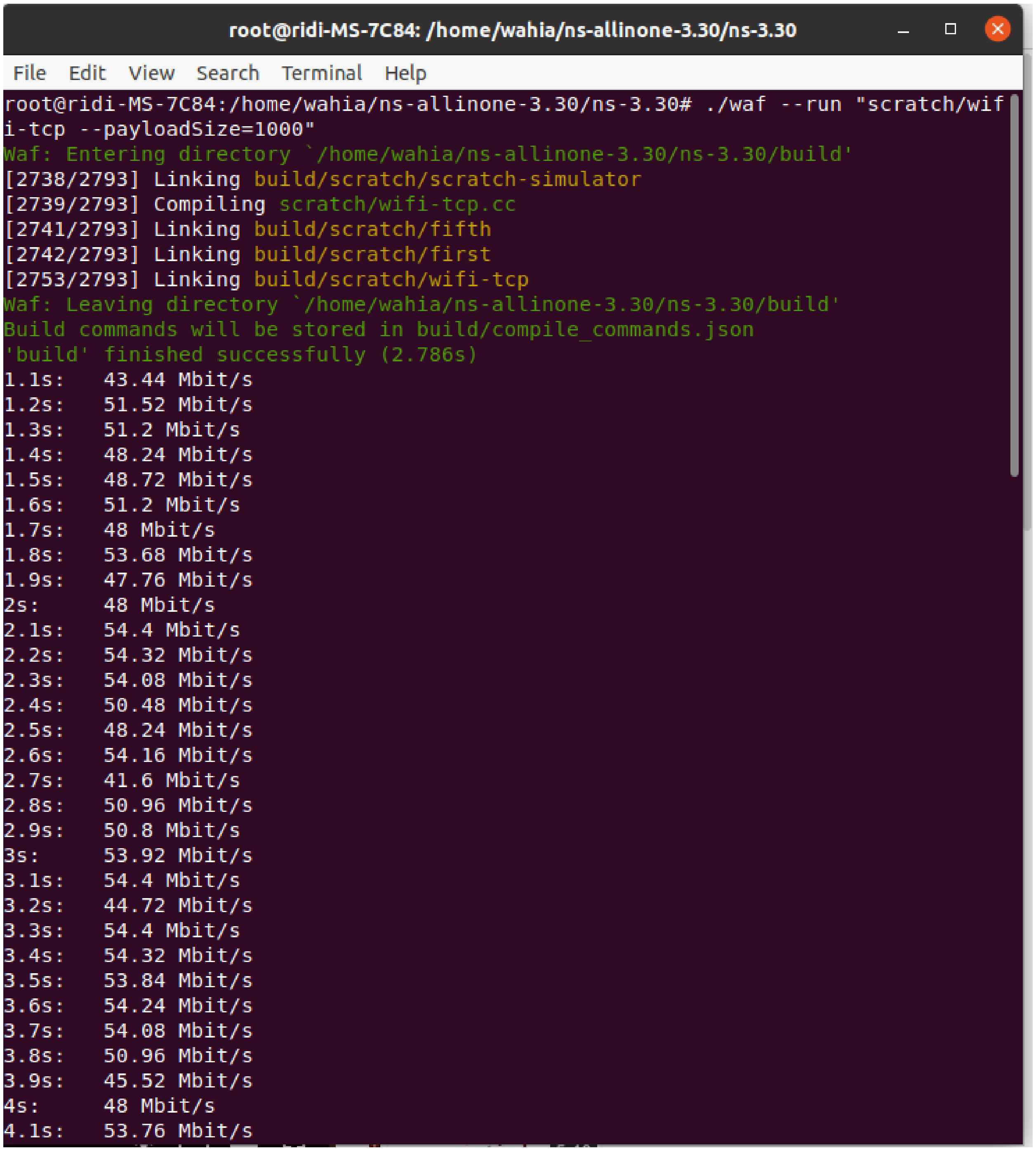
```
  Ipv4InterfaceContainer apInterface;

  apInterface = address.Assign (apDevice);

  Ipv4InterfaceContainer staInterface;

  staInterface = address.Assign (staDevices);



  /* Populate routing table */

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();



  /* Install TCP Receiver on the access point */

  PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), 9));

  ApplicationContainer sinkApp = sinkHelper.Install (apWifiNode);

  sink = StaticCast<PacketSink> (sinkApp.Get (0));



  /* Install TCP/UDP Transmitter on the station */

  OnOffHelper server ("ns3::TcpSocketFactory", (InetSocketAddress (apInterface.GetAddress
(0), 9)));

  server.SetAttribute ("PacketSize", UintegerValue (payloadSize));

  server.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));

  server.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));

  server.SetAttribute ("DataRate", DataRateValue (DataRate (dataRate)));

  ApplicationContainer serverApp = server.Install (staWifiNode);



  /* Start Applications */
```

```
sinkApp.Start (Seconds (0.0));

serverApp.Start (Seconds (1.0));

Simulator::Schedule (Seconds (1.1), &CalculateThroughput);


/* Enable Traces */

if (pcapTracing)

  {

    wifiPhy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);

    wifiPhy.EnablePcap ("AccessPoint", apDevice);

    wifiPhy.EnablePcap ("Station", staDevices);

  }


/* Start Simulation */

Simulator::Stop (Seconds (simulationTime + 1));

Simulator::Run ();


double averageThroughput = ((sink->GetTotalRx () * 8) / (1e6 * simulationTime));


Simulator::Destroy ();


if (averageThroughput < 50)

  {

    NS_LOG_ERROR ("Obtained throughput is not in the expected boundaries!");
```

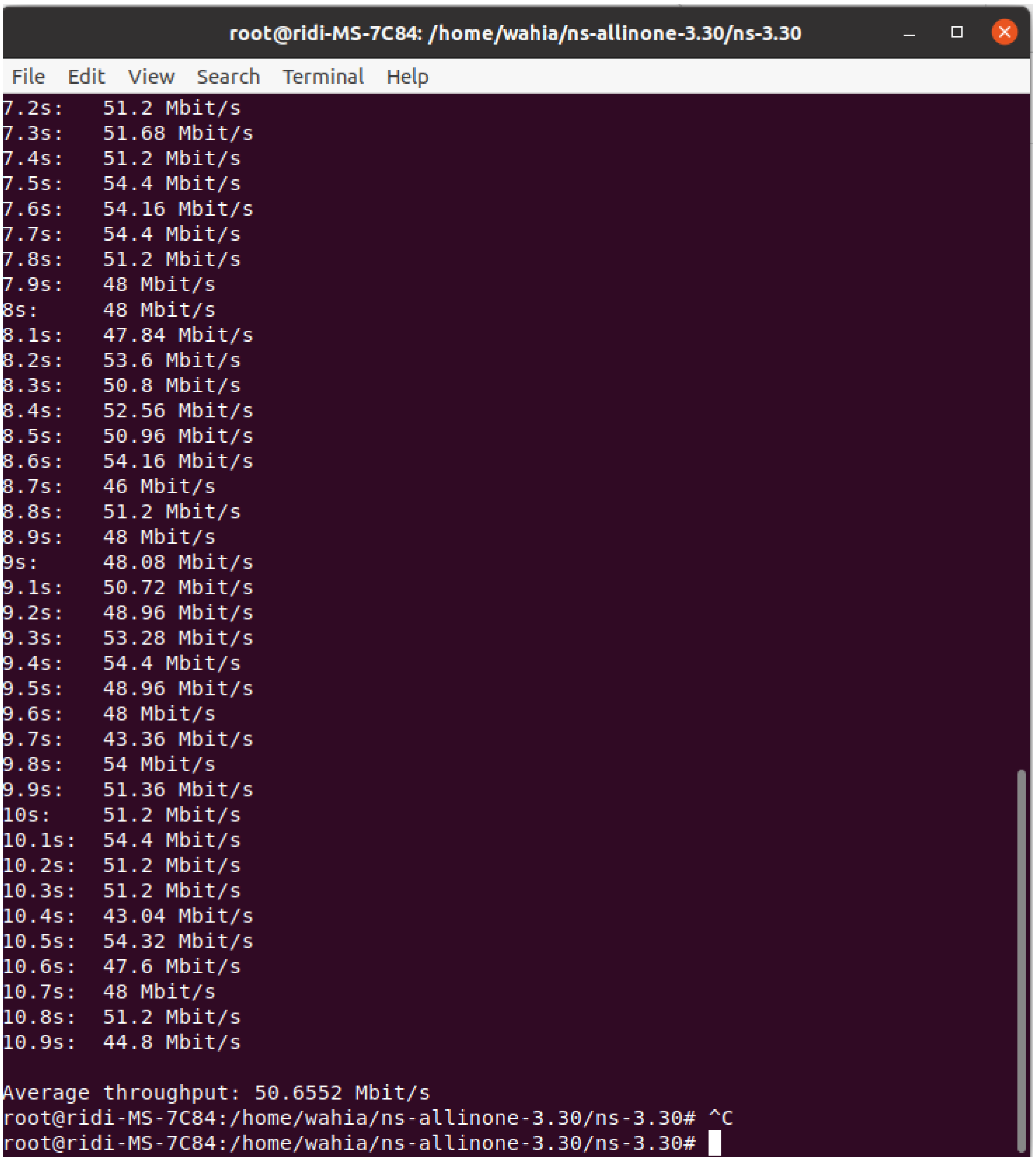
```
      exit (1);

    }

  std::cout << "\nAverage throughput: " << averageThroughput << " Mbit/s" << std::endl;

  return 0;

}
```

Output:

```
WiFi-TCP
payloadSize=1000
Average throughput: 50.6552 Mbit/s
```

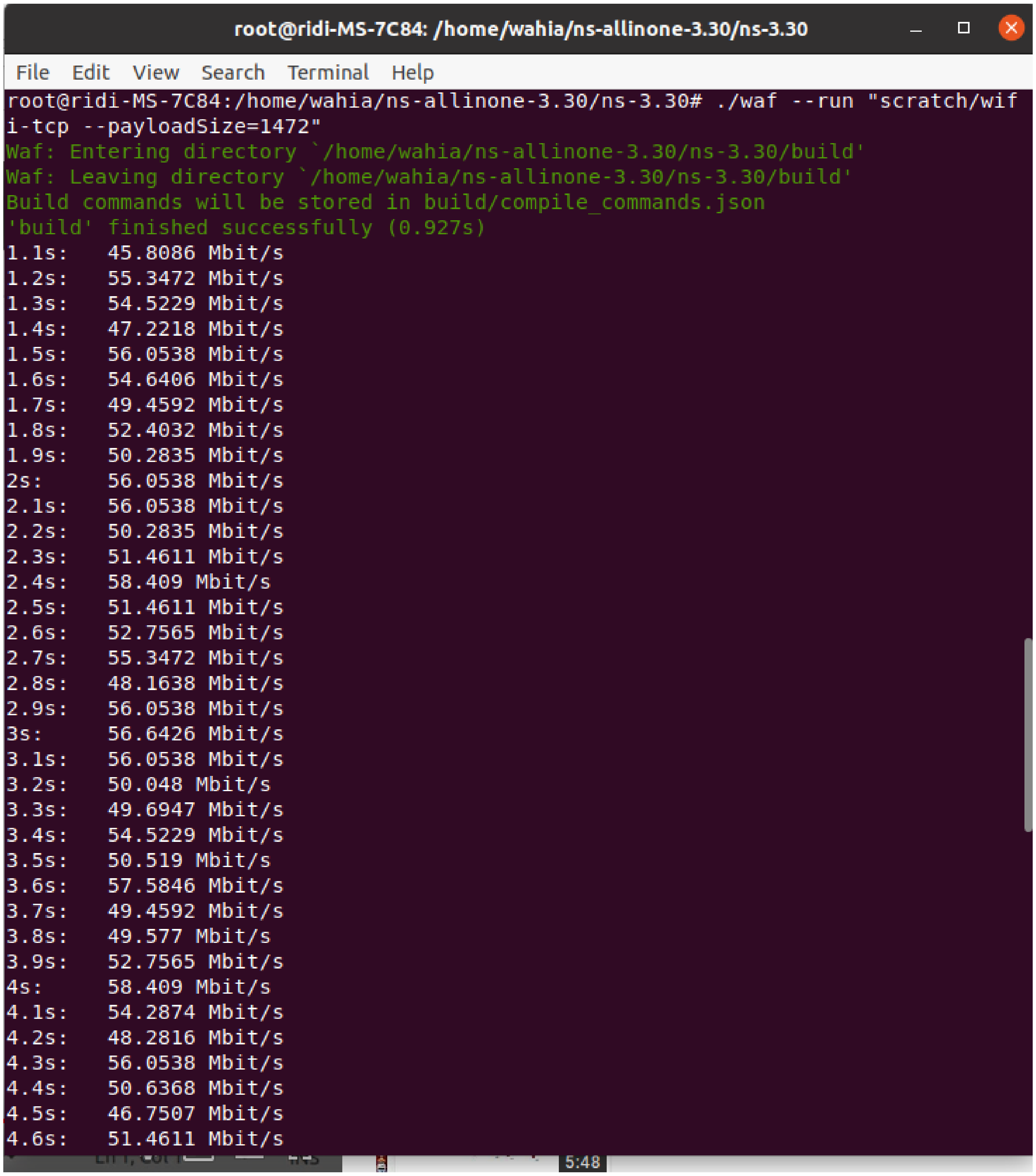

root@ridi-MS-7C84: /home/wahia/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help

```
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30# ./waf --run "scratch/wif
i-tcp --payloadSize=1000"
Waf: Entering directory `/home/wahia/ns-allinone-3.30/ns-3.30/build'
[2738/2793] Linking build/scratch/scratch-simulator
[2739/2793] Compiling scratch/wifi-tcp.cc
[2741/2793] Linking build/scratch/fifth
[2742/2793] Linking build/scratch/first
[2753/2793] Linking build/scratch/wifi-tcp
Waf: Leaving directory `/home/wahia/ns-allinone-3.30/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.786s)
1.1s:   43.44 Mbit/s
1.2s:   51.52 Mbit/s
1.3s:   51.2 Mbit/s
1.4s:   48.24 Mbit/s
1.5s:   48.72 Mbit/s
1.6s:   51.2 Mbit/s
1.7s:   48 Mbit/s
1.8s:   53.68 Mbit/s
1.9s:   47.76 Mbit/s
2s:     48 Mbit/s
2.1s:   54.4 Mbit/s
2.2s:   54.32 Mbit/s
2.3s:   54.08 Mbit/s
2.4s:   50.48 Mbit/s
2.5s:   48.24 Mbit/s
2.6s:   54.16 Mbit/s
2.7s:   41.6 Mbit/s
2.8s:   50.96 Mbit/s
2.9s:   50.8 Mbit/s
3s:     53.92 Mbit/s
3.1s:   54.4 Mbit/s
3.2s:   44.72 Mbit/s
3.3s:   54.4 Mbit/s
3.4s:   54.32 Mbit/s
3.5s:   53.84 Mbit/s
3.6s:   54.24 Mbit/s
3.7s:   54.08 Mbit/s
3.8s:   50.96 Mbit/s
3.9s:   45.52 Mbit/s
4s:     48 Mbit/s
4.1s:   53.76 Mbit/s
```

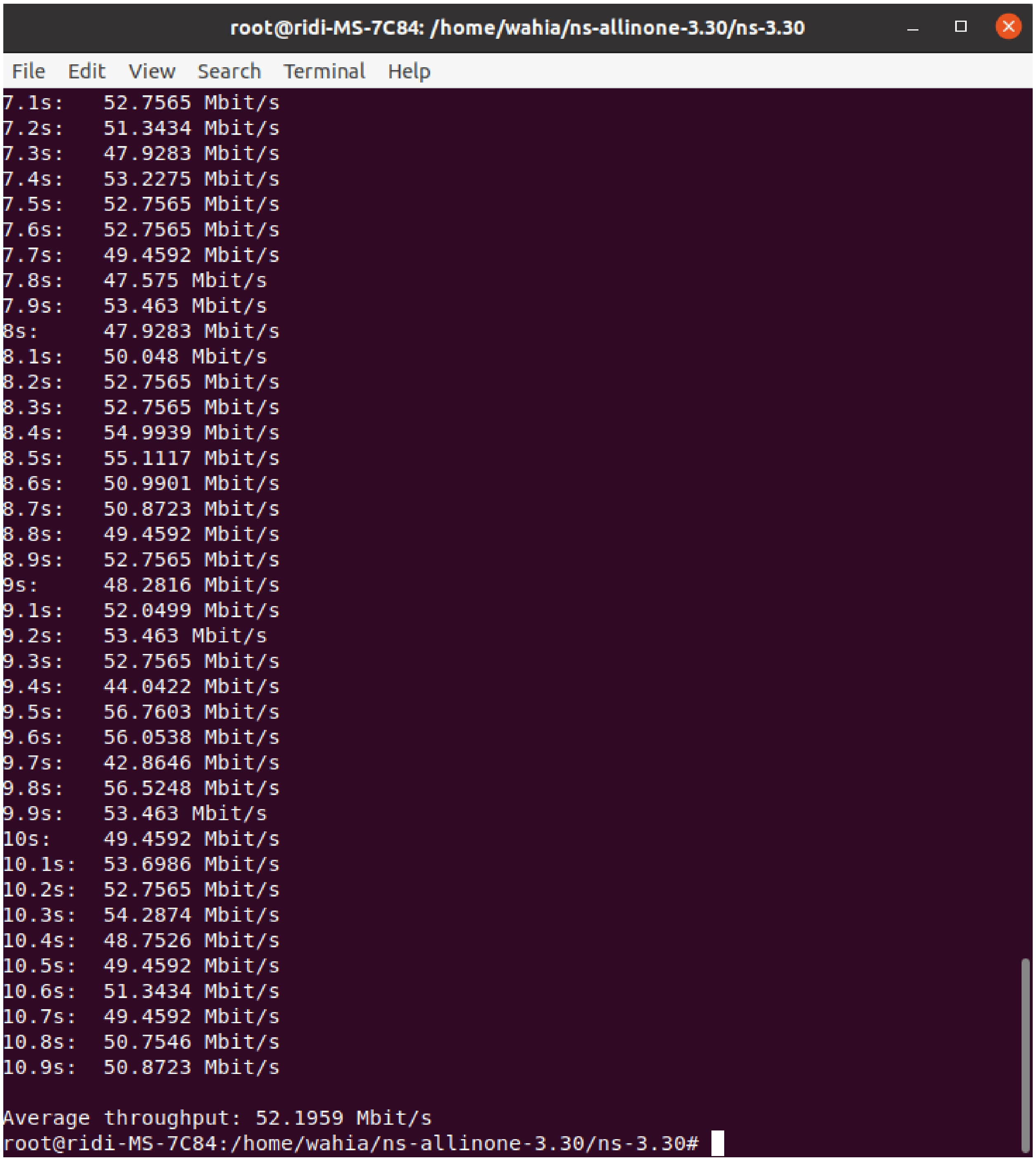

root@ridi-MS-7C84: /home/wahia/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help

```
7.2s:   51.2 Mbit/s
7.3s:   51.68 Mbit/s
7.4s:   51.2 Mbit/s
7.5s:   54.4 Mbit/s
7.6s:   54.16 Mbit/s
7.7s:   54.4 Mbit/s
7.8s:   51.2 Mbit/s
7.9s:   48 Mbit/s
8s:     48 Mbit/s
8.1s:   47.84 Mbit/s
8.2s:   53.6 Mbit/s
8.3s:   50.8 Mbit/s
8.4s:   52.56 Mbit/s
8.5s:   50.96 Mbit/s
8.6s:   54.16 Mbit/s
8.7s:   46 Mbit/s
8.8s:   51.2 Mbit/s
8.9s:   48 Mbit/s
9s:     48.08 Mbit/s
9.1s:   50.72 Mbit/s
9.2s:   48.96 Mbit/s
9.3s:   53.28 Mbit/s
9.4s:   54.4 Mbit/s
9.5s:   48.96 Mbit/s
9.6s:   48 Mbit/s
9.7s:   43.36 Mbit/s
9.8s:   54 Mbit/s
9.9s:   51.36 Mbit/s
10s:    51.2 Mbit/s
10.1s:  54.4 Mbit/s
10.2s:  51.2 Mbit/s
10.3s:  51.2 Mbit/s
10.4s:  43.04 Mbit/s
10.5s:  54.32 Mbit/s
10.6s:  47.6 Mbit/s
10.7s:  48 Mbit/s
10.8s:  51.2 Mbit/s
10.9s:  44.8 Mbit/s

Average throughput: 50.6552 Mbit/s
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30# ^C
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30# 
```

payloadSize=1472
Average throughput: 52.1959 Mbit/s

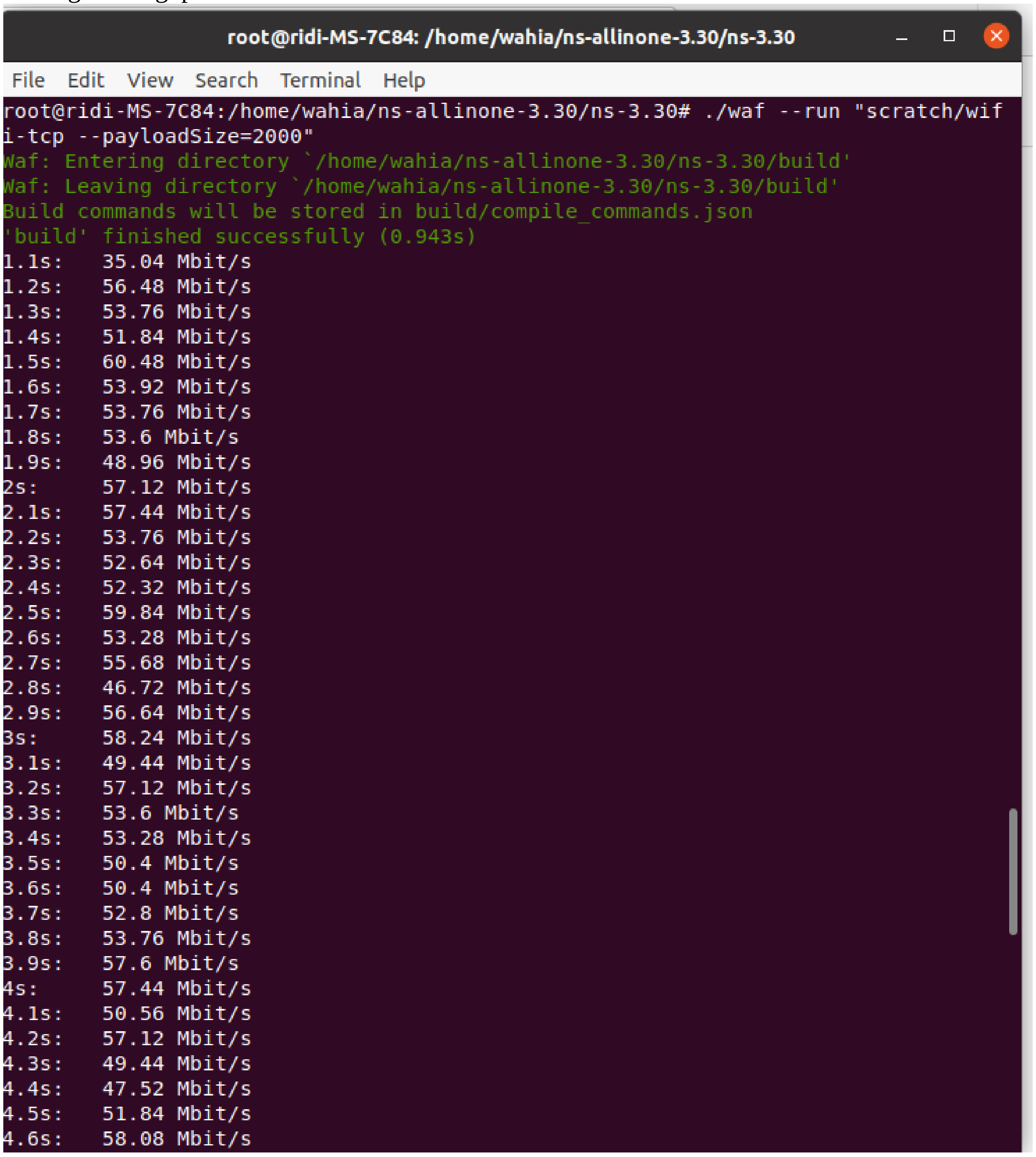

root@ridi-MS-7C84: /home/wahia/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help

```
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30# ./waf --run "scratch/wif
i-tcp --payloadSize=1472"
Waf: Entering directory `/home/wahia/ns-allinone-3.30/ns-3.30/build'
Waf: Leaving directory `/home/wahia/ns-allinone-3.30/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.927s)
1.1s:   45.8086 Mbit/s
1.2s:   55.3472 Mbit/s
1.3s:   54.5229 Mbit/s
1.4s:   47.2218 Mbit/s
1.5s:   56.0538 Mbit/s
1.6s:   54.6406 Mbit/s
1.7s:   49.4592 Mbit/s
1.8s:   52.4032 Mbit/s
1.9s:   50.2835 Mbit/s
2s:     56.0538 Mbit/s
2.1s:   56.0538 Mbit/s
2.2s:   50.2835 Mbit/s
2.3s:   51.4611 Mbit/s
2.4s:   58.409 Mbit/s
2.5s:   51.4611 Mbit/s
2.6s:   52.7565 Mbit/s
2.7s:   55.3472 Mbit/s
2.8s:   48.1638 Mbit/s
2.9s:   56.0538 Mbit/s
3s:     56.6426 Mbit/s
3.1s:   56.0538 Mbit/s
3.2s:   50.048 Mbit/s
3.3s:   49.6947 Mbit/s
3.4s:   54.5229 Mbit/s
3.5s:   50.519 Mbit/s
3.6s:   57.5846 Mbit/s
3.7s:   49.4592 Mbit/s
3.8s:   49.577 Mbit/s
3.9s:   52.7565 Mbit/s
4s:     58.409 Mbit/s
4.1s:   54.2874 Mbit/s
4.2s:   48.2816 Mbit/s
4.3s:   56.0538 Mbit/s
4.4s:   50.6368 Mbit/s
4.5s:   46.7507 Mbit/s
4.6s:   51.4611 Mbit/s
```

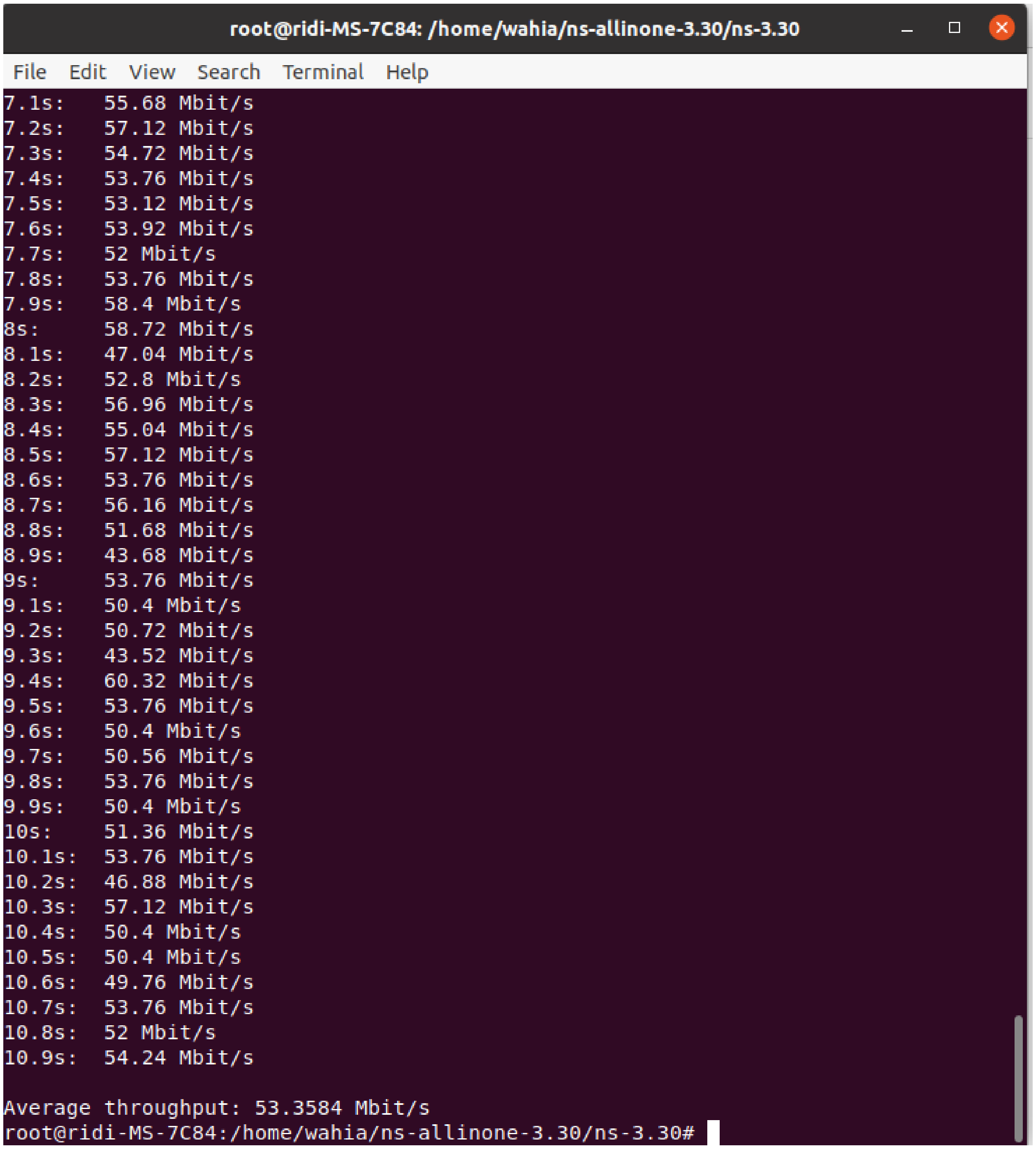

root@ridi-MS-7C84: /home/wahia/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help

```
7.1s:   52.7565 Mbit/s
7.2s:   51.3434 Mbit/s
7.3s:   47.9283 Mbit/s
7.4s:   53.2275 Mbit/s
7.5s:   52.7565 Mbit/s
7.6s:   52.7565 Mbit/s
7.7s:   49.4592 Mbit/s
7.8s:   47.575 Mbit/s
7.9s:   53.463 Mbit/s
8s:     47.9283 Mbit/s
8.1s:   50.048 Mbit/s
8.2s:   52.7565 Mbit/s
8.3s:   52.7565 Mbit/s
8.4s:   54.9939 Mbit/s
8.5s:   55.1117 Mbit/s
8.6s:   50.9901 Mbit/s
8.7s:   50.8723 Mbit/s
8.8s:   49.4592 Mbit/s
8.9s:   52.7565 Mbit/s
9s:     48.2816 Mbit/s
9.1s:   52.0499 Mbit/s
9.2s:   53.463 Mbit/s
9.3s:   52.7565 Mbit/s
9.4s:   44.0422 Mbit/s
9.5s:   56.7603 Mbit/s
9.6s:   56.0538 Mbit/s
9.7s:   42.8646 Mbit/s
9.8s:   56.5248 Mbit/s
9.9s:   53.463 Mbit/s
10s:    49.4592 Mbit/s
10.1s:  53.6986 Mbit/s
10.2s:  52.7565 Mbit/s
10.3s:  54.2874 Mbit/s
10.4s:  48.7526 Mbit/s
10.5s:  49.4592 Mbit/s
10.6s:  51.3434 Mbit/s
10.7s:  49.4592 Mbit/s
10.8s:  50.7546 Mbit/s
10.9s:  50.8723 Mbit/s

Average throughput: 52.1959 Mbit/s
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30#
```

payloadSize=2000
Average throughput: 53.3584 Mbit/s

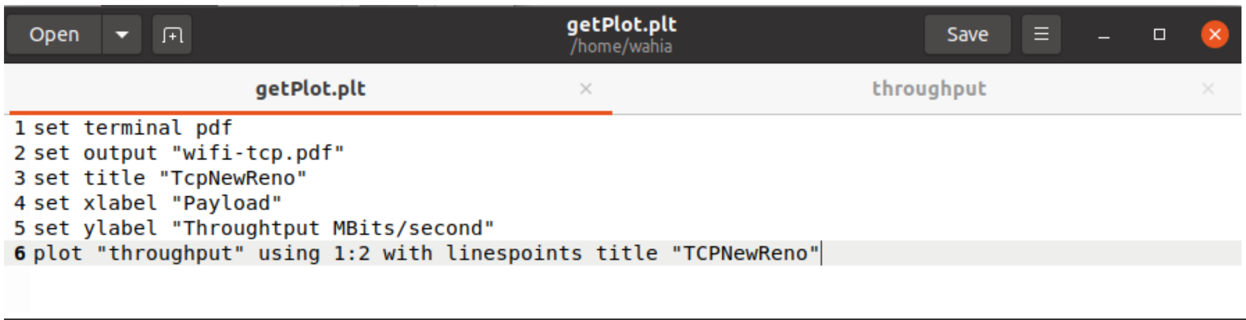

root@ridi-MS-7C84: /home/wahia/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help

```
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30# ./waf --run "scratch/wif
i-tcp --payloadSize=2000"
Waf: Entering directory `/home/wahia/ns-allinone-3.30/ns-3.30/build'
Waf: Leaving directory `/home/wahia/ns-allinone-3.30/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.943s)
1.1s:   35.04 Mbit/s
1.2s:   56.48 Mbit/s
1.3s:   53.76 Mbit/s
1.4s:   51.84 Mbit/s
1.5s:   60.48 Mbit/s
1.6s:   53.92 Mbit/s
1.7s:   53.76 Mbit/s
1.8s:   53.6 Mbit/s
1.9s:   48.96 Mbit/s
2s:     57.12 Mbit/s
2.1s:   57.44 Mbit/s
2.2s:   53.76 Mbit/s
2.3s:   52.64 Mbit/s
2.4s:   52.32 Mbit/s
2.5s:   59.84 Mbit/s
2.6s:   53.28 Mbit/s
2.7s:   55.68 Mbit/s
2.8s:   46.72 Mbit/s
2.9s:   56.64 Mbit/s
3s:     58.24 Mbit/s
3.1s:   49.44 Mbit/s
3.2s:   57.12 Mbit/s
3.3s:   53.6 Mbit/s
3.4s:   53.28 Mbit/s
3.5s:   50.4 Mbit/s
3.6s:   50.4 Mbit/s
3.7s:   52.8 Mbit/s
3.8s:   53.76 Mbit/s
3.9s:   57.6 Mbit/s
4s:     57.44 Mbit/s
4.1s:   50.56 Mbit/s
4.2s:   57.12 Mbit/s
4.3s:   49.44 Mbit/s
4.4s:   47.52 Mbit/s
4.5s:   51.84 Mbit/s
4.6s:   58.08 Mbit/s
```

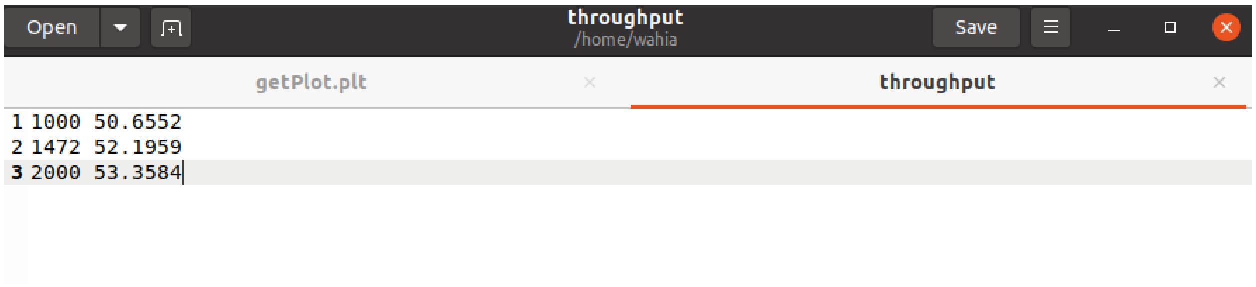

root@ridi-MS-7C84: /home/wahia/ns-allinone-3.30/ns-3.30
File Edit View Search Terminal Help

```
7.1s:   55.68 Mbit/s
7.2s:   57.12 Mbit/s
7.3s:   54.72 Mbit/s
7.4s:   53.76 Mbit/s
7.5s:   53.12 Mbit/s
7.6s:   53.92 Mbit/s
7.7s:   52 Mbit/s
7.8s:   53.76 Mbit/s
7.9s:   58.4 Mbit/s
8s:     58.72 Mbit/s
8.1s:   47.04 Mbit/s
8.2s:   52.8 Mbit/s
8.3s:   56.96 Mbit/s
8.4s:   55.04 Mbit/s
8.5s:   57.12 Mbit/s
8.6s:   53.76 Mbit/s
8.7s:   56.16 Mbit/s
8.8s:   51.68 Mbit/s
8.9s:   43.68 Mbit/s
9s:     53.76 Mbit/s
9.1s:   50.4 Mbit/s
9.2s:   50.72 Mbit/s
9.3s:   43.52 Mbit/s
9.4s:   60.32 Mbit/s
9.5s:   53.76 Mbit/s
9.6s:   50.4 Mbit/s
9.7s:   50.56 Mbit/s
9.8s:   53.76 Mbit/s
9.9s:   50.4 Mbit/s
10s:    51.36 Mbit/s
10.1s:  53.76 Mbit/s
10.2s:  46.88 Mbit/s
10.3s:  57.12 Mbit/s
10.4s:  50.4 Mbit/s
10.5s:  50.4 Mbit/s
10.6s:  49.76 Mbit/s
10.7s:  53.76 Mbit/s
10.8s:  52 Mbit/s
10.9s:  54.24 Mbit/s

Average throughput: 53.3584 Mbit/s
root@ridi-MS-7C84:/home/wahia/ns-allinone-3.30/ns-3.30#
```

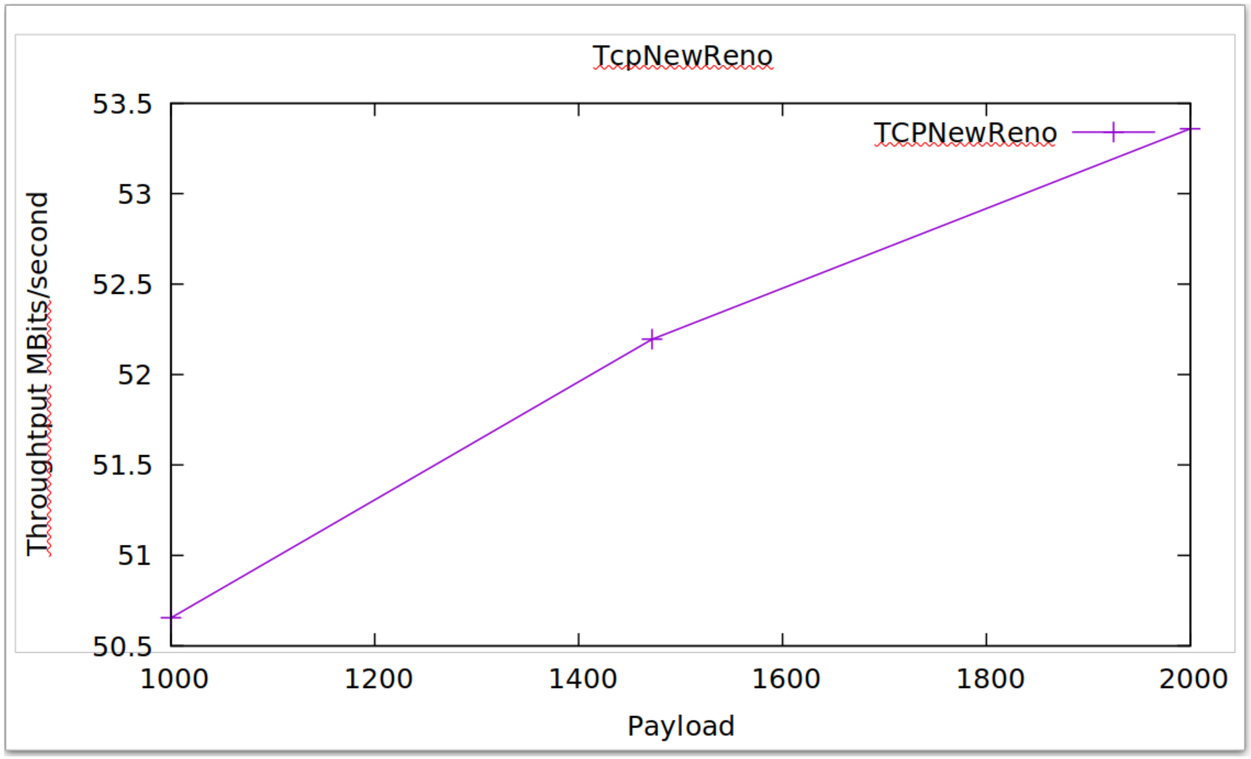
Plotting the throughput:



```
set terminal pdf
set output "wifi-tcp.pdf"
set title "TcpNewReno"
set xlabel "Payload"
set ylabel "Throughtput MBits/second"
plot "throughput" using 1:2 with linespoints title "TCPNewReno"
```



```
1000 50.6552
1472 52.1959
2000 53.3584
```



<u>Conclusion:</u>

From this lab, we learned about TCP internals and the difference between each of the variants using NS-3 tracing mechanism. For this we first compile , execute and calculate the throughput and then compare the various TCP protocols in NS3 for wireless network.