运用 ns-3 模拟数据中心（datacenter）实验报告

一.实验工具

Ubuntu、Vmware 虚拟机、ns-3 仿真器、Wireshark 抓包工具

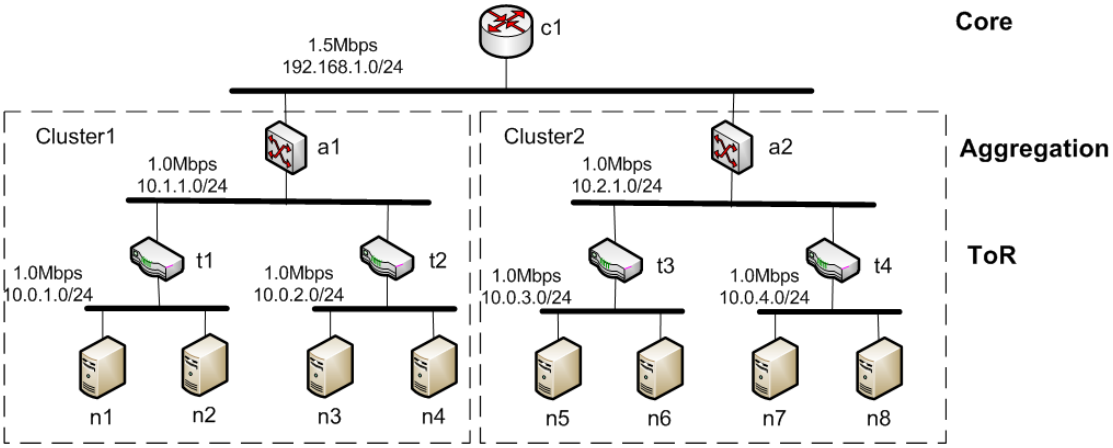
二.实验步骤

1.编写程序模拟网络拓扑及网络消息传输状态（多对多和多对一）

2.利用 Wireshark 抓包工具对输出的.pcap 文件进行分析

3.实验总结

三.网络拓扑



图（1）

四.代码以及解释

库函数调用

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;
```

```
NS_LOG_COMPONENT_DEFINE ("DataCenter1");

int main (int argc, char *argv[])
{
    bool verbose = true;

    if (verbose)
    {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

 //建立上节点的网络
    NodeContainer coragg1;
    coragg1.Create(2);

    NodeContainer coragg2;
    coragg2.Add(coragg1.Get(0));
    coragg2.Create(1);


    //建立中间部分的节点
    NodeContainer aggrtoR1,aggrtoR2;
    aggrtoR1.Add(coragg1.Get(1));
    aggrtoR1.Create(2);
    aggrtoR2.Add(coragg2.Get(1));
    aggrtoR2.Create(2);

    //建立底层节点 csmaNodes
```

```
NodeContainer csmaNodes1,csmaNodes2,csmaNodes3,csmaNodes4;

csmaNodes1.Add(aggrtoR1.Get(1));
csmaNodes1.Create(2);
csmaNodes2.Add(aggrtoR1.Get(2));
csmaNodes2.Create(2);
csmaNodes3.Add(aggrtoR2.Get(1));
csmaNodes3.Create(2);
csmaNodes4.Add(aggrtoR2.Get(2));
csmaNodes4.Create(2);



//建立上层拓扑
PointToPointHelper ptp1,ptp2;
ptp1.SetDeviceAttribute ("DataRate", StringValue ("1.5Mbps"));
ptp1.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (500)));
ptp2.SetDeviceAttribute ("DataRate", StringValue ("1.5Mbps"));
ptp2.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (500)));

NetDeviceContainer devicePtp1,devicePtp2;
devicePtp1=ptp1.Install (coragg1);
devicePtp2=ptp2.Install (coragg2);

//建立 csma 拓扑
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("1Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (500)));
```

```
    //中层拓扑
    NetDeviceContainer  deviceAggToR1,deviceAggToR2;
    deviceAggToR1=csma.Install (aggrtoR1);
    deviceAggToR2=csma.Install (aggrtoR2);

    //底层拓扑
    NetDeviceContainer
deviceCsmaNodes1,deviceCsmaNodes2,deviceCsmaNodes3,deviceCsmaNodes4;
    deviceCsmaNodes1=csma.Install (csmaNodes1);
    deviceCsmaNodes2=csma.Install (csmaNodes2);
    deviceCsmaNodes3=csma.Install (csmaNodes3);
    deviceCsmaNodes4=csma.Install (csmaNodes4);

    //安装协议栈
    InternetStackHelper stack;
    stack.Install (coragg1);
    stack.Install (coragg2.Get(1));
    stack.Install (aggrtoR1.Get(1));
    stack.Install (aggrtoR1.Get(2));
    stack.Install (aggrtoR2.Get(1));
    stack.Install (aggrtoR2.Get(2));
    stack.Install (csmaNodes1.Get(1));
    stack.Install (csmaNodes1.Get(2));
    stack.Install (csmaNodes2.Get(1));
    stack.Install (csmaNodes2.Get(2));
    stack.Install (csmaNodes3.Get(1));
    stack.Install (csmaNodes3.Get(2));
    stack.Install (csmaNodes4.Get(1));
    stack.Install (csmaNodes4.Get(2));
```

```
    //分配网络上的地址
    Ipv4AddressHelper address;

    address.SetBase ("192.168.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesPtp1 = address.Assign (devicePtp1);

    address.SetBase ("192.168.2.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesPtp2 = address.Assign (devicePtp2);

    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesAggToR1 = address.Assign
(deviceAggToR1);

    address.SetBase ("10.2.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesAggToR2 = address.Assign
(deviceAggToR2);

    address.SetBase ("10.0.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesCsma1 = address.Assign
(deviceCsmaNodes1);

    address.SetBase ("10.0.2.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesCsma2 = address.Assign
(deviceCsmaNodes2);

    address.SetBase ("10.0.3.0", "255.255.255.0");
    Ipv4InterfaceContainer interfacesCsma3 = address.Assign
(deviceCsmaNodes3);

    address.SetBase ("10.0.4.0", "255.255.255.0");
```

```
    Ipv4InterfaceContainer interfacesCsma4 = address.Assign
(deviceCsmaNodes4);



    //设置 n1 到 n5 的通路

    PacketSinkHelper packetSinkHelper1("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma3.GetAddress(1),8080));

    ApplicationContainer sinkApp1 =
packetSinkHelper1.Install(csmaNodes3.Get(1));



    sinkApp1.Start(Seconds(0.0));

    sinkApp1.Stop(Seconds(20.0));



    OnOffHelper client1("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma3.GetAddress(1), 8080));

    client1.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

    client1.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

    client1.SetAttribute ("DataRate", DataRateValue (DataRate
("1.0Mbps")));

    client1.SetAttribute ("PacketSize", UintegerValue (2000));



    ApplicationContainer clientApp1 = client1.Install (csmaNodes1.Get(1));

    clientApp1.Start(Seconds (1.0 ));

    clientApp1.Stop (Seconds (21.0));



    csma.EnablePcap ("Pattern1 n1 to n5", deviceCsmaNodes3.Get (1), true);



    //设置 n2 到 n6 的通路
```

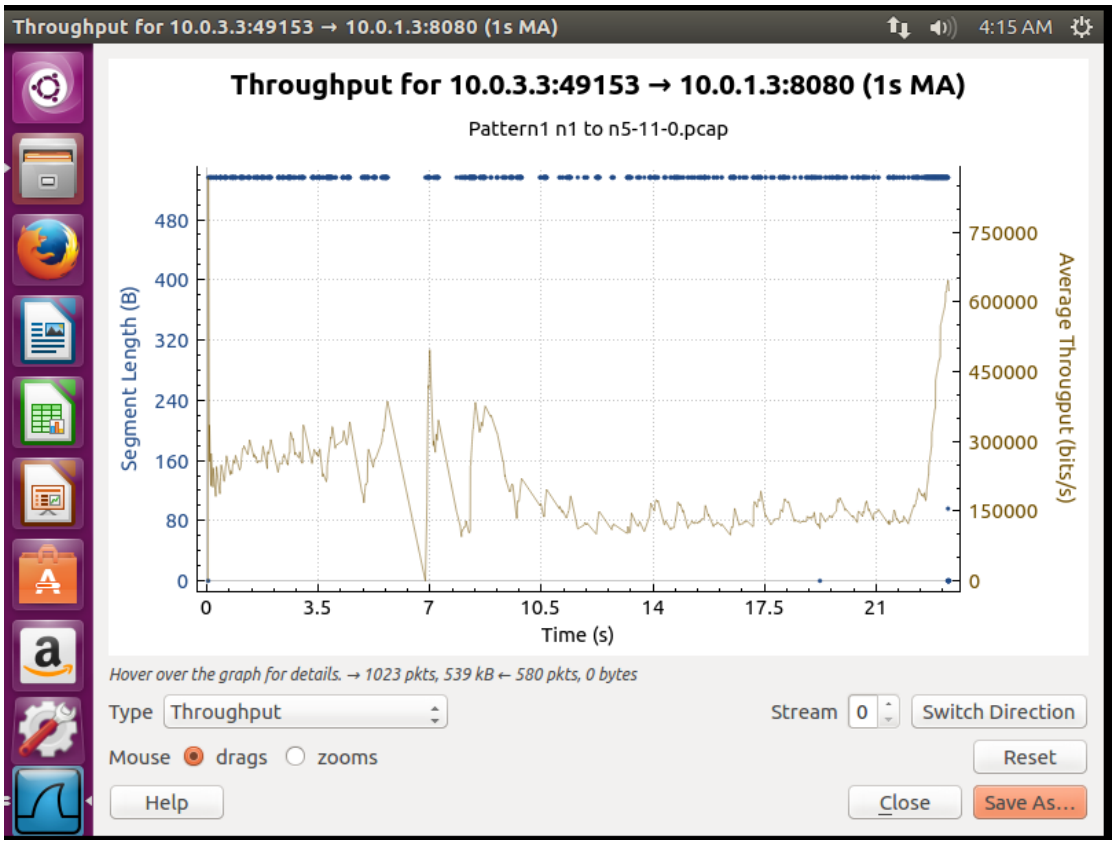
```
PacketSinkHelper
packetSinkHelper2("ns3::TcpSocketFactory",InetSocketAddress(interfacesCsma1.GetAd
dress(2),8080));

ApplicationContainer sinkApp2 = packetSinkHelper2.Install(csmaNodes1.Get(2));

sinkApp2.Start(Seconds(0.0));

sinkApp2.Stop(Seconds(20.0));


OnOffHelper client2("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(2), 8080));

client2.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client2.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client2.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client2.SetAttribute ("PacketSize", UintegerValue (2000));

ApplicationContainer clientApp2 = client2.Install (csmaNodes3.Get(2));

clientApp2.Start(Seconds (1.0 ));

clientApp2.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern1 n6 to n2", deviceCsmaNodes1.Get (2), true);


csma.EnablePcap ("Pattern1 n6 to n2", deviceCsmaNodes1.Get (2), true);



//设置 n7 到 n3 的通路

PacketSinkHelper
packetSinkHelper3("ns3::TcpSocketFactory",InetSocketAddress(interfacesCsma4.GetAd
dress(1),8080));

ApplicationContainer sinkApp3 = packetSinkHelper3.Install(csmaNodes4.Get(1));

sinkApp3.Start(Seconds(0.0));

sinkApp3.Stop(Seconds(20.0));


OnOffHelper client3("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma4.GetAddress(1), 8080));
```

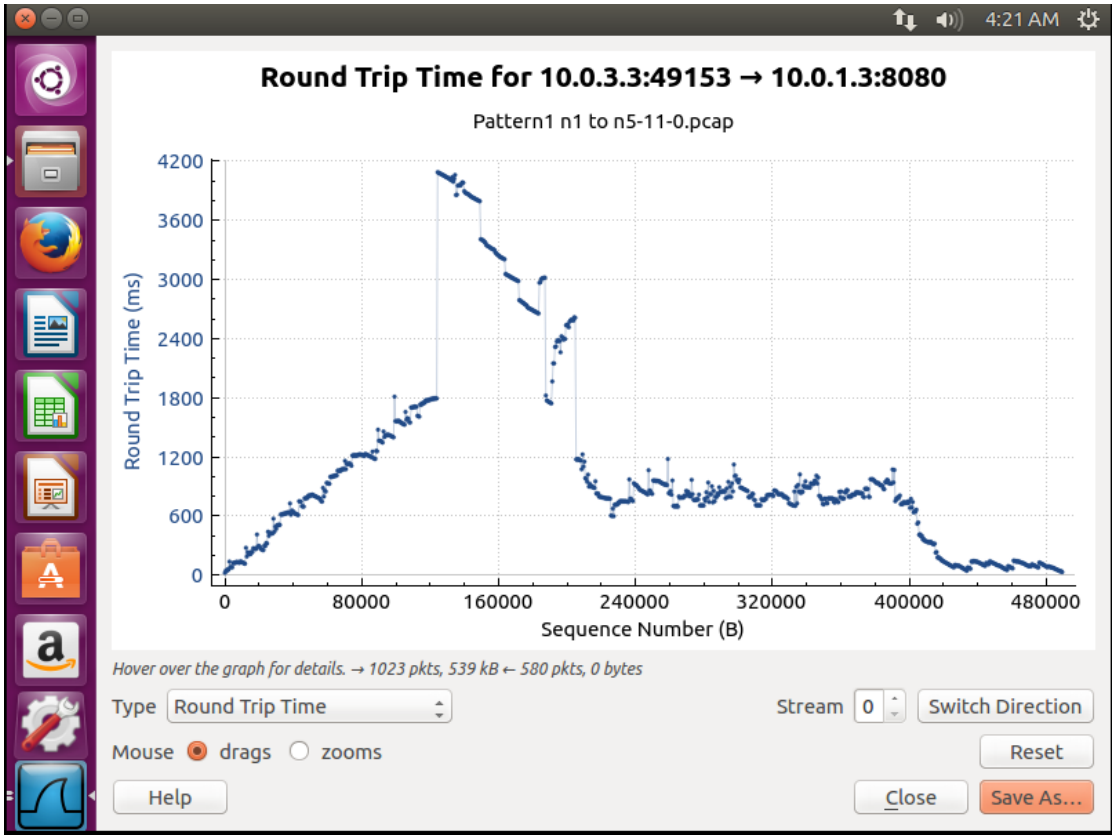
```
client3.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client3.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client3.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client3.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp3 = client3.Install (csmaNodes2.Get(1));

clientApp3.Start(Seconds (1.0 ));

clientApp3.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern1 n3 to n7", deviceCsmaNodes4.Get (1), true);


//设置 n4 到 n8 的通路

PacketSinkHelper
packetSinkHelper4("ns3::TcpSocketFactory",InetSocketAddress(interfacesCsma2.GetAd
dress(2),8080));

ApplicationContainer sinkApp4 = packetSinkHelper4.Install(csmaNodes4.Get(2));

sinkApp4.Start(Seconds(0.0));

sinkApp4.Stop(Seconds(20.0));


OnOffHelper client4("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma2.GetAddress(2), 8080));

client4.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client4.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client4.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client4.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp4 = client4.Install (csmaNodes4.Get(2));

clientApp4.Start(Seconds (1.0 ));

clientApp4.Stop (Seconds (21.0));
```

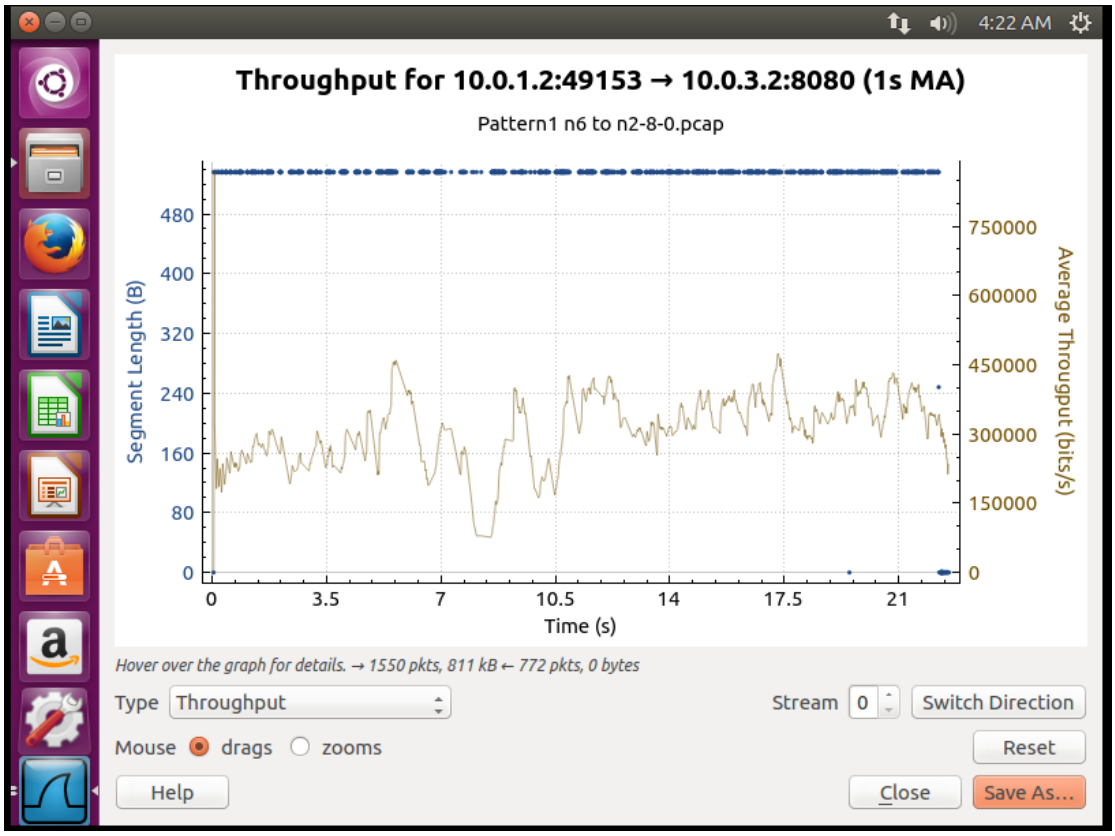
```
    csma.EnablePcap ("Pattern1 n8 to n4", deviceCsmaNodes2.Get (2), true);


    ptp1.EnablePcapAll("Pattern1 ptp1");
    ptp2.EnablePcapAll("Pattern1 ptp2");
    Ipv4GlobalRoutingHelper::PopulateRoutingTables ();



    Simulator::Run ();
    Simulator::Destroy ();
    return 0;
}
```

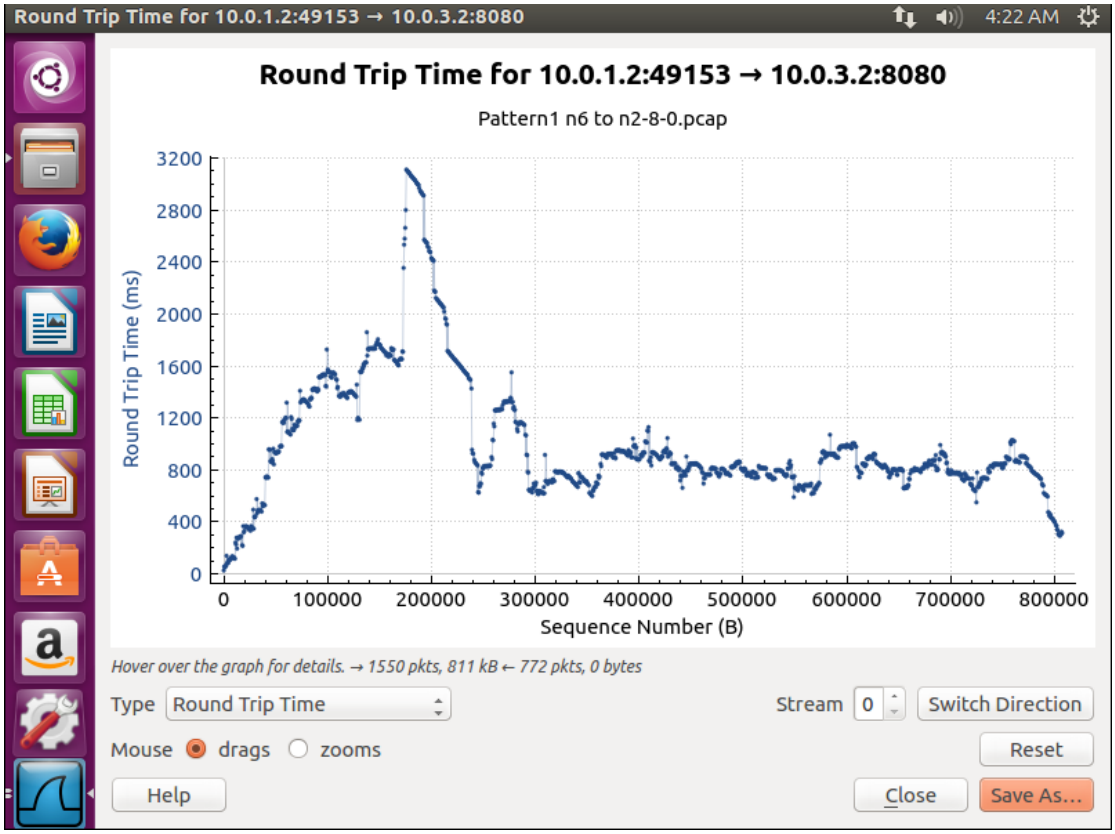
通过运行代码，可输出四个文件，分别为：Pattern1 n1 to n5-11-0.pcap，Pattern1 n3 to n7-13-0.pcap，Pattern1 n6 to n2-8-0.pcap，Pattern1 n8 to n4-10-0.pcap.

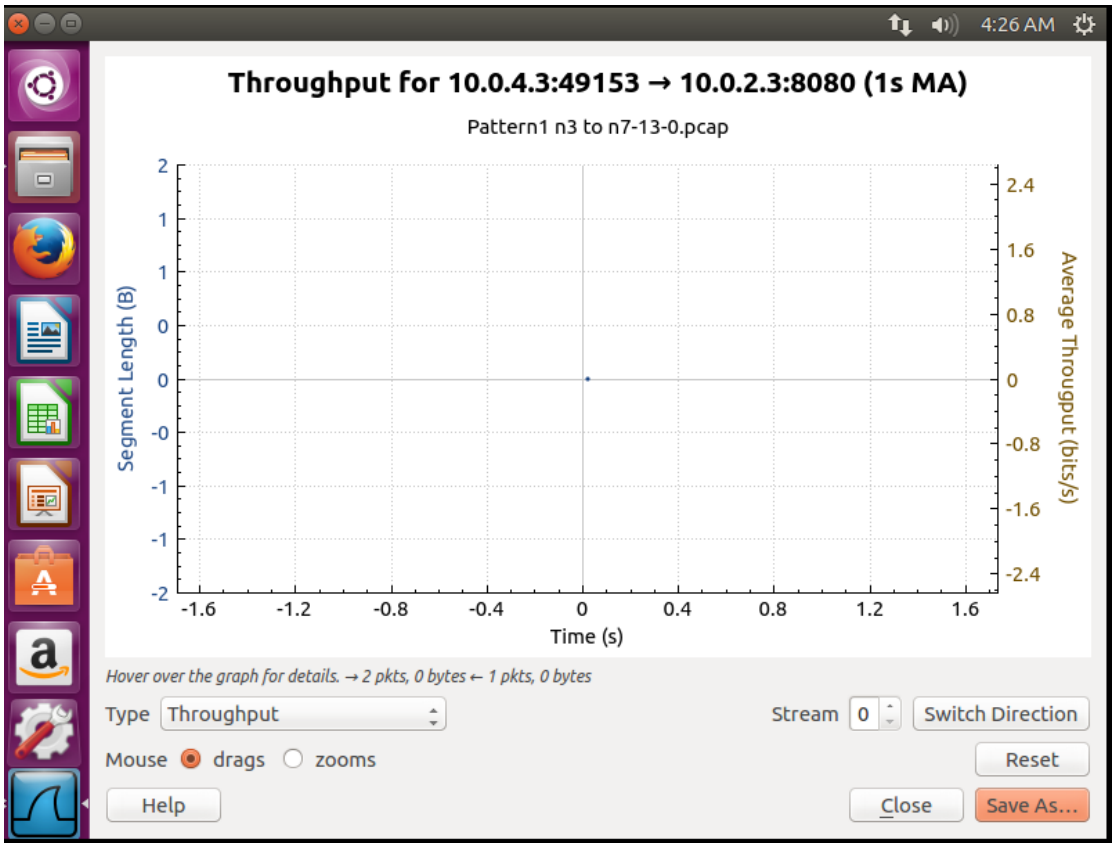
对以上四个文件通过 Wireshark 进行分析，各主机对的吞吐量和 rrt 如图所示：

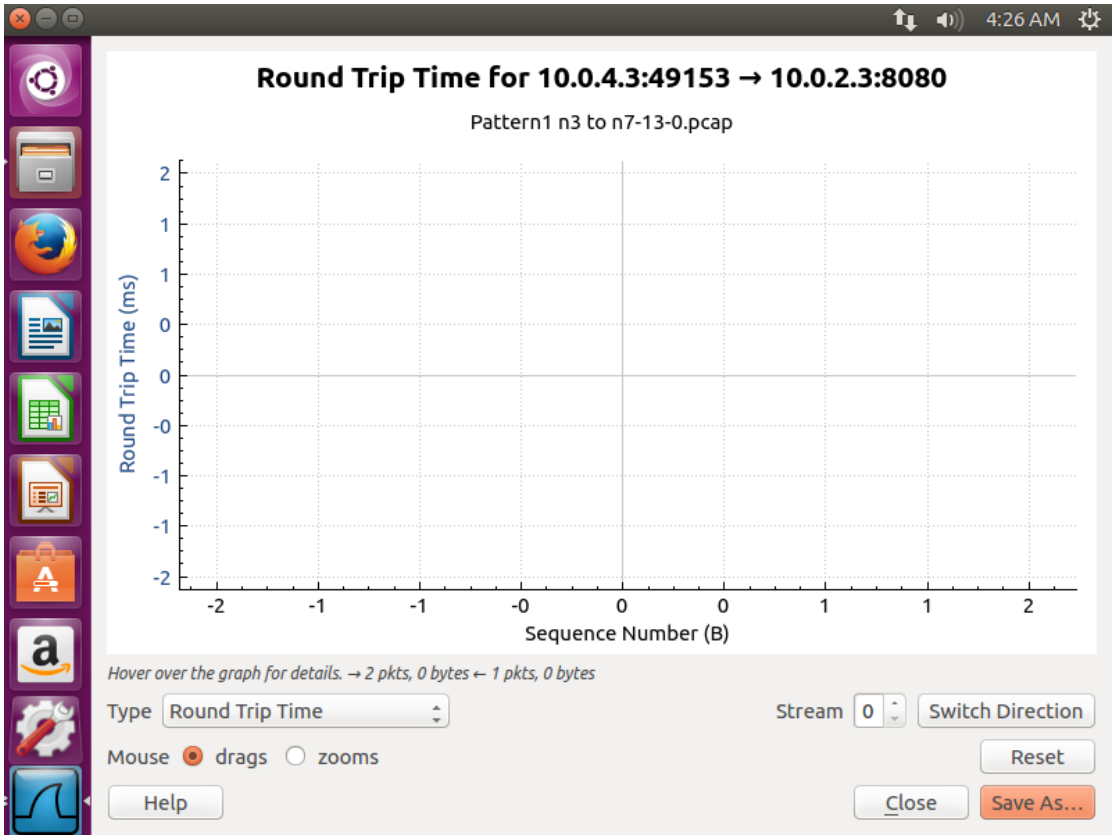


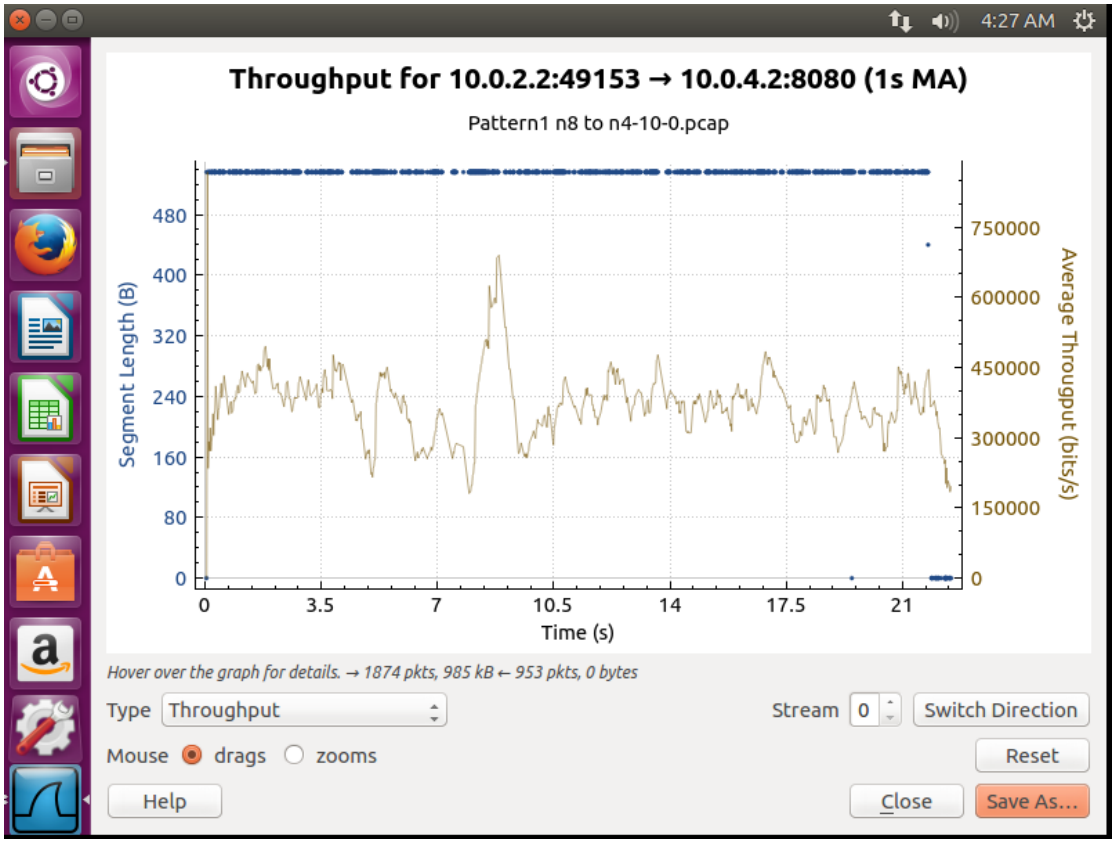

4:21 AM
Round Trip Time for 10.0.3.3:49153 → 10.0.1.3:8080
Pattern1 n1 to n5-11-0.pcap
Round Trip Time (ms)
4200
3600
3000
2400
1800
1200
600
0
0
80000
160000
240000
320000
400000
480000
Sequence Number (B)
Hover over the graph for details. → 1023 pkts, 539 kB ← 580 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

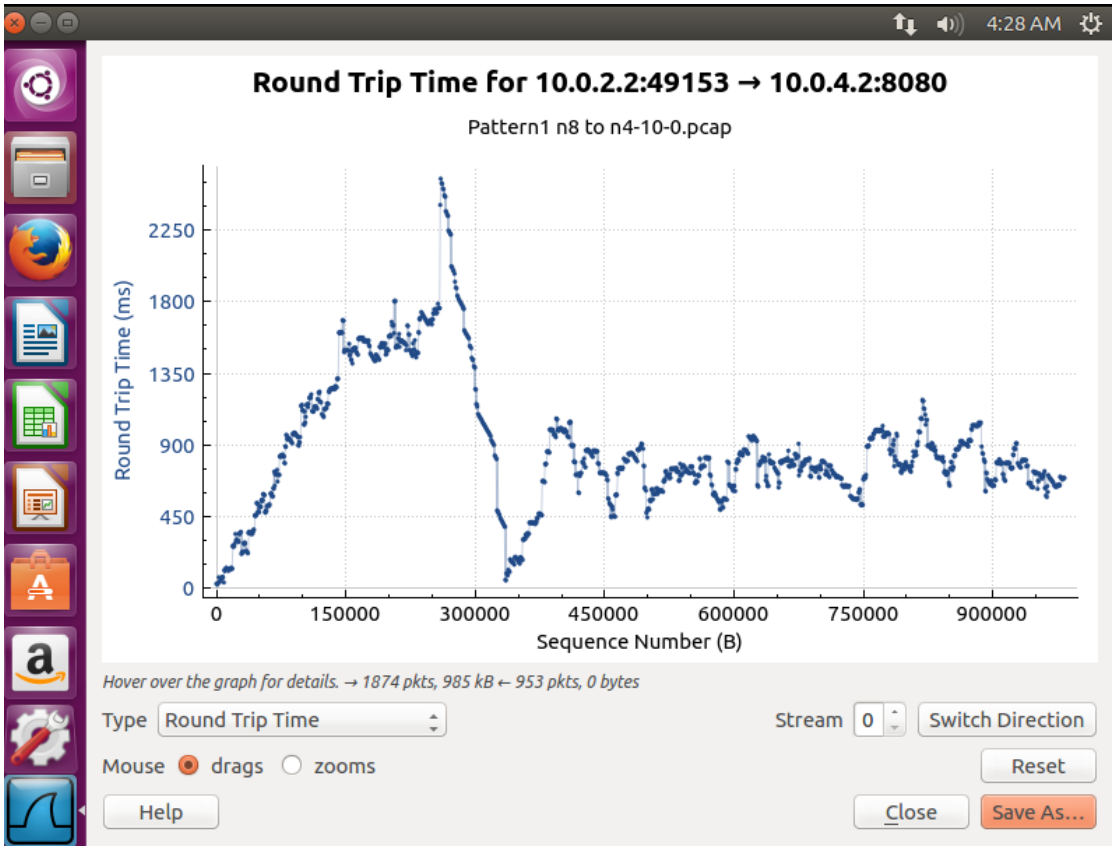

4:22 AM
Throughput for 10.0.1.2:49153 → 10.0.3.2:8080 (1s MA)
Pattern1 n6 to n2-8-0.pcap
Segment Length (B)
480
400
320
240
160
80
0
750000
600000
450000
300000
150000
0
Average Throughput (bits/s)
0
3.5
7
10.5
14
17.5
21
Time (s)
Hover over the graph for details. → 1550 pkts, 811 kB ← 772 pkts, 0 bytes
Type
Throughput
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...


Round Trip Time for 10.0.1.2:49153 → 10.0.3.2:8080
4:22 AM
Round Trip Time for 10.0.1.2:49153 → 10.0.3.2:8080
Pattern1 n6 to n2-8-0.pcap
Round Trip Time (ms)
3200
2800
2400
2000
1600
1200
800
400
0
0
100000
200000
300000
400000
500000
600000
700000
800000
Sequence Number (B)
Hover over the graph for details. → 1550 pkts, 811 kB ← 772 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...


4:26 AM
Throughput for 10.0.4.3:49153 → 10.0.2.3:8080 (1s MA)
Pattern1 n3 to n7-13-0.pcap
Segment Length (B)
2
1
1
0
0
-0
-1
-1
-2
-1.6
-1.2
-0.8
-0.4
0
0.4
0.8
1.2
1.6
Time (s)
2.4
1.6
0.8
0
-0.8
-1.6
-2.4
Average Throughput (bits/s)
Hover over the graph for details. → 2 pkts, 0 bytes ← 1 pkts, 0 bytes
Type
Throughput
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...


4:26 AM
Round Trip Time for 10.0.4.3:49153 → 10.0.2.3:8080
Pattern1 n3 to n7-13-0.pcap
Round Trip Time (ms)
Sequence Number (B)
Hover over the graph for details. → 2 pkts, 0 bytes ← 1 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...


4:27 AM
Throughput for 10.0.2.2:49153 → 10.0.4.2:8080 (1s MA)
Pattern1 n8 to n4-10-0.pcap
Segment Length (B)
Average Throughput (bits/s)
Time (s)
Hover over the graph for details. → 1874 pkts, 985 kB ← 953 pkts, 0 bytes
Type
Throughput
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...



Pattern 1 分析

1.从数据包流向可以看出，数据包是按照先前预想的进行发送和接收，所以数据包流向是正常的。

2.从 TCP throughput 来分析，可以看出四条链路的吞吐量的变化情况较大，但是从观察可以看出四条链路的平均带宽差不多。 说明，四条链路之间是存在竞争的，并且该竞争较为公平。

2.Pattern 2

//库函数调用

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
```

```
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("DataCenter2");

int main (int argc, char *argv[])
{
    bool verbose = true;

    if (verbose)
    {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }


    //建立上层节点 ptpNodes
    NodeContainer coragg1;
    coragg1.Create(2);

    NodeContainer coragg2;
    coragg2.Add(coragg1.Get(0));
    coragg2.Create(1);


    //建立中间网络  aggrtoR1:
aggregation1->toRswitch1,aggrtoR2:aggregation2->toRswitch2
    NodeContainer aggrtoR1,aggrtoR2;
    aggrtoR1.Add(coragg1.Get(1));
```

```
aggrtoR1.Create(2);
aggrtoR2.Add(coragg2.Get(1));
aggrtoR2.Create(2);

//建立底层网络 csmaNodes
NodeContainer csmaNodes1,csmaNodes2,csmaNodes3,csmaNodes4;

csmaNodes1.Add(aggrtoR1.Get(1));
csmaNodes1.Create(2);
csmaNodes2.Add(aggrtoR1.Get(2));
csmaNodes2.Create(2);
csmaNodes3.Add(aggrtoR2.Get(1));
csmaNodes3.Create(2);
csmaNodes4.Add(aggrtoR2.Get(2));
csmaNodes4.Create(2);



//建立上层 ptp 拓扑
PointToPointHelper ptp1,ptp2;
ptp1.SetDeviceAttribute ("DataRate", StringValue ("1.5Mbps"));
ptp1.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (500)));
ptp2.SetDeviceAttribute ("DataRate", StringValue ("1.5Mbps"));
ptp2.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (500)));

NetDeviceContainer devicePtp1,devicePtp2;
devicePtp1=ptp1.Install (coragg1);
devicePtp2=ptp2.Install (coragg2);

//建立 csma 拓扑
```

```
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("1Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (500)));

//中间网络的拓扑
NetDeviceContainer    deviceAggToR1,deviceAggToR2;
deviceAggToR1=csma.Install (aggrtoR1);
deviceAggToR2=csma.Install (aggrtoR2);

//底层网络拓扑
NetDeviceContainer
deviceCsmaNodes1,deviceCsmaNodes2,deviceCsmaNodes3,deviceCsmaNodes4;
deviceCsmaNodes1=csma.Install (csmaNodes1);
deviceCsmaNodes2=csma.Install (csmaNodes2);
deviceCsmaNodes3=csma.Install (csmaNodes3);
deviceCsmaNodes4=csma.Install (csmaNodes4);

//安装协议栈
InternetStackHelper stack;
stack.Install (coragg1);
stack.Install (coragg2.Get(1));
stack.Install (aggrtoR1.Get(1));
stack.Install (aggrtoR1.Get(2));
stack.Install (aggrtoR2.Get(1));
stack.Install (aggrtoR2.Get(2));
stack.Install (csmaNodes1.Get(1));
stack.Install (csmaNodes1.Get(2));
stack.Install (csmaNodes2.Get(1));
stack.Install (csmaNodes2.Get(2));
stack.Install (csmaNodes3.Get(1));
```

```
stack.Install (csmaNodes3.Get(2));
stack.Install (csmaNodes4.Get(1));
stack.Install (csmaNodes4.Get(2));

//为网络上的节点分配 ip 地址
Ipv4AddressHelper address;

address.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesPtp1 = address.Assign (devicePtp1);

address.SetBase ("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesPtp2 = address.Assign (devicePtp2);

address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesAggToR1 = address.Assign (deviceAggToR1);

address.SetBase ("10.2.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesAggToR2 = address.Assign (deviceAggToR2);

address.SetBase ("10.0.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesCsma1 = address.Assign (deviceCsmaNodes1);

address.SetBase ("10.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesCsma2 = address.Assign (deviceCsmaNodes2);

address.SetBase ("10.0.3.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesCsma3 = address.Assign (deviceCsmaNodes3);

address.SetBase ("10.0.4.0", "255.255.255.0");
Ipv4InterfaceContainer interfacesCsma4 = address.Assign (deviceCsmaNodes4);
```

```
//建立一对多模式

//设置 n1 节点为 sinkApp

PacketSinkHelper packetSinkHelper("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1),8080));

ApplicationContainer sinkApp = packetSinkHelper.Install(csmaNodes1.Get(1));


sinkApp.Start(Seconds(0.0));

sinkApp.Stop(Seconds(80.0));


csma.EnablePcap("N1 recieved packets", deviceCsmaNodes1.Get(1),true);




//设置 n2 到 n1

OnOffHelper client1("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client1.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client1.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client1.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client1.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp1 = client1.Install (csmaNodes1.Get(2));

clientApp1.Start(Seconds (1.0 ));

clientApp1.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n2 to n1 ", deviceCsmaNodes1.Get (2), true);
```

```
//设置 n3 到 n1

OnOffHelper client2("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client2.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client2.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client2.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client2.SetAttribute ("PacketSize", UintegerValue (2000));

ApplicationContainer clientApp2 = client2.Install (csmaNodes2.Get(1));

clientApp2.Start(Seconds (1.0 ));

clientApp2.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n3 to n1", deviceCsmaNodes2.Get (1), true);



//设置 n4 到 n1

OnOffHelper client3("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client3.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client3.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client3.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client3.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp3 = client3.Install (csmaNodes2.Get(2));

clientApp3.Start(Seconds (1.0 ));

clientApp3.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n4 to n1", deviceCsmaNodes2.Get (2), true);
```

```
//设置 n5 到 n1

OnOffHelper client4("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client4.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client4.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client4.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client4.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp4 = client4.Install (csmaNodes3.Get(1));

clientApp4.Start(Seconds (1.0 ));

clientApp4.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n5 to n1", deviceCsmaNodes3.Get (1), true);



//设置 n6 到 n1

OnOffHelper client5("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client5.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client5.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client5.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client5.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp5 = client5.Install (csmaNodes3.Get(2));

clientApp5.Start(Seconds (1.0 ));

clientApp5.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n6 to n1", deviceCsmaNodes3.Get (2), true);
```

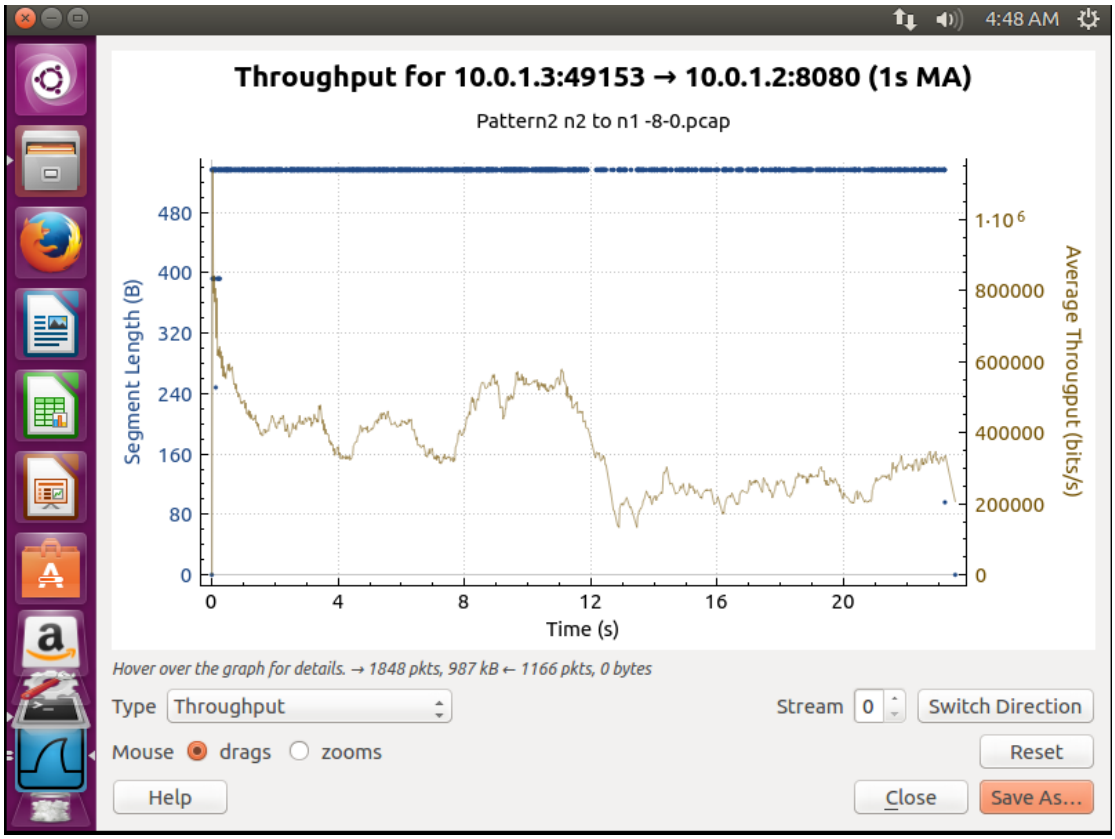
```
//设置 n7 到 n1

OnOffHelper client6("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client6.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client6.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client6.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client6.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp6 = client6.Install (csmaNodes4.Get(1));

clientApp6.Start(Seconds (1.0 ));

clientApp6.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n7 to n1", deviceCsmaNodes4.Get (1), true);


//设置 n8 到 n1

OnOffHelper client7("ns3::TcpSocketFactory",
InetSocketAddress(interfacesCsma1.GetAddress(1), 8080));

client7.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=50]"));

client7.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

client7.SetAttribute ("DataRate", DataRateValue (DataRate ("1.0Mbps")));

client7.SetAttribute ("PacketSize", UintegerValue (2000));


ApplicationContainer clientApp7 = client7.Install (csmaNodes4.Get(2));

clientApp7.Start(Seconds (1.0 ));

clientApp7.Stop (Seconds (21.0));


csma.EnablePcap ("Pattern2 n8 to n1", deviceCsmaNodes4.Get (2), true);
```

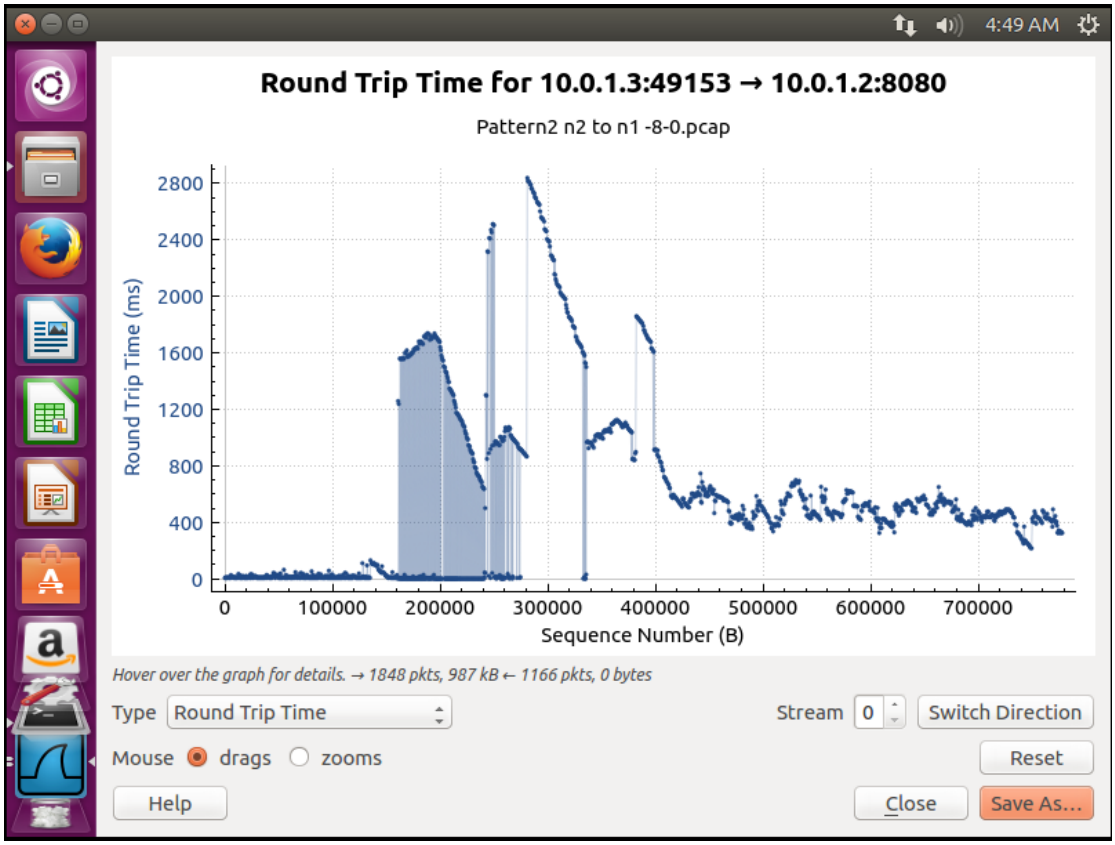
```
    Ipv4GlobalRoutingHelper::PopulateRoutingTables ();


    Simulator::Run ();
    Simulator::Destroy ();
    return 0;
}
```

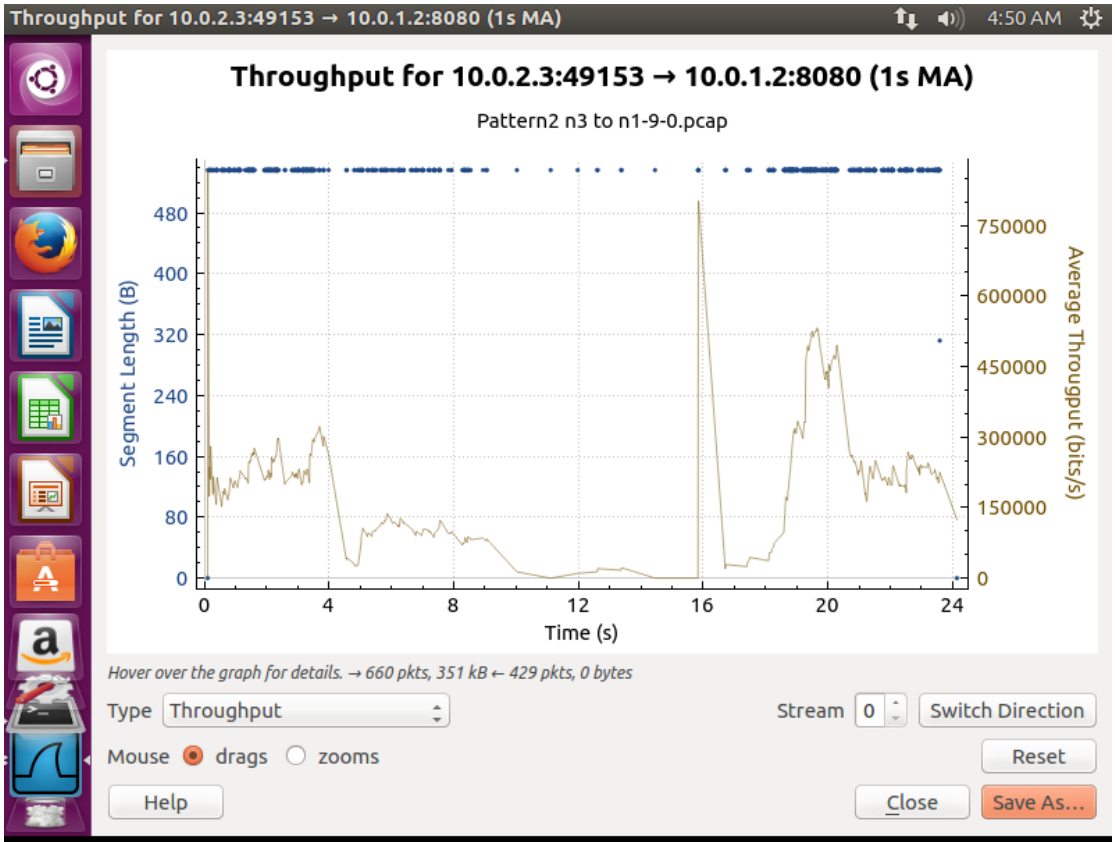
多对一实验仿真结果

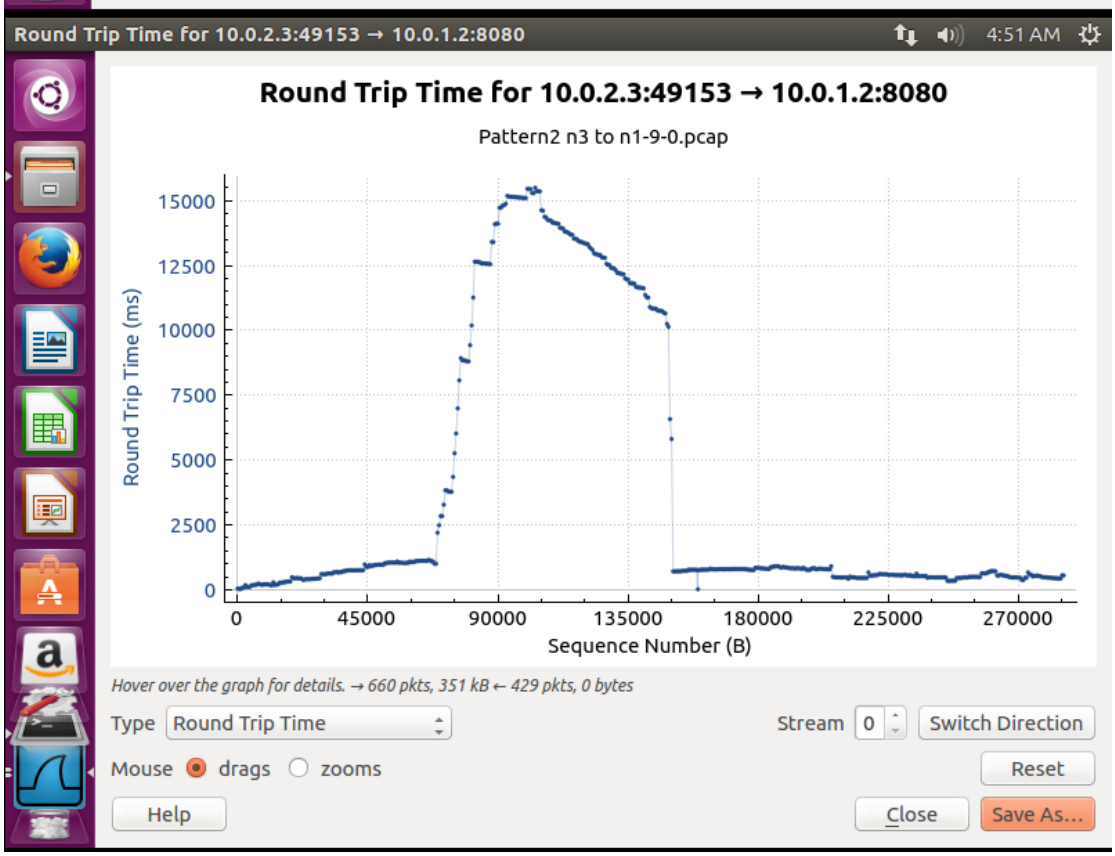
采用 Wireshark 对实验输出的.pcap 文件进行分析。分析位置为主机 h1 的端口。上述程序输出的文件名为：Pattern2 n2 to n1 -8-0.pcap，Pattern2 n3 to n1 -9-0.pcap，Pattern2 n4 to n1 -10-0.pcap，Pattern2 n5 to n1 -11-0.pcap，Pattern2 n6 to n1 -12-0.pcap，Pattern2 n7 to n1 -13-0.pcap，Pattern2 n8 to n1 -14-0.pcap。

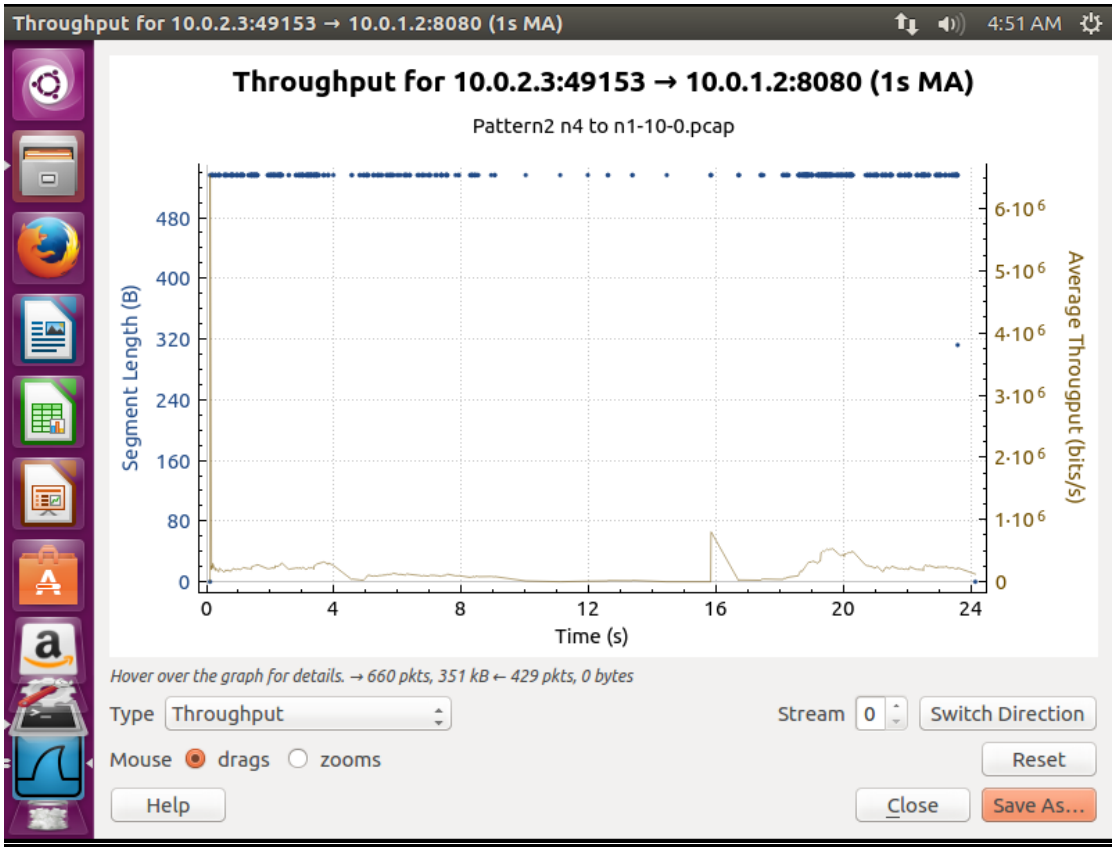
实验结果的吞吐量和 rtt 如下图：

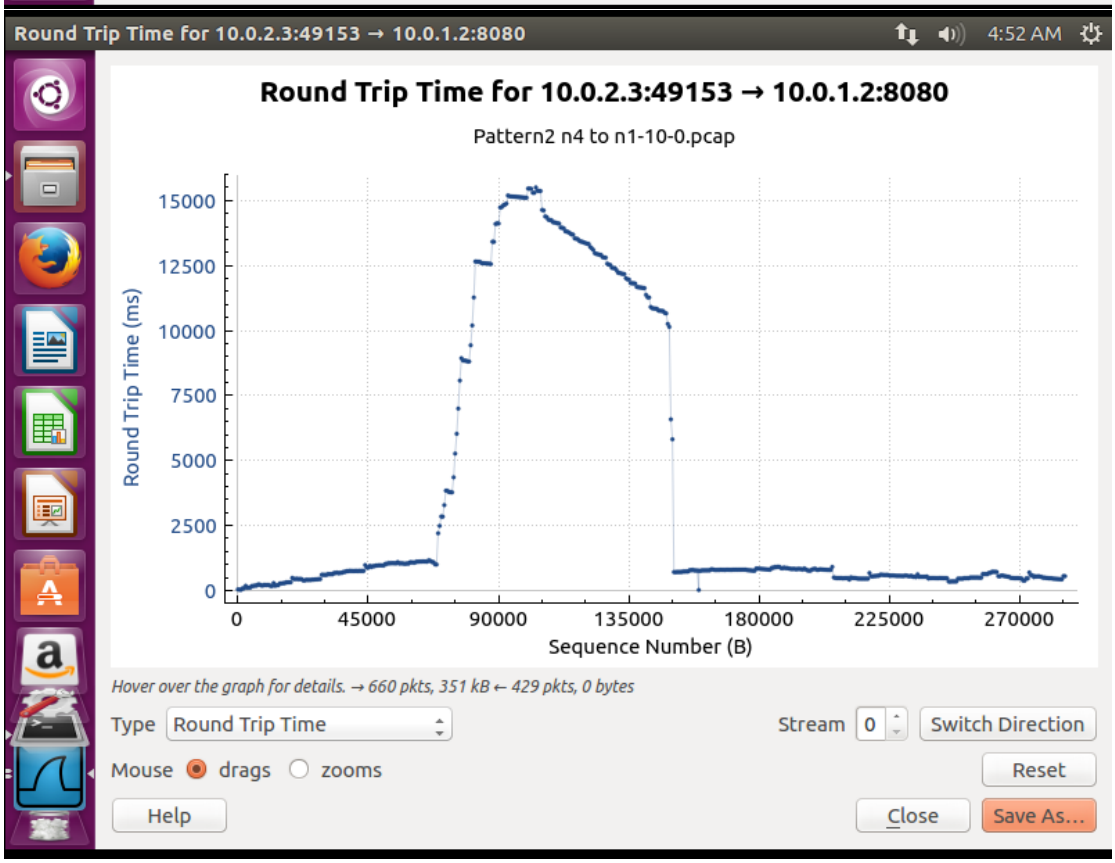


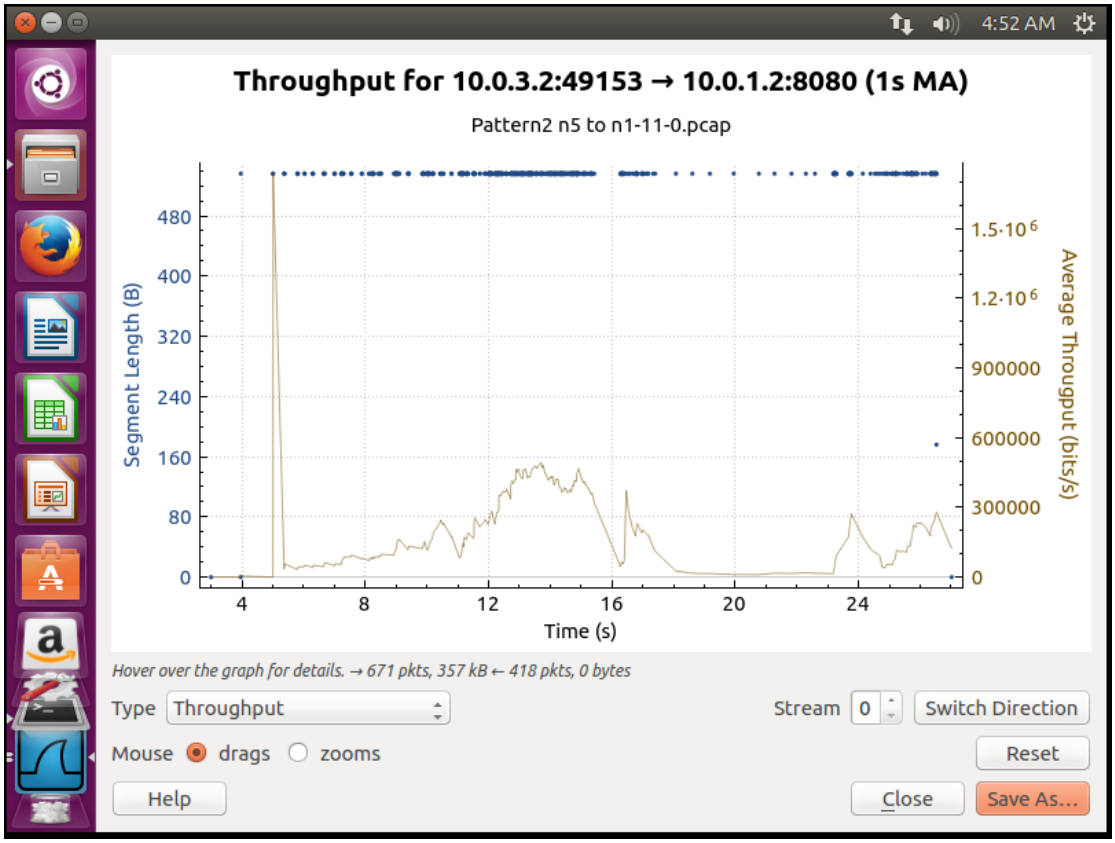

4:49 AM
Round Trip Time for 10.0.1.3:49153 → 10.0.1.2:8080
Pattern2 n2 to n1 -8-0.pcap
Round Trip Time (ms)
2800
2400
2000
1600
1200
800
400
0
0
100000
200000
300000
400000
500000
600000
700000
Sequence Number (B)
Hover over the graph for details. → 1848 pkts, 987 kB ← 1166 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

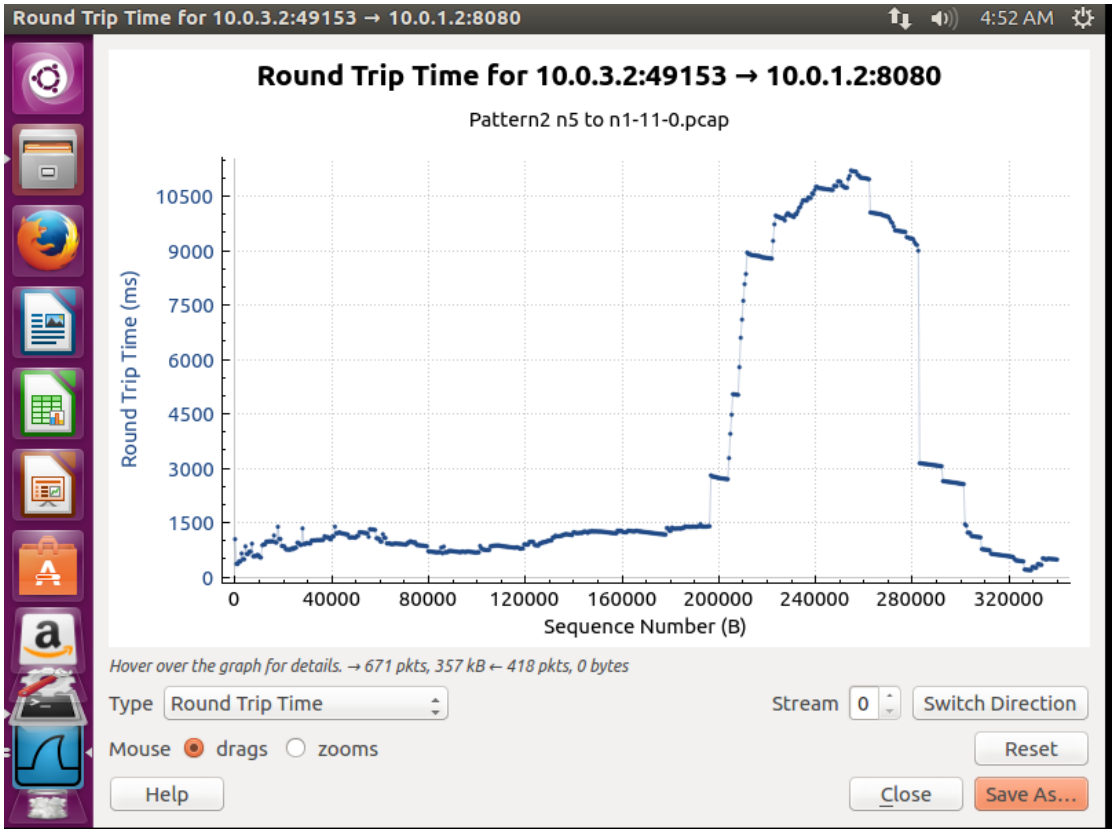

Throughput for 10.0.2.3:49153 → 10.0.1.2:8080 (1s MA)
4:50 AM
Throughput for 10.0.2.3:49153 → 10.0.1.2:8080 (1s MA)
Pattern2 n3 to n1-9-0.pcap
Segment Length (B)
Average Throughput (bits/s)
Time (s)
0
80
160
240
320
400
480
0
150000
300000
450000
600000
750000
0
4
8
12
16
20
24
Hover over the graph for details. → 660 pkts, 351 kB ← 429 pkts, 0 bytes
Type
Throughput
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

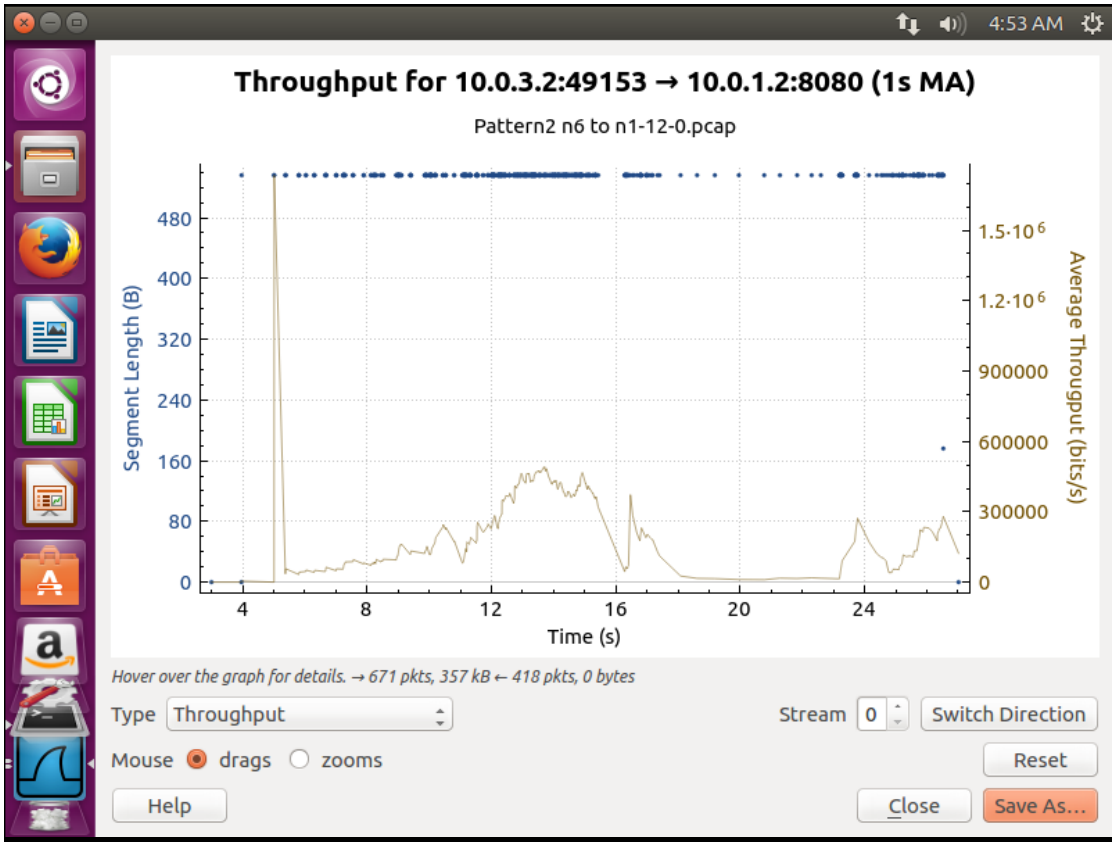

Round Trip Time for 10.0.2.3:49153 → 10.0.1.2:8080
4:51 AM
Round Trip Time for 10.0.2.3:49153 → 10.0.1.2:8080
Pattern2 n3 to n1-9-0.pcap
Round Trip Time (ms)
Sequence Number (B)
0
2500
5000
7500
10000
12500
15000
0
45000
90000
135000
180000
225000
270000
Hover over the graph for details. → 660 pkts, 351 kB ← 429 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

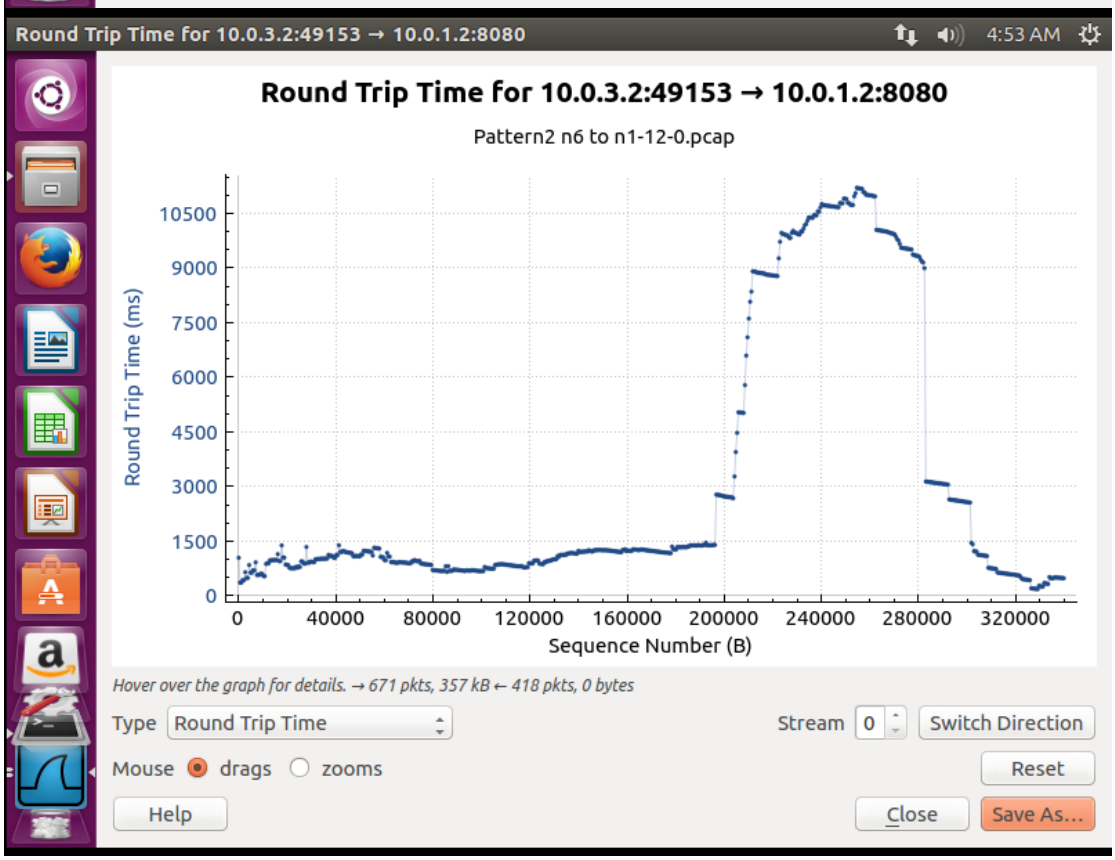

Throughput for 10.0.2.3:49153 → 10.0.1.2:8080 (1s MA)
4:51 AM
Throughput for 10.0.2.3:49153 → 10.0.1.2:8080 (1s MA)
Pattern2 n4 to n1-10-0.pcap
Segment Length (B)
Average Throughput (bits/s)
Time (s)
Hover over the graph for details. → 660 pkts, 351 kB ← 429 pkts, 0 bytes
Type
Throughput
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

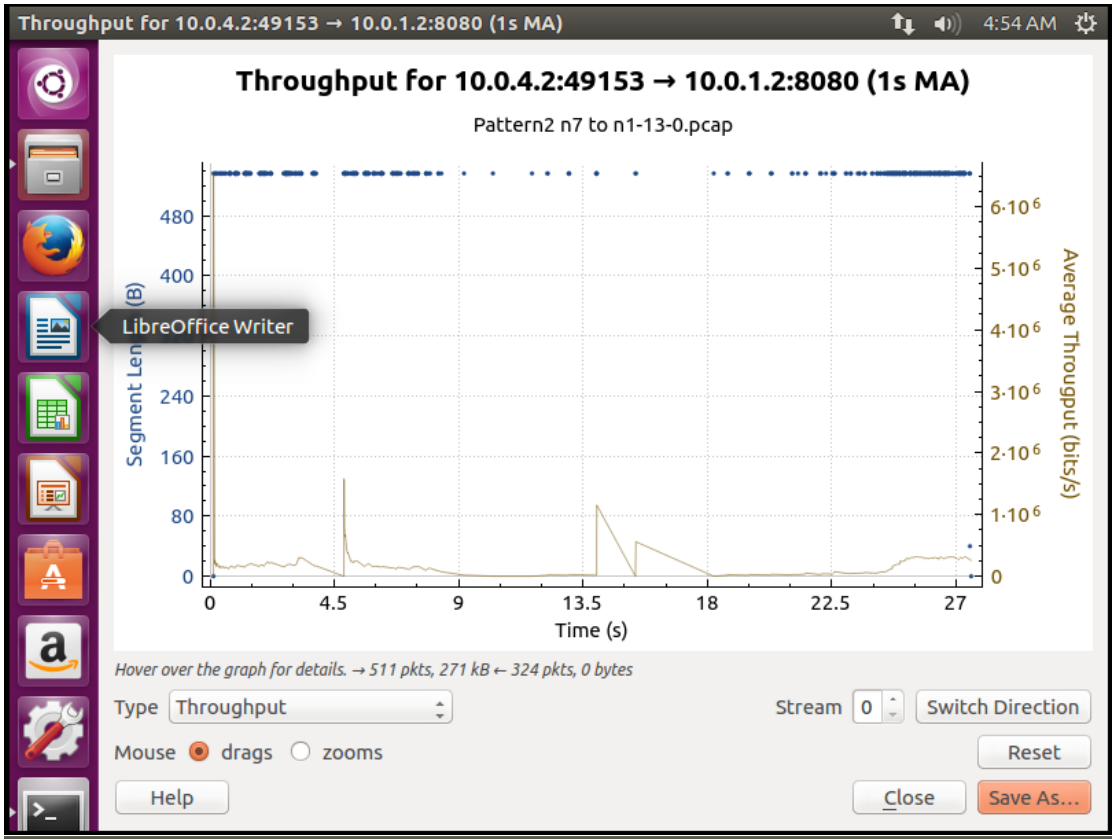

Round Trip Time for 10.0.2.3:49153 → 10.0.1.2:8080
4:52 AM
Round Trip Time for 10.0.2.3:49153 → 10.0.1.2:8080
Pattern2 n4 to n1-10-0.pcap
Round Trip Time (ms)
Sequence Number (B)
Hover over the graph for details. → 660 pkts, 351 kB ← 429 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

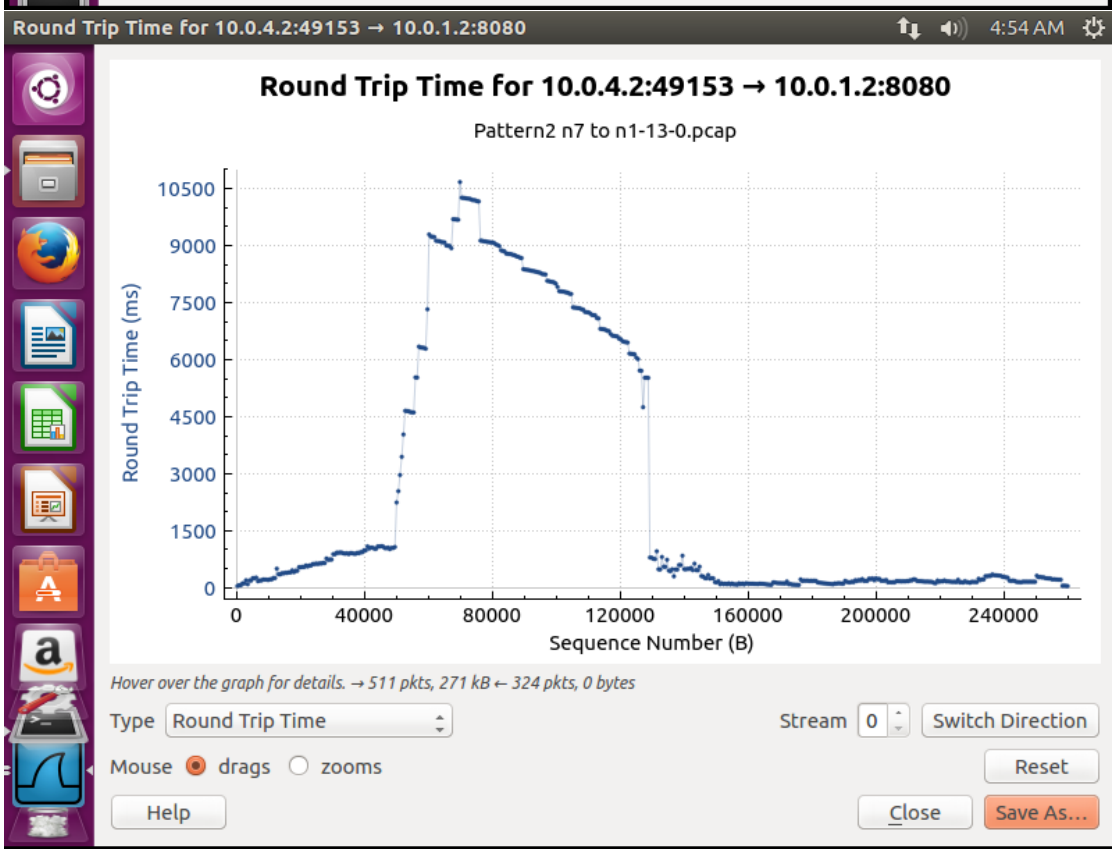

4:52 AM
Throughput for 10.0.3.2:49153 → 10.0.1.2:8080 (1s MA)
Pattern2 n5 to n1-11-0.pcap
Segment Length (B)
480
400
320
240
160
80
0
1.5·10⁶
1.2·10⁶
900000
600000
300000
0
Average Throughput (bits/s)
4
8
12
16
20
24
Time (s)
Hover over the graph for details. → 671 pkts, 357 kB ← 418 pkts, 0 bytes
Type
Throughput
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

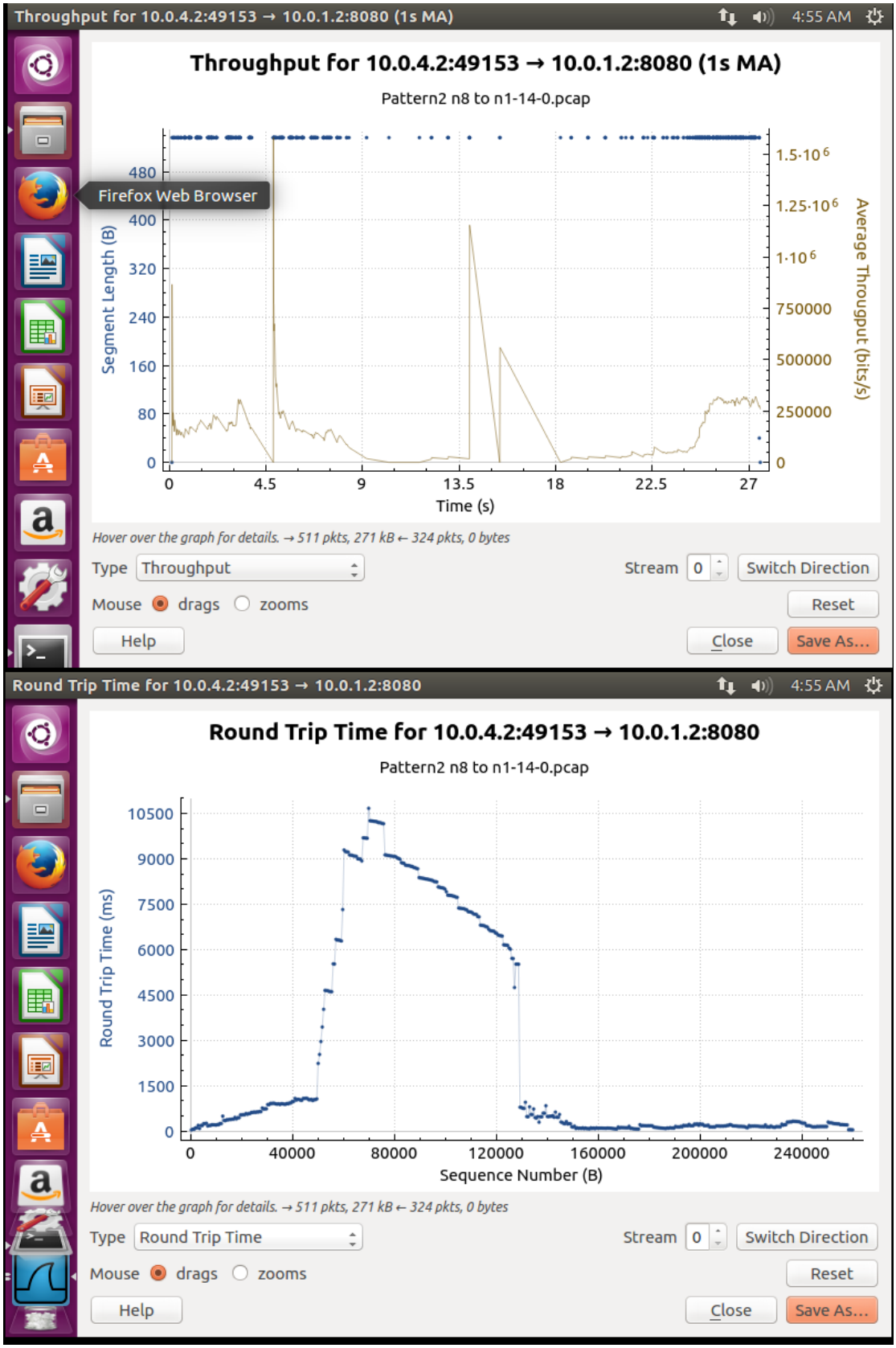

Round Trip Time for 10.0.3.2:49153 → 10.0.1.2:8080
4:52 AM
Round Trip Time for 10.0.3.2:49153 → 10.0.1.2:8080
Pattern2 n5 to n1-11-0.pcap
Round Trip Time (ms)
10500
9000
7500
6000
4500
3000
1500
0
0
40000
80000
120000
160000
200000
240000
280000
320000
Sequence Number (B)
Hover over the graph for details. → 671 pkts, 357 kB ← 418 pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...


Throughput for 10.0.3.2:49153 → 10.0.1.2:8080 (1s MA)
Pattern2 n6 to n1-12-0.pcap
Segment Length (B)
Average Throughput (bits/s)
Time (s)
Hover over the graph for details. → 671 pkts, 357 kB ← 418 pkts, 0 bytes
Type Throughput
Stream 0
Switch Direction
Mouse drags zooms
Reset
Help
Close
Save As...


Round Trip Time for 10.0.3.2:49153 → 10.0.1.2:8080
Round Trip Time for 10.0.3.2:49153 → 10.0.1.2:8080
Pattern2 n6 to n1-12-0.pcap
Round Trip Time (ms)
Sequence Number (B)
Hover over the graph for details. → 671 pkts, 357 kB ← 418 pkts, 0 bytes
Type Round Trip Time
Stream 0
Switch Direction
Mouse drags zooms
Reset
Help
Close
Save As...


Throughput for 10.0.4.2:49153 → 10.0.1.2:8080 (1s MA)
4:54 AM
Throughput for 10.0.4.2:49153 → 10.0.1.2:8080 (1s MA)
Pattern2 n7 to n1-13-0.pcap
LibreOffice Writer
Segment Len (B)
Average Throughput (bits/s)
Time (s)
Hover over the graph for details. → 511 pkts, 271 kB ← 324 pkts, 0 bytes
Type Throughput
Stream 0
Switch Direction
Mouse drags zooms
Reset
Help
Close
Save As...


Round Trip Time for 10.0.4.2:49153 → 10.0.1.2:8080
4:54 AM
Round Trip Time for 10.0.4.2:49153 → 10.0.1.2:8080
Pattern2 n7 to n1-13-0.pcap
Round Trip Time (ms)
Sequence Number (B)
Hover over the graph for details. → 511 pkts, 271 kB ← 324 pkts, 0 bytes
Type Round Trip Time
Stream 0
Switch Direction
Mouse drags zooms
Reset
Help
Close
Save As...


Throughput for 10.0.4.2:49153 → 10.0.1.2:8080 (1s MA)
4:55 AM
Throughput for 10.0.4.2:49153 → 10.0.1.2:8080 (1s MA)
Pattern2 n8 to n1-14-0.pcap
Firefox Web Browser
Segment Length (B)
Average Throughput (bits/s)
Time (s)
Hover over the graph for details. → 511 pkts, 271 kB ← 324 pkts, 0 bytes
Type Throughput
Stream 0
Switch Direction
Mouse drags zooms
Reset
Help
Close
Save As...
Round Trip Time for 10.0.4.2:49153 → 10.0.1.2:8080
4:55 AM
Round Trip Time for 10.0.4.2:49153 → 10.0.1.2:8080
Pattern2 n8 to n1-14-0.pcap
Round Trip Time (ms)
Sequence Number (B)
Hover over the graph for details. → 511 pkts, 271 kB ← 324 pkts, 0 bytes
Type Round Trip Time
Stream 0
Switch Direction
Mouse drags zooms
Reset
Help
Close
Save As...

Pattern2 分析

1. 从数据包流向可以看出，数据包是按照先前预想的进行发送和接收，所以数据包流向是正常的。

2. 从 TCP throughput 来分析，可以看出四条链路的吞吐量的变化情况较大，且该处吞吐量已经接近链路的物理上限，但是从观察可以看出四条链路的平均带宽差不多。 说明，四条链路之间是存在竞争的，并且该竞争较为公平。

五．实验总结

在本次实验中，学习使用在 Linux 下使用 ns-3 仿真模拟软件，取得了较好的效果，对于现有数据中心网络拓扑有了较深刻的体会。