



Report on
Project Phase 1: Enable Queueing QoS and Traffic Shaping
and Policing using Discrete Event Simulator OMNeT++ and
INET Framework

Submitted to
Dr. A. Agarwal
Real-time Multimedia Communication over Internet (ELEC6181)
Department of Electrical and Computer Engineering
Gina Cody School of Engineering and Computer Science

By
Pranav Kumar Jha
Student ID: 40081750

Abstract
In this report, different queueing methods have been used to enhance the network performance for time sensitive applications, also traffic shaping and policing have been applied to enhance the Quality of Service (QoS).

4th February, 2022

Contents

List of Figures	2
1 Experiment 1: Enable Queueing QoS	5
1.1 Introduction	5
1.1.1 Network Topology	5
1.1.2 General Parameters and Applications	6
1.1.3 Different Queueing Methods	8
1.2 Results: Best Effort (DropTail) Queueing Method	9
1.2.1 Results for App 0 at Host H2	9
1.2.2 Results for App 1 at Host H2	10
1.2.3 Results for App 2 at Host H2	11
1.3 Results: Weighted Round Robin (DiffServ) Queueing Method	13
1.3.1 Results for App 0 at Host H2	13
1.3.2 Results for App 1 at Host H2	14
1.3.3 Results for App 2 at Host H2	16
2 Experiment 2: Traffic shaping and policing	18
2.1 Introduction	18
2.2 Network Topology	18
2.2.1 General Parameters and Applications	19
2.2.2 Shaping and Policing	21
2.3 Results: Best effort (DropTail) queueing without policing and shaping	23
2.3.1 Results for App 0 at Host H2	23
2.3.2 Results for App 1 at Host H2	24
2.3.3 Results for App 2 at Host H2	25
2.4 Results: Best effort (DropTail) queueing with policing and shaping	26
2.4.1 Results for App 0 at Host H2	26
2.4.2 Results for App 1 at Host H2	28
2.4.3 Results for App 2 at Host H2	29
2.5 Results: WRR queueing with policing and shaping	30
2.5.1 Results for App 0 at Host H2	30
2.5.2 Results for App 1 at Host H2	32
2.5.3 Results for App 2 at Host H2	33
3 Discussion	35
3.1 Comparison of Queueing Strategies in Experiment 1: Enable Queueing QoS . . .	35

3.1.1	App 0 at Host H2	35
3.1.2	App 1 at Host H2	35
3.1.3	App 2 at Host H2	36
3.2	Comparison of Queueing Strategies in Experiment 2: Traffic shaping and policing	36
3.2.1	App 0 at Host H2	36
3.2.2	App 1 at Host H2	37
3.2.3	App 2 at Host H2	38
4	Conclusion	40
	References	41

Objectives

The objective of this report is

1. To use weighted queueing at the routers and access the performance of the network.
 - This activity is done and reported in Section 1.
2. To exercise router queueing with and without the traffic shaping and policing strategy for different queueing methods.
 - This activity is done and reported in Section 2.
3. To measure the changes in the performance in the above mentioned scenarios.
 - The results have been discussed in detail in Section 3.
4. To conclude the activities along with the technical facts that have been learnt.
 - This is presented in Section 4 which concludes the report with a summary of the results.

List of Figures

1.1	Network topology with best effort	5
1.2	Source package.ned file contents	6
1.3	Configuration of general parameters and applications in omnetpp.ini file	7
1.4	Configuration of network parameters and statistical measurements	8
1.5	Weighted round robin (Diffserv) and best effort (DropTail) queueing	8
1.6	Configuration of filters.xml and OSPFConfig.xml files in omnetpp.ini	9
1.7	End to end delays for all 870 packets of app 0 received at host H2	9
1.8	Received out of order packets vector for app 0 at host H2	10
1.9	Received packets vector for app 0 at host H2	10
1.10	End to end delays for all 4237 packets of app 1 received at host H2	10
1.11	Received packets vector for app 1 at host H2	11
1.12	Sequence number vector of received packets for app 1 at host H2	11
1.13	Throughput vector for app 1 at host H2	11
1.14	End to end delay vector for app 2 received at host H2	12
1.15	Received packets vector for app 2 at host H2	12
1.16	Sent packets vector for app 2 at host H2	12
1.17	End to end delays for all 742 packets of app 0 received at host H2	13
1.18	Packets out of order received vector for app 0 at host H2	13
1.19	Received packets for app 0 at host H2	14
1.20	End to end delays for all 4163 packets of app 1 received at host H2	14
1.21	Received packets vector for app 1 at host H2	15
1.22	Sequence number vector for received packets for app 1 at host H2	15
1.23	Throughput vector for app 1 at host H2	15
1.24	End to end delay vector for app 2 received at host H2	16
1.25	Received packets vector for app 2 at host H2	16
1.26	Packets sent for app 2 at host H2	17
2.1	Network Topology	18

2.2	Source package.ned file contents	19
2.3	Configuration of general parameters and applications in omnetpp.ini file	20
2.4	Configuration of network parameters and statistical measurements	21
2.5	Configuring shaping on Router R2	21
2.6	Configuring policing on Router R4	22
2.7	Best effort DropTail queueing without policing and shaping	22
2.8	Best effort DropTail queueing with policing and shaping	22
2.9	Weighted round robin with traffic policing and shaping	22
2.10	End to end delays for all 792 packets of app 0 received at host H2	23
2.11	Packets Received Out of Order	23
2.12	Received packets vector for app 0 at host H2	23
2.13	Packets Sent	24
2.14	End to end delays for all 4283 packets of app 1 received at host H2	24
2.15	Received packets vector for app 1 at host H2	24
2.16	Sequence number vector of received packets for app 1 at host H2	25
2.17	Throughput vector for app 1 at host H2	25
2.18	End to end delay vector for app 2 received at host H2	25
2.19	Received packets vector for app 2 at host H2	26
2.20	Sent packets vector for app 2 at host H2	26
2.21	End to end delays for all 1246 packets of app 0 received at host H2	27
2.22	Received out of order packets vector for app 0 at host H2	27
2.23	Received packets vector for app 0 at host H2	27
2.24	Packets Sent	28
2.25	End to end delays for all 468 packets of app 1 received at host H2	28
2.26	Received packets vector for app 1 at host H2	28
2.27	Sequence number vector of received packets for app 1 at host H2	29
2.28	Throughput vector for app 1 at host H2	29
2.29	End to end delay vector for app 2 received at host H2	29
2.30	Received packets vector for app 2 at host H2	30

2.31	Sent packets vector for app 2 at host H2	30
2.32	End to end delays for all 487 packets of app 0 received at host H2	31
2.33	Received out of order packets vector for app 0 at host H2	31
2.34	Received packets vector for app 0 at host H2	31
2.35	Packets Sent	32
2.36	End to end delays for all 795 packets of app 1 received at host H2	32
2.37	Received packets vector for app 1 at host H2	32
2.38	Sequence number vector of received packets for app 1 at host H2	33
2.39	Throughput vector for app 1 at host H2	33
2.40	End to end delay vector for app 2 received at host H2	33
2.41	Received packets vector for app 2 at host H2	34
2.42	Sent packets vector for app 2 at host H2	34

1 Experiment 1: Enable Queueing QoS

1.1 Introduction

In this experiment, I have used the weighted Queueing at routers to enhance the performance of the network for the applications which are time sensitive. I have also provided the comparison of the performance and quality of network services measured before and after applying quality of service.

1.1.1 Network Topology

The network with best effort performance is designed and shown in the Fig. 1.1. The performance is measured and discussed in further sections.

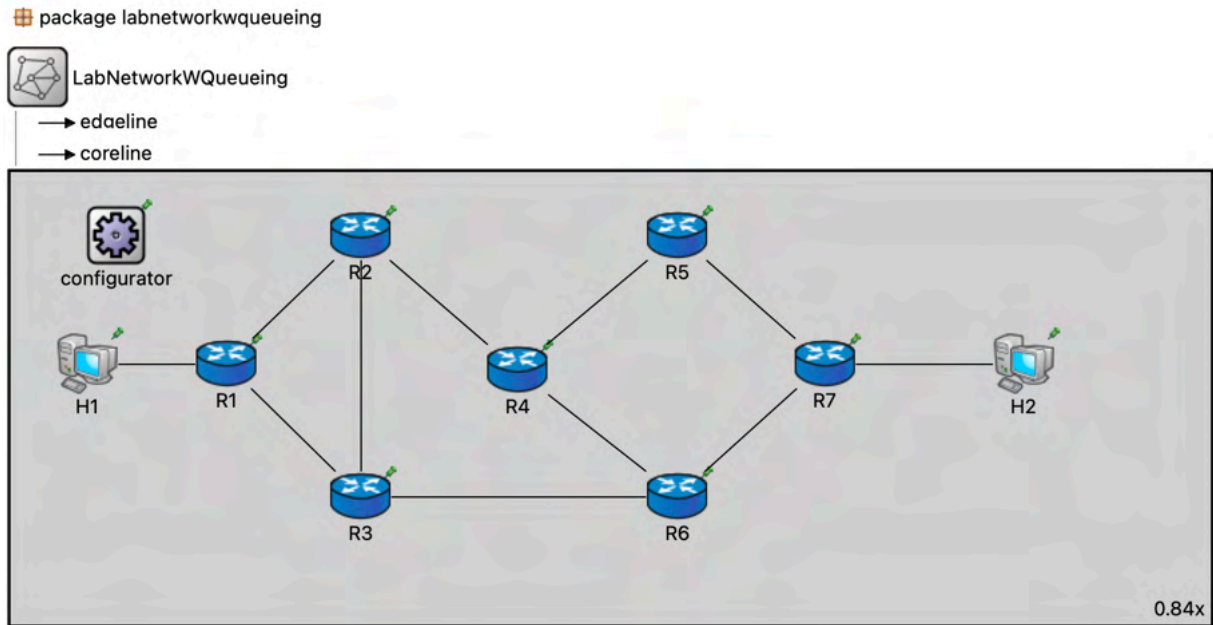
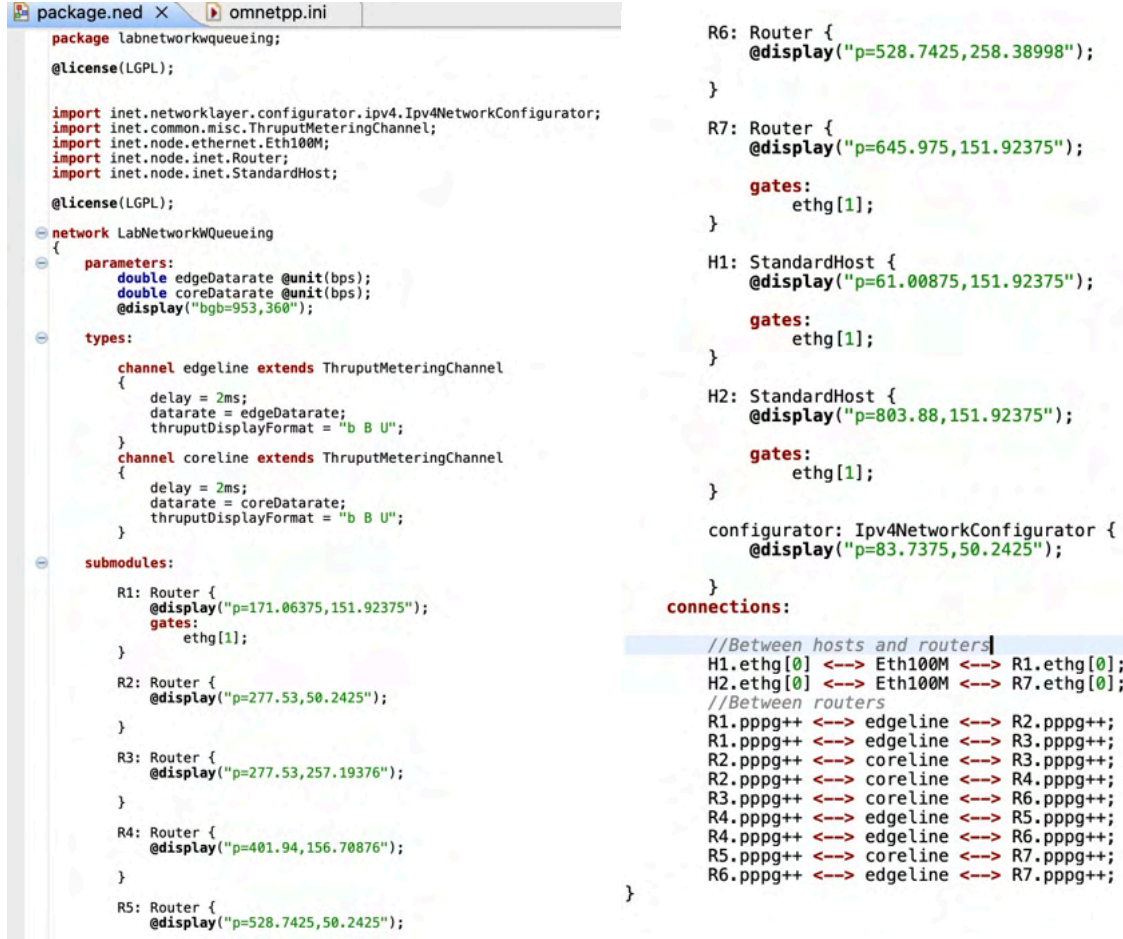


Figure 1.1: Network topology with best effort

The network in Fig. 1.1 is build in OMNet++ with INET framework. H1 and H2 are the standard Ipv4 hosts with Stream Control Transmission Protocol (SCTP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) layers and applications. R1-R7 are the routers where R1 and R7 are connected to the sender H1 and receiver H2 with ethernet 100M EtherFrame-connections, respectively. The routers are connected among themselves via edgeline

and coreline communication channels. The edgeline and coreline communication channels are having the delay of 2ms, the edgeDatarate is set to 32 kbps and the coreDatarate is set to 16 kbps. The network topology used in the package.ned source file is provided below in Fig. 1.2 [1]



```

package labNetworkQueueing;
@license(LGPL);

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.common.misc.ThroughputMeteringChannel;
import inet.node.ethernet.Eth100M;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;
@license(LGPL);

network LabNetworkQueueing
{
    parameters:
        double edgeDatarate @unit(bps);
        double coreDatarate @unit(bps);
        @display("bgb=953,360");

    types:
        channel edgeline extends ThroughputMeteringChannel
        {
            delay = 2ms;
            datarate = edgeDatarate;
            throughputDisplayFormat = "b B U";
        }
        channel coreline extends ThroughputMeteringChannel
        {
            delay = 2ms;
            datarate = coreDatarate;
            throughputDisplayFormat = "b B U";
        }

    submodules:
        R1: Router {
            @display("p=171.06375,151.92375");
            gates:
                ethg[1];
        }
        R2: Router {
            @display("p=277.53,50.2425");
        }
        R3: Router {
            @display("p=277.53,257.19376");
        }
        R4: Router {
            @display("p=401.94,156.70876");
        }
        R5: Router {
            @display("p=528.7425,50.2425");
        }
        R6: Router {
            @display("p=528.7425,258.38998");
        }
        R7: Router {
            @display("p=645.975,151.92375");
            gates:
                ethg[1];
        }
        H1: StandardHost {
            @display("p=61.00875,151.92375");
            gates:
                ethg[1];
        }
        H2: StandardHost {
            @display("p=803.88,151.92375");
            gates:
                ethg[1];
        }
        configurator: Ipv4NetworkConfigurator {
            @display("p=83.7375,50.2425");
        }

    connections:
        //Between hosts and routers
        H1.ethg[0] <==> Eth100M <==> R1.ethg[0];
        H2.ethg[0] <==> Eth100M <==> R7.ethg[0];
        //Between routers
        R1.pppg++ <==> edgeline <==> R2.pppg++;
        R1.pppg++ <==> edgeline <==> R3.pppg++;
        R2.pppg++ <==> coreline <==> R3.pppg++;
        R2.pppg++ <==> coreline <==> R4.pppg++;
        R3.pppg++ <==> coreline <==> R6.pppg++;
        R4.pppg++ <==> edgeline <==> R5.pppg++;
        R4.pppg++ <==> edgeline <==> R6.pppg++;
        R5.pppg++ <==> coreline <==> R7.pppg++;
        R6.pppg++ <==> edgeline <==> R7.pppg++;
}

```

Figure 1.2: Source package.ned file contents

1.1.2 General Parameters and Applications

The general parameters in the experiment used for queueing and Open Shortest Path First (OSPF) in the omnetpp.ini file is presented below in Fig. 1.3(a) and to run, UdpBasicBurst (voice streaming), UdpBasicApp (video streaming) and TcpBasicClientApp (web streaming) are used as shown in the Fig. 1.3(b) [1].

```

[General]

network = LabNetworkQueueing

sim-time-limit = 1250s
**.result-recording-modes = all
**.scalar-recording = true

# default queues
**.queue.typename = "EtherQosQueue"
**.queue.dataQueue.typename = "DropTailQueue"
**.queue.packetCapacity = 100
**.queue.dataQueue.packetCapacity = 100

# Enable OSPF on all routers
**.R*.hasOspf = true
**.R*.ospf.ospfConfig = xmldoc("OSPFConfig.xml")

```

(a) General parameters

```

(Config Apps)
**.H7.numApps = 3

# first app: voice streaming
**.H1.app[0].typename = "UdpBasicBurst"
**.H1.app[0].destAddresses = "H2"
**.H1.app[0].chooseDestAddrMode = "once"
**.H1.app[0].destPort = 1000
**.H1.app[0].startTime = uniform(1s,2s)
**.H1.app[0].stopTime = 1200s
**.H1.app[0].messageLength = 172B # 160B voice + 12B Rtp header
**.H1.app[0].burstDuration = exponential(0.352s)
**.H1.app[0].sleepDuration = exponential(0.650s)
**.H1.app[0].sendInterval = 20ms

**.H2.app[0].typename = "UdpBasicBurst"
**.H2.app[0].localPort = 1000
**.H2.app[0].delayLimit = 0ms
**.H2.app[0].destAddresses = ""
**.H2.app[0].chooseDestAddrMode = "once"
**.H2.app[0].destPort = 0
**.H2.app[0].messageLength = 0B
**.H2.app[0].burstDuration = 0s
**.H2.app[0].sleepDuration = 0s
**.H2.app[0].sendInterval = 0ms

#second app: video
**.H1.app[1].typename = "UdpBasicApp"
**.H1.app[1].destPort = 2000
**.H1.app[1].startTime = uniform(1s,2s)
**.H1.app[1].stopTime = 1200s
**.H1.app[1].sendInterval = 40ms
**.H1.app[1].messageLength = 500B
**.H1.app[1].destAddresses = "H2"

**.H2.app[1].typename = "UdpSink" #"UdpEchoApp"
**.H2.app[1].localPort = 2000

#third app: tcp traffic
**.H1.app[2].typename = "TcpBasicClientApp"
**.H1.app[2].connectAddress = "H2"
**.H1.app[2].connectPort = 3000
**.H1.app[2].numRequestsPerSession = 1 # HTTP 1.0
**.H1.app[2].requestLength = 50B
**.H1.app[2].replyLength = 100B
**.H1.app[2].thinkTime = 40ms
**.H1.app[2].idleInterval = 10s
**.H1.app[2].stopTime = 1200s

**.H2.app[2].typename = "TcpGenericServerApp"
**.H2.app[2].localPort = 3000

```

(b) Applications

Figure 1.3: Configuration of general parameters and applications in omnetpp.ini file

Further, to configure the network with different link configurations, the edgelines and the corelines have been used between the routers for better security, reliable and to serve packets as fast as possible [2]. Also, the traffic classifier and markers TC1 are used at R1 and R7 basically to allow Quality of Service (QoS) enabled networks to identify different types of traffic at the routers R1 and R7 as shown in the Fig. 1.4, and associate those traffic types with specific markings [3].

```

[Config Exp]
**.edgeDataRate = 32kbps
**.coreDataRate = 16kbps
**.R1.eth[*].ingressTC.typeName = "TC1"
**.R7.eth[*].ingressTC.typeName = "TC1"

**.ingressTC.numClasses = 3
**.ingressTC.classifier.filters = xmldoc("filters.xml", "//experiment[@id='default']")
**.ingressTC.marker.dscps = "AF11 AF21 AF31 AF41 BE"

# statistics
**.H?.app[*].sentPk.result-recording-modes = count
**.H?.app[*].rcvdPk.result-recording-modes = count
**.H?.app[*].dropPk.result-recording-modes = vector
**.H?.app[*].endToEndDelay.result-recording-modes = vector # for computing median

**.R?.ppp[*].**Queue.rcvdPk.result-recording-modes = count
**.R?.ppp[*].**Queue.dropPk.result-recording-modes = count
**.R?.ppp[*].**Queue.queueLength.result-recording-modes = timeavg
**.R?.ppp[*].**Queue.queueingTime.result-recording-modes = vector # for computing median

**.R?.ppp[*].**Queue.*.scalar-recording = true

**.app[*].sentPk.scalar-recording = true
**.app[*].rcvdPk.scalar-recording = true
**.app[*].endToEndDelay.scalar-recording = true

**.afQueue.*.scalar-recording = true

```

Figure 1.4: Configuration of network parameters and statistical measurements

1.1.3 Different Queueing Methods

As shown in Fig. 1.5, weighted round robin (WRR) and best effort queueing methods are used and the received packets, dropped packets, queue length and queueing time are being measured in both of the above mentioned scenarios.

```

[Config Exp1DiffServ] # with Diffserv Queue
extends = Apps, Exp
**.R?.ppp[*].ppp.queue.typeName = "DSQueue1" #Diffserv Queue
**.R?.ppp[*].ppp.queue.packetCapacity = 100 #-1
**.R?.ppp[*].ppp.queue.*.packetCapacity = 100
**.R?.ppp[*].ppp.queue.wrr.weights = "10 40 5 1 1"

[Config Exp2DropTail] # with DropTailQueue
extends = Apps, Exp
**.ppp[*].ppp.queue.typeName = "DropTailQueue"
**.eth[*].mac.queue.typeName = "EtherQosQueue"
**.eth[*].mac.queue.dataQueue.typeName = "DropTailQueue"
**.queue.packetCapacity = 100

```

Figure 1.5: Weighted round robin (Diffserv) and best effort (DropTail) queueing

The xml files used for OSPF configuration and traffic classification are included in the project as filters.xml and OSPFConfig.xml and the contents in those files are shown below in Fig. 1.6(a) and Fig. 1.6(b), respectively.

```

package.ned  omnetpp.ini  filters.xml  OSPFConfig.xml
<filters>
<experiment id="default">
<filter srcAddress="H1" destPort="1000" gate="0"/>
<filter srcAddress="H1" destPort="2000" gate="1"/>
<filter srcAddress="H1" destPort="3000" gate="2"/>
<filter srcAddress="H2" gate="0"/>
</experiment>
</filters>

```

(a) filters.xml file

```

package.ned  omnetpp.ini  filters.xml  OSPFConfig.xml
<OSPFASConfig>
<Area id="0.0.0.0">
<AddressRange address="H1" mask="H1" />
<AddressRange address="H2" mask="H2" />

<AddressRange address="R1>R2" mask="R1>R2" />
<AddressRange address="R2>R1" mask="R2>R1" />

<AddressRange address="R1>R3" mask="R1>R3" />
<AddressRange address="R3>R1" mask="R3>R1" />

<AddressRange address="R2>R3" mask="R2>R3" />
<AddressRange address="R3>R2" mask="R3>R2" />

<AddressRange address="R2>R4" mask="R2>R4" />
<AddressRange address="R4>R2" mask="R4>R2" />

<AddressRange address="R3>R6" mask="R3>R6" />
<AddressRange address="R6>R3" mask="R6>R3" />

<AddressRange address="R4>R5" mask="R4>R5" />
<AddressRange address="R5>R4" mask="R5>R4" />

<AddressRange address="R4>R6" mask="R4>R6" />
<AddressRange address="R6>R4" mask="R6>R4" />

<AddressRange address="R5>R7" mask="R5>R7" />
<AddressRange address="R7>R5" mask="R7>R5" />

<AddressRange address="R6>R7" mask="R6>R7" />
<AddressRange address="R7>R6" mask="R7>R6" />
</Area>
<Router name="**" RFC1583Compatible="true">
<BroadcastInterface ifName="eth[*]" areaID="0.0.0.0" interfaceOutputCost="0" />
<PointToPointInterface ifName="ppp[*]" areaID="0.0.0.0"
interfaceOutputCost="0" />
</Router>
</OSPFASConfig>

```

(b) OSPFConfig.xml file

Figure 1.6: Configuration of filters.xml and OSPFConfig.xml files in omnetpp.ini

1.2 Results: Best Effort (DropTail) Queueing Method

1.2.1 Results for App 0 at Host H2

End to End Delay

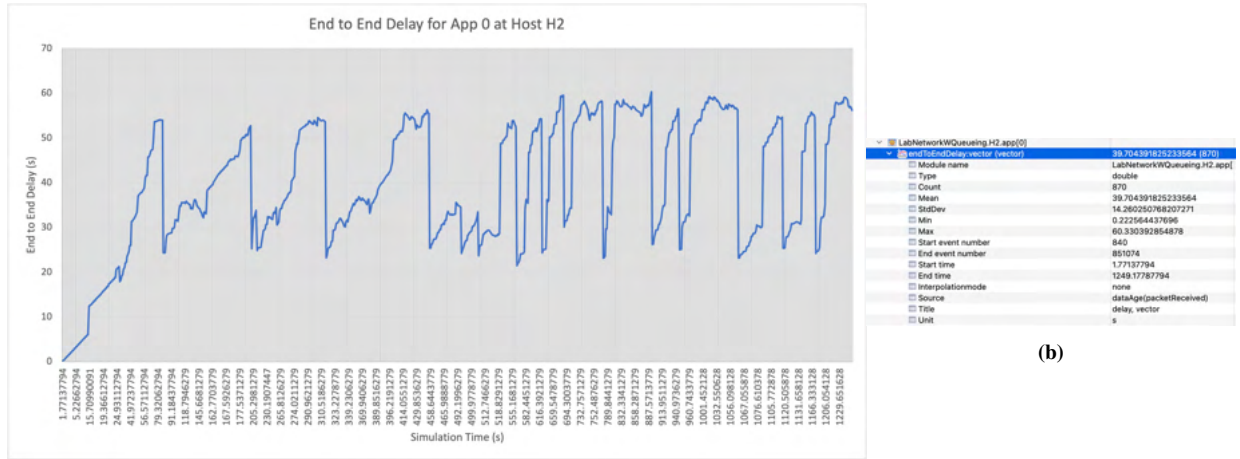


Figure 1.7: End to end delays for all 870 packets of app 0 received at host H2

Received Out of Order Packets

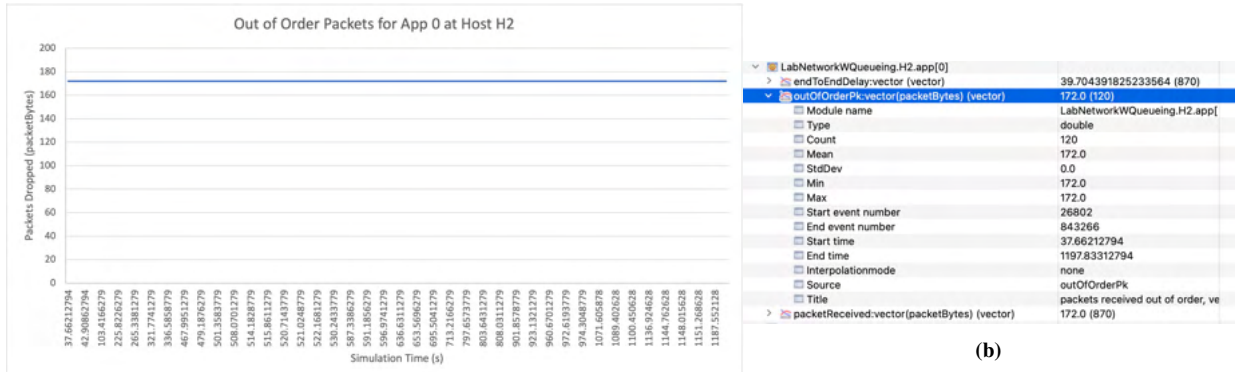


Figure 1.8: Received out of order packets vector for app 0 at host H2

Received Packets

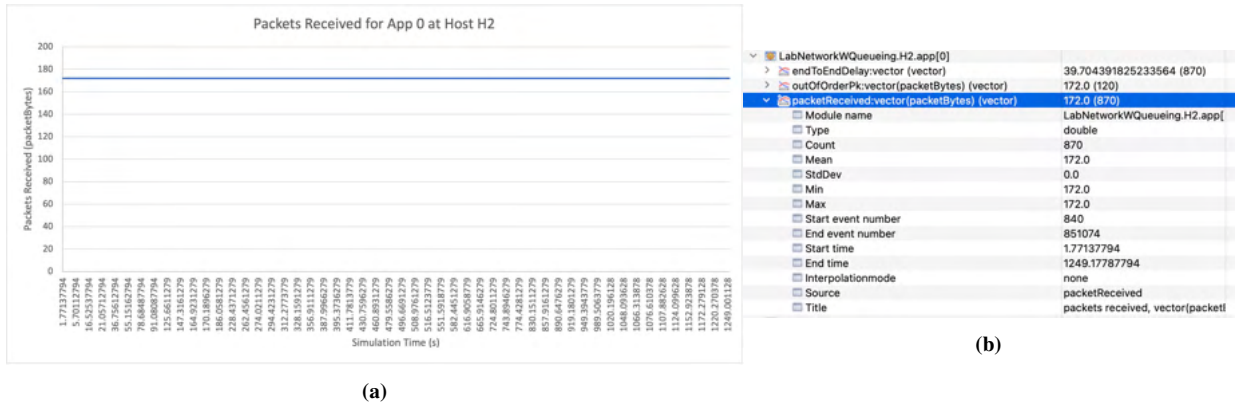


Figure 1.9: Received packets vector for app 0 at host H2

1.2.2 Results for App 1 at Host H2

End to End Delay

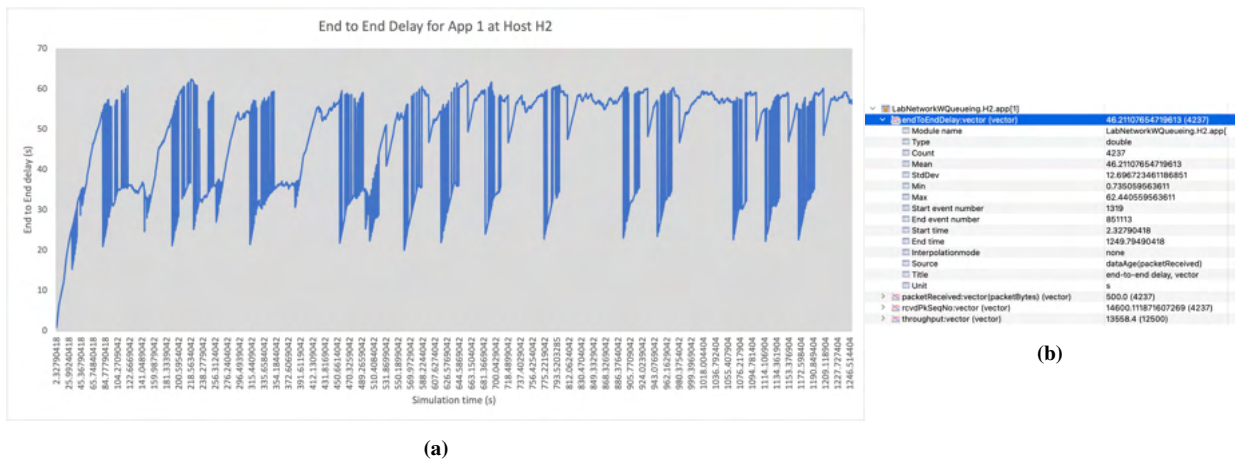


Figure 1.10: End to end delays for all 4237 packets of app 1 received at host H2

Received Packets

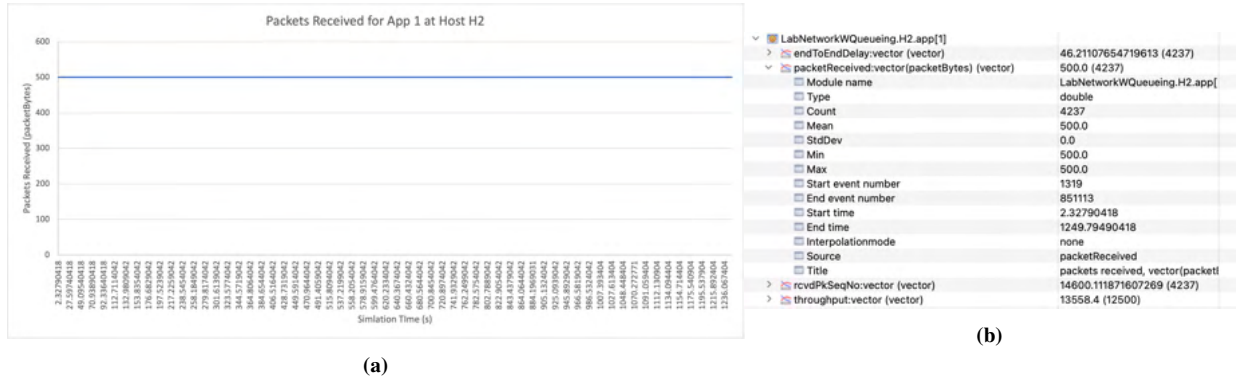


Figure 1.11: Received packets vector for app 1 at host H2

Sequence Number for Received Packets

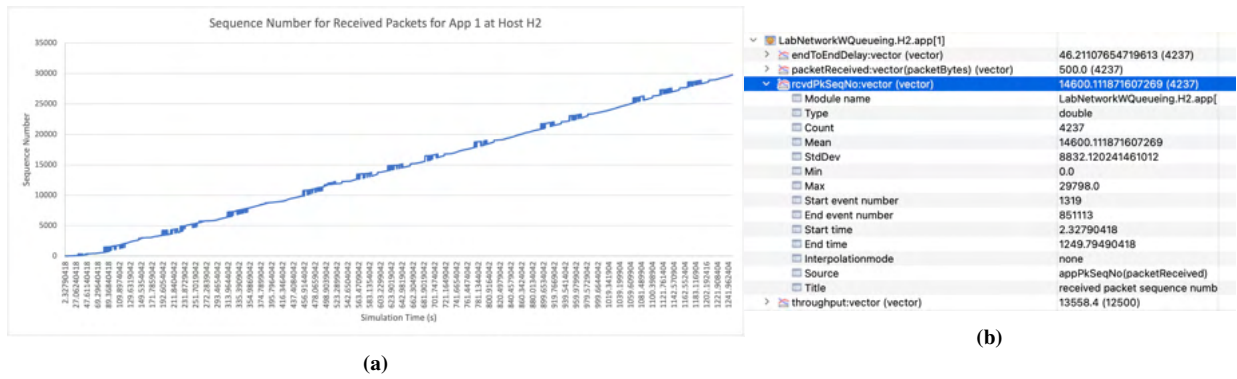


Figure 1.12: Sequence number vector of received packets for app 1 at host H2

Throughput Vector

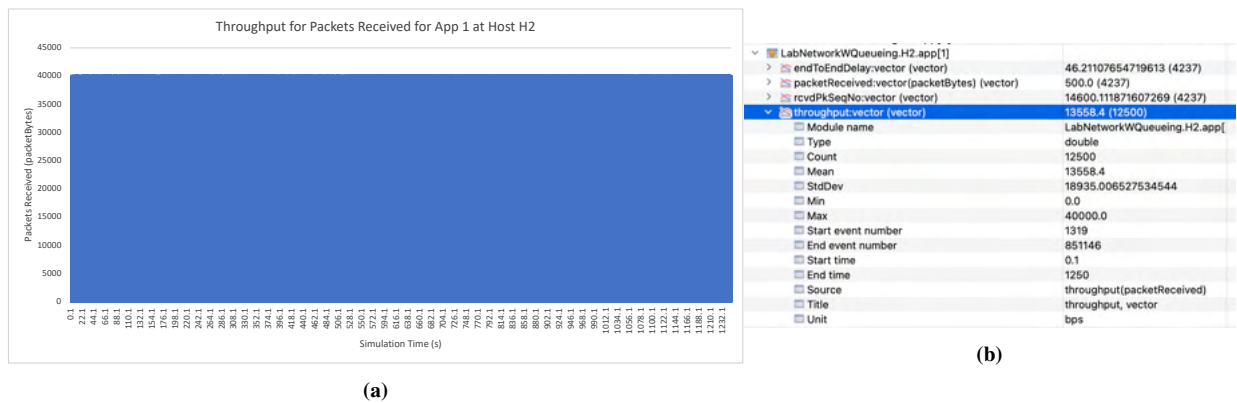
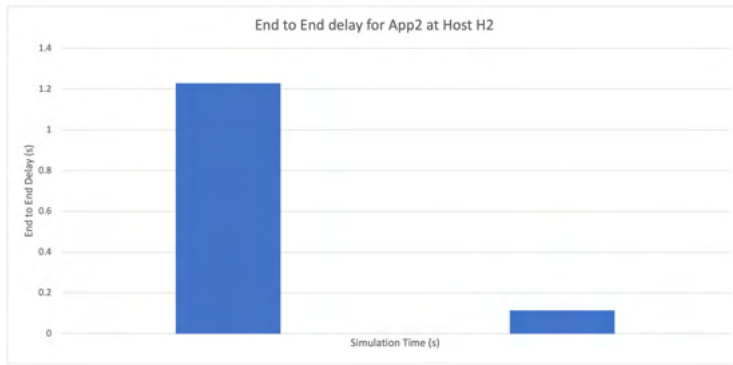


Figure 1.13: Throughput vector for app 1 at host H2

1.2.3 Results for App 2 at Host H2

End to End Delay



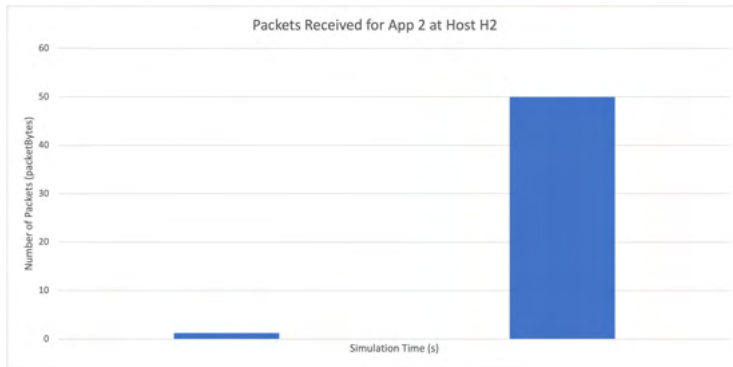
(a)

LabNetworkWQueueing.H2.app[2]	
endToEndDelay:vector (vector)	0.11476514 (1)
Module name	LabNetworkWQueueing.H2.app[2]
Type	double
Count	1
Mean	0.11476514
StdDev	Infinity
Min	0.11476514
Max	0.11476514
Start event number	396
End event number	396
Start time	1.22881162
End time	1.22881162
Interpolationmode	none
Source	dataAge(packetReceived)
Title	end-to-end delay, vector
Unit	s
packetReceived:vector(packetBytes) (vector)	50.0 (1)
packetSent:vector(packetBytes) (vector)	100.0 (1)

(b)

Figure 1.14: End to end delay vector for app 2 received at host H2

Received Packets



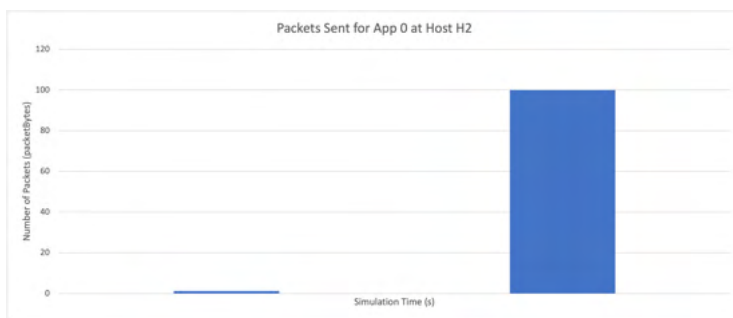
(a)

LabNetworkWQueueing.H2.app[2]	
endToEndDelay:vector (vector)	0.11476514 (1)
packetReceived:vector(packetBytes) (vector)	50.0 (1)
Module name	LabNetworkWQueueing.H2.app[2]
Type	double
Count	1
Mean	50.0
StdDev	NaN
Min	50.0
Max	50.0
Start event number	396
End event number	396
Start time	1.22881162
End time	1.22881162
Interpolationmode	none
Source	packetReceived
Title	packets received, vector(packetBytes)
packetSent:vector(packetBytes) (vector)	100.0 (1)

(b)

Figure 1.15: Received packets vector for app 2 at host H2

Sent Packets



(a)

LabNetworkWQueueing.H2.app[2]	
endToEndDelay:vector (vector)	0.11476514 (1)
packetReceived:vector(packetBytes) (vector)	50.0 (1)
packetSent:vector(packetBytes) (vector)	100.0 (1)
Module name	LabNetworkWQueueing.H2.app[2]
Type	double
Count	1
Mean	100.0
StdDev	NaN
Min	100.0
Max	100.0
Start event number	396
End event number	396
Start time	1.22881162
End time	1.22881162
Interpolationmode	none
Source	packetSent
Title	packets sent, vector(packetBytes)

(b)

Figure 1.16: Sent packets vector for app 2 at host H2

1.3 Results: Weighted Round Robin (DiffServ) Queueing Method

1.3.1 Results for App 0 at Host H2

End to End Delay

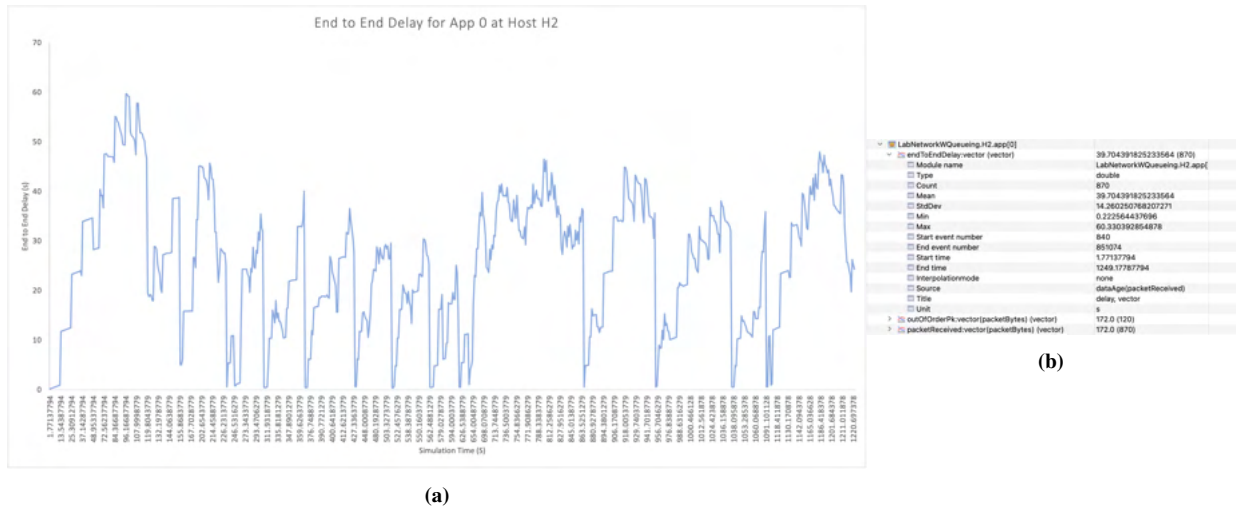


Figure 1.17: End to end delays for all 742 packets of app 0 received at host H2

Received Out of Order Packets

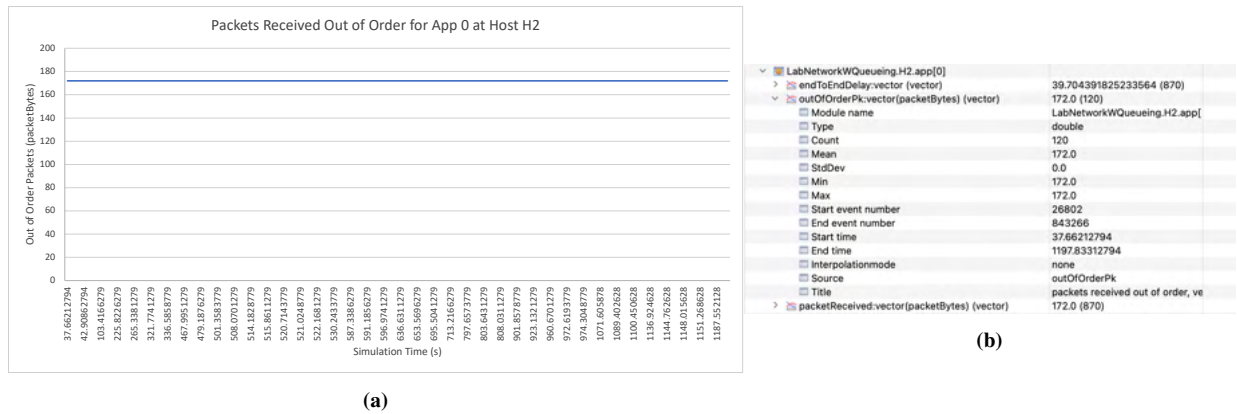


Figure 1.18: Packets out of order received vector for app 0 at host H2

Received Packets

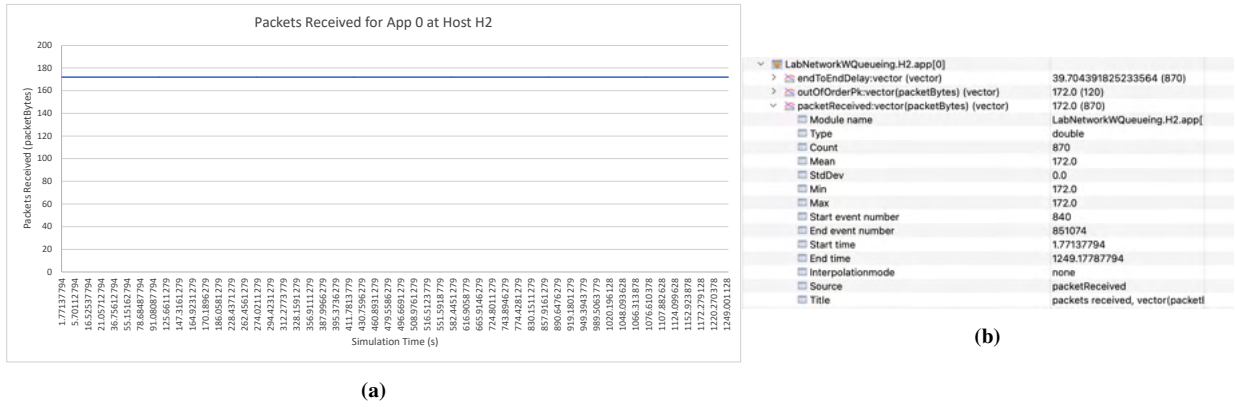


Figure 1.19: Received packets for app 0 at host H2

1.3.2 Results for App 1 at Host H2

End to End Delay

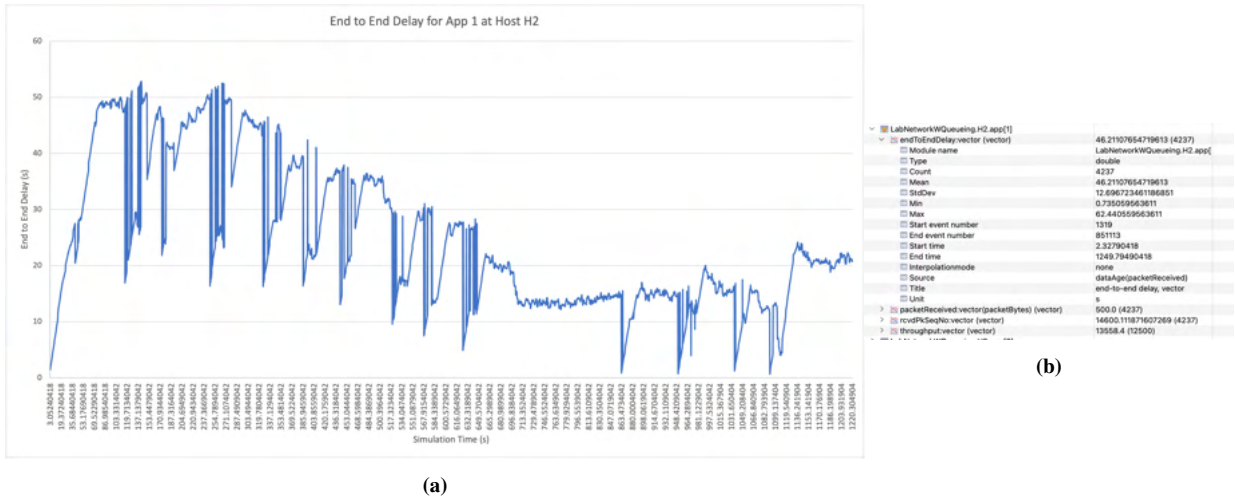
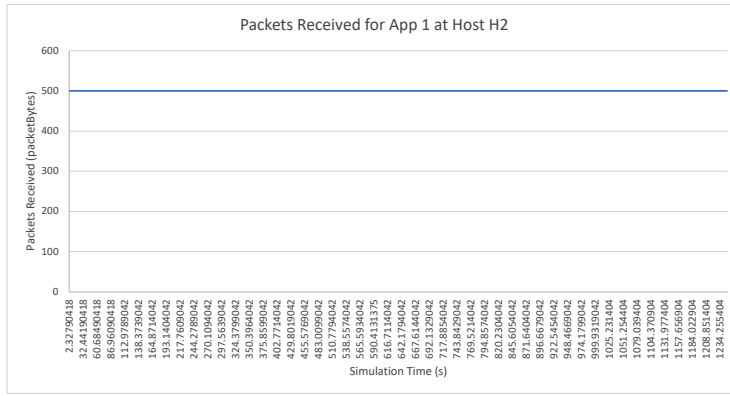


Figure 1.20: End to end delays for all 4163 packets of app 1 received at host H2

Received Packets



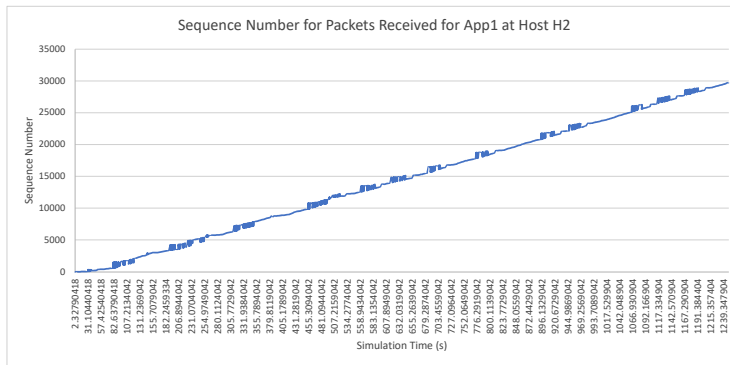
(a)

LabNetworkWQueueing.H2_app[1]	
endToEndDelay:vector (vector)	46.21107654719613 (4237)
packetReceived:vector(packetBytes) (vector)	500.0 (4237)
Module name	LabNetworkWQueueing.H2_app[1]
Type	double
Count	4237
Mean	500.0
StdDev	0.0
Min	500.0
Max	500.0
Start event number	1319
End event number	851113
Start time	2.32790418
End time	1249.79490418
Interpolationmode	none
Source	packetReceived
Title	packets received, vector(packetBytes)
rcvdPkSeqNo:vector (vector)	14600.111871607269 (4237)
throughput:vector (vector)	13558.4 (12500)

(b)

Figure 1.21: Received packets vector for app 1 at host H2

Sequence Number for Received Packets



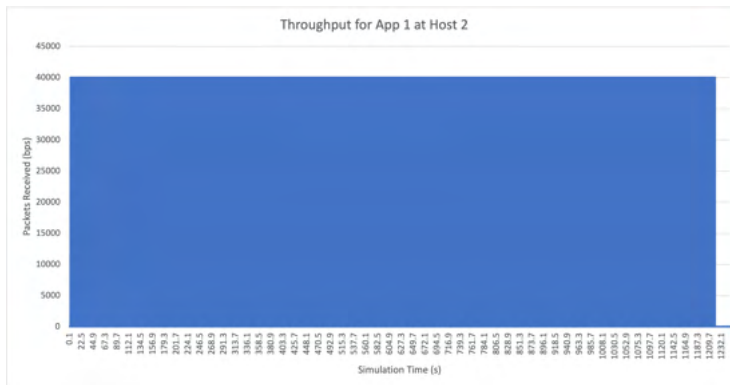
(a)

LabNetworkWQueueing.H2_app[1]	
endToEndDelay:vector (vector)	46.21107654719613 (4237)
packetReceived:vector(packetBytes) (vector)	500.0 (4237)
rcvdPkSeqNo:vector (vector)	14600.111871607269 (4237)
Module name	LabNetworkWQueueing.H2_app[1]
Type	double
Count	4237
Mean	14600.111871607269
StdDev	8832.120241461012
Min	0.0
Max	29798.0
Start event number	1319
End event number	851113
Start time	2.32790418
End time	1249.79490418
Interpolationmode	none
Source	appPkSeqNo(packetReceived)
Title	received packet sequence number
throughput:vector (vector)	13558.4 (12500)

(b)

Figure 1.22: Sequence number vector for received packets for app 1 at host H2

Throughput Vector



(a)

LabNetworkWQueueing.H2_app[1]	
endToEndDelay:vector (vector)	24.481471631205547 (4163)
packetReceived:vector(packetBytes) (vector)	500.0 (4163)
rcvdPkSeqNo:vector (vector)	14581.249099207302 (4163)
throughput:vector (vector)	13321.6 (12500)
Module name	LabNetworkWQueueing.H2_app[1]
Type	double
Count	12500
Mean	13321.6
StdDev	18852.78249046041
Min	0.0
Max	40000.0
Start event number	1698
End event number	839632
Start time	0.1
End time	1250
Source	throughput(packetReceived)
Title	throughput, vector
Unit	bps

(b)

Figure 1.23: Throughput vector for app 1 at host H2

1.3.3 Results for App 2 at Host H2

End to End Delay

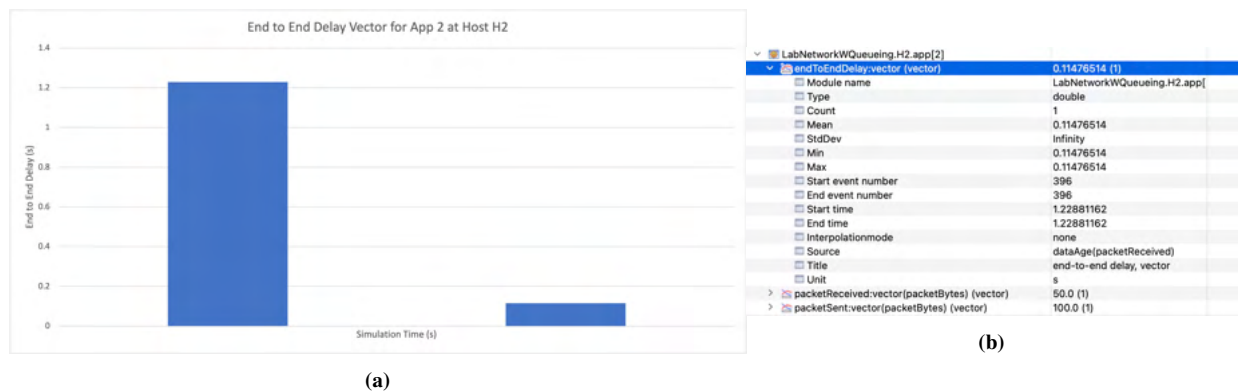


Figure 1.24: End to end delay vector for app 2 received at host H2

Received Packets

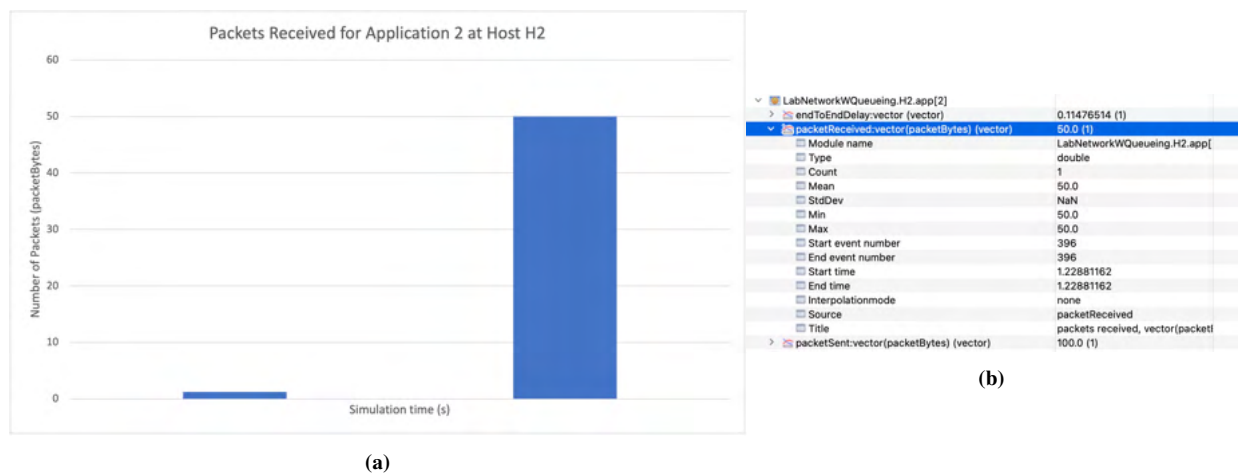


Figure 1.25: Received packets vector for app 2 at host H2

Sent Packets

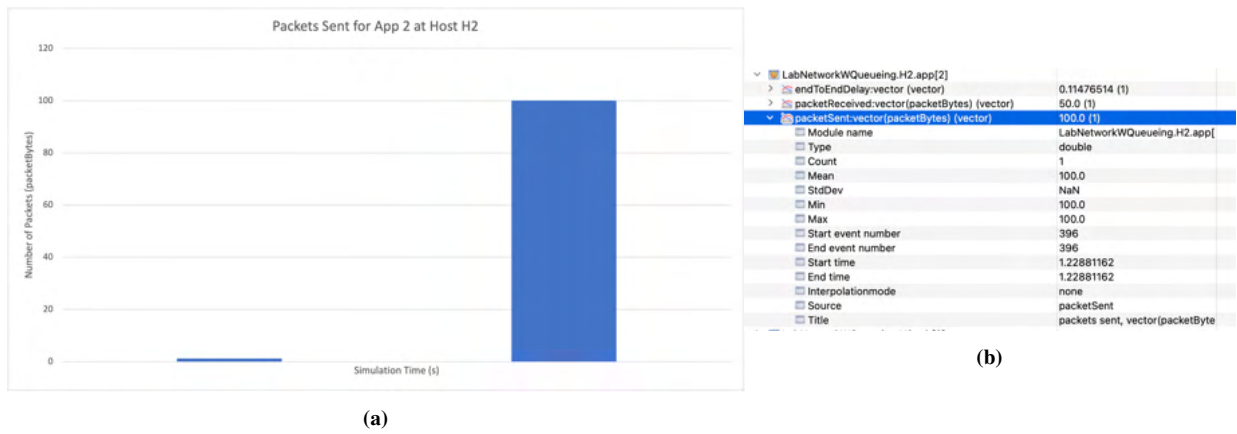


Figure 1.26: Packets sent for app 2 at host H2

2 Experiment 2: Traffic shaping and policing

2.1 Introduction

In this experiment, the focus is to enhance knowledge about traffic shaping and policing. Also, the router queuing method will be exercised while working with shaping and policing in order to enhance QoS.

2.2 Network Topology

The network is designed and shown in the Fig. 2.1. The performance is measured and discussed in the further sections.

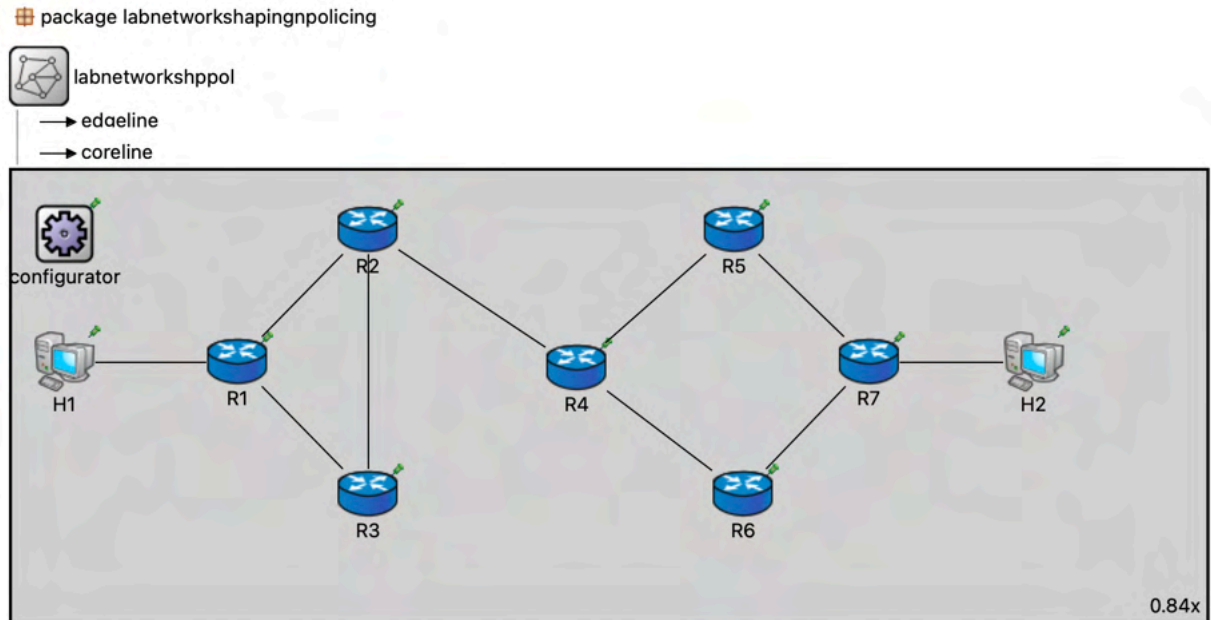


Figure 2.1: Network Topology

The network in Fig. 2.1 is build in OMNet++ with INET framework. H1 and H2 are the standard Ipv4 hosts with SCTP, TCP and UDP layers and applications. R1-R7 are the routers where R1 and R7 are connected to the sender H1 and receiver H2 with ethernet 100M EtherFrame-connections, respectively. The routers are connected among themselves via edgeline and coreline communication channels. The edgeline and coreline communication channels are having the delay of 2ms, the edgeDatarate is set to 32 kbps and the coreDatarate is set to 16 kbps. The network topology used in the package.ned source file is provided below in Fig. 2.2 [1]

```

package labnetworkshapingnopoling;

@license(LGPL);

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.common.misc.ThruputMeteringChannel;
import inet.node.ethernet.Eth100M;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;

@license(LGPL);

network labnetworkshppol
{
    parameters:
        double edgeDatarate @unit(bps);
        double coreDatarate @unit(bps);
        @display("bgb=953,360");

    types:
        channel edgeline extends ThruputMeteringChannel
        {
            delay = 2ms;
            datarate = edgeDatarate;
            thruputDisplayFormat = "b B U";
        }
        channel coreline extends ThruputMeteringChannel
        {
            delay = 2ms;
            datarate = coreDatarate;
            thruputDisplayFormat = "b B U";
        }

    submodules:
        R1: Router {
            @display("p=179.4375,151.92375");
            gates:
                ethg[1];
        }
        R2: Router {
            @display("p=283.51126,46.65375");
        }
        R3: Router {
            @display("p=283.51126,258.38998");
        }
        R4: Router {
            @display("p=449.79,156.70876");
        }
        R5: Router {
            @display("p=576.59247,46.65375");
        }
        R6: Router {
            @display("p=582.5737,257.19376");
        }
        R7: Router {
            @display("p=683.0588,151.92375");
            gates:
                ethg[1];
        }
        H1: StandardHost {
            @display("p=41.86875,151.92375");
            gates:
                ethg[1];
        }
        H2: StandardHost {
            @display("p=814.64624,151.92375");
            gates:
                ethg[1];
        }
        configurator: Ipv4NetworkConfigurator {
            @display("p=42,51");
        }

    connections:
        //Between hosts and routers
        H1.ethg[0] <==> Eth100M <==> R1.ethg[0];
        H2.ethg[0] <==> Eth100M <==> R7.ethg[0];
        //Between routers
        R1.pppg++ <==> edgeline <==> R2.pppg++;
        R1.pppg++ <==> edgeline <==> R3.pppg++;
        R2.pppg++ <==> coreline <==> R3.pppg++;
        R2.pppg++ <==> coreline <==> R4.pppg++;
        R4.pppg++ <==> edgeline <==> R5.pppg++;
        R4.pppg++ <==> edgeline <==> R6.pppg++;
        R5.pppg++ <==> coreline <==> R7.pppg++;
        R6.pppg++ <==> edgeline <==> R7.pppg++;
}

```

Figure 2.2: Source package.ned file contents

2.2.1 General Parameters and Applications

The general parameters in the experiment used for queueing and Open Shortest Path First (OSPF) in the omnetpp.ini file is presented below in Fig. 2.3(a) and to run, UdpBasicBurst (voice streaming), UdpBasicApp (video streaming) and TcpBasicClientApp (web streaming) are used as shown in the Fig. 2.3(b) [1].

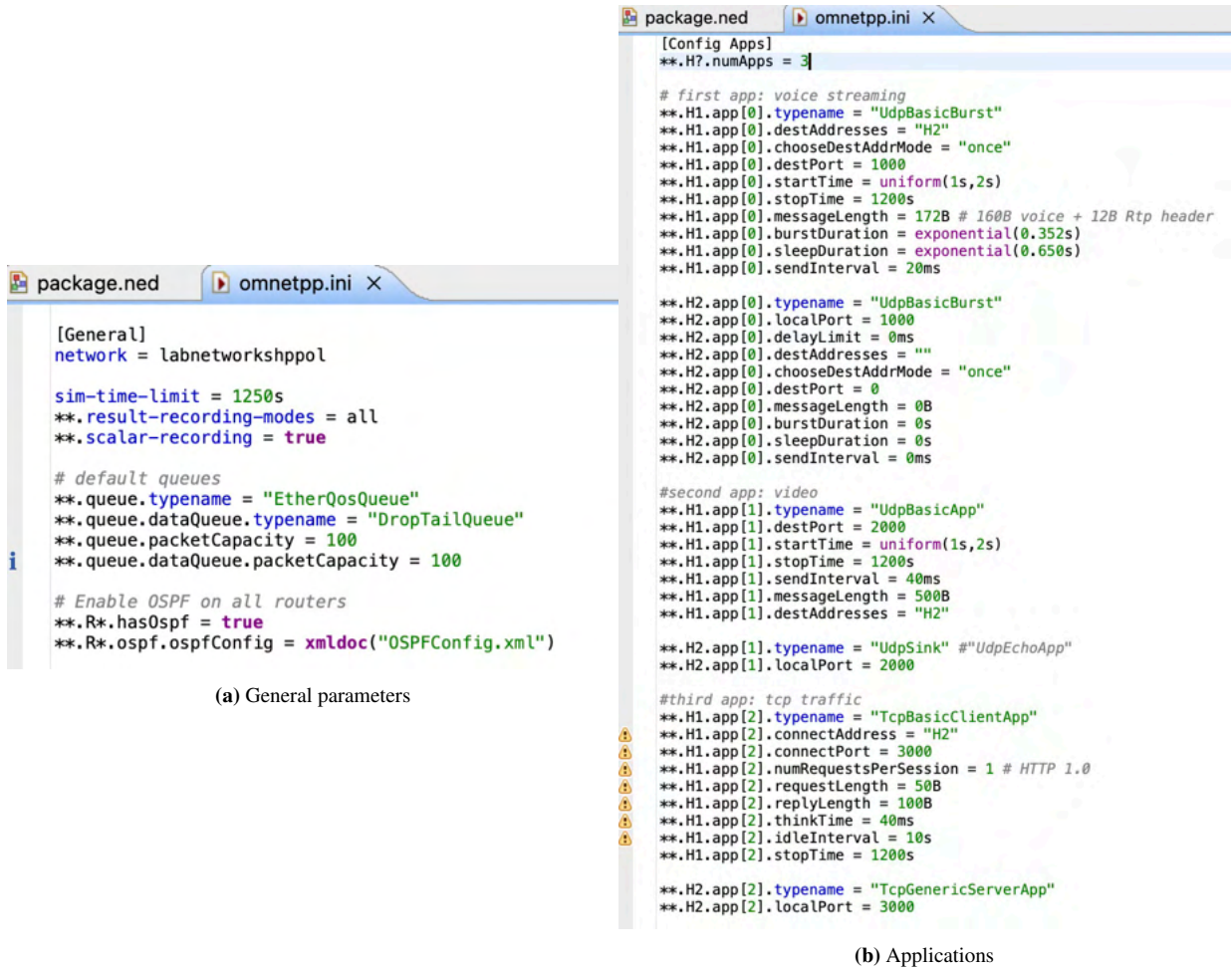


Figure 2.3: Configuration of general parameters and applications in omnetpp.ini file

Further, to configure the network with different link configurations, the edgelines and the corelines have been used between the routers for better security, reliable and to serve packets as fast as possible [2]. Also, the traffic classifier and markers TC1 are used at R1 and R7 basically to allow QoS enabled networks to identify different types of traffic at the routers R1 and R7 as shown in the Fig. 2.4, and associate those traffic types with specific markings [3].


```

package.ned  omnetpp.ini X

[Config Exp]
**.edgeDataRate = 32kbps
**.coreDataRate = 16kbps

#Traffic Classifier and Marker
**.R1.eth[*].ingressTC.typeName = "TC1"
**.R7.eth[*].ingressTC.typeName = "TC1"
**.ingressTC.numClasses = 3
**.ingressTC.classifier.filters = xmldoc("filters.xml", "//experiment[@id='default']")
**.ingressTC.marker.dscps = "AF11 AF21 AF31 AF41 BE"

# statistics
**.H?.app[*].sentPk.result-recording-modes = count
**.H?.app[*].rcvdPk.result-recording-modes = count
**.H?.app[*].dropPk.result-recording-modes = vector
**.H?.app[*].endToEndDelay.result-recording-modes = vector # for computing median

**.R?.ppp[*].**Queue.rcvdPk.result-recording-modes = count
**.R?.ppp[*].**Queue.dropPk.result-recording-modes = count
**.R?.ppp[*].**Queue.queueLength.result-recording-modes = timeavg
**.R?.ppp[*].**Queue.queueingTime.result-recording-modes = vector # for computing median
**.R?.ppp[*].**Queue.*.scalar-recording = true

**.app[*].sentPk.scalar-recording = true
**.app[*].rcvdPk.scalar-recording = true
**.app[*].endToEndDelay.scalar-recording = true

**.afQueue.*.scalar-recording = true

```

Figure 2.4: Configuration of network parameters and statistical measurements

2.2.2 Shaping and Policing

Configuring Shaping on Router R2

```

package.ned  omnetpp.ini X

[Config PolicingAndShaping]

#Shaping on Router 2
**.R2.ppp[2].egressTC.typeName = "TrafficConditioner"

#**.R2.ppp[2].ppp.queue.interfaceTableModule = "^.^.^interfaceTable"

**.R2.ppp[2].egressTC.efMeter.cir = "70%"
**.R2.ppp[2].egressTC.efMeter.cbs = 50KiB
**.R2.ppp[2].egressTC.defaultMeter.cir = "30%"
**.R2.ppp[2].egressTC.defaultMeter.cbs = 2KiB
**.R2.ppp[2].egressTC.defaultMeter.ebs = 4KiB

```

Figure 2.5: Configuring shaping on Router R2

Configuring Policing on Router R4


```

#Policing on Router 4
**R4.ppp[0].ingressTC.typeName = "TrafficConditioner"

***R4.ppp[0].ppp.queue.interfaceTableModule = "^.^interfaceTable"

**R4.ppp[0].ingressTC.efMeter.cir = "50%"
**R4.ppp[0].ingressTC.efMeter.cbs = 40KiB
**R4.ppp[0].ingressTC.defaultMeter.cir = "25%"
**R4.ppp[0].ingressTC.defaultMeter.cbs = 2KiB
**R4.ppp[0].ingressTC.defaultMeter.ebs = 4KiB

[Config Exp1DropTailWithoutPolicingAndShaping] #Best Effort without Policing and Shaping
#description = "Drop Tail Queueing Without Traffic Conditioning"

```

Figure 2.6: Configuring policing on Router R4

Configuring DropTail queueing without policing and shaping

```

[Config Exp1DropTailWithoutPolicingAndShaping] #Best Effort without Policing and Shaping
#description = "Drop Tail Queueing Without Traffic Conditioning"

extends = Apps, Exp
**ppp[*].ppp.queue.typeName = "DropTailQueue"
**eth[*].mac.queue.typeName = "EtherQosQueue"
**eth[*].mac.queue.dataQueue.typeName = "DropTailQueue"
**queue.packetCapacity = 100

```

Figure 2.7: Best effort DropTail queueing without policing and shaping

Configuring DropTail queueing with policing and shaping

```

[Config Exp2DropTailWithPolicingAndShaping] #Best Effort with Policing and Shaping
#description = "Drop Tail Queueing With Traffic Policing and Shaping"

extends = Apps, Exp, PolicingAndShaping

**ppp[*].ppp.queue.typeName = "DropTailQueue"
**eth[*].mac.queue.typeName = "EtherQosQueue"
**eth[*].mac.queue.dataQueue.typeName = "DropTailQueue"
**queue.packetCapacity = 100

```

Figure 2.8: Best effort DropTail queueing with policing and shaping

Configuring WRR queueing with policing and shaping

```

[Config Exp3RoundRobinWithPolicingAndShaping] # Round Robin with Policing and Shaping
#description = "Diffserv Queueing With Traffic Policing and Shaping"

extends = Apps, Exp, PolicingAndShaping

**R7.ppp[*].ppp.queue.typeName = "DSQueue1" #Diffserv Queue
**R7.ppp[*].ppp.queue.packetCapacity = 100 #-1
**R7.ppp[*].ppp.queue.*.packetCapacity = 100
**R7.ppp[*].ppp.queue.wrr.weights = "10 40 5 1 1"

```

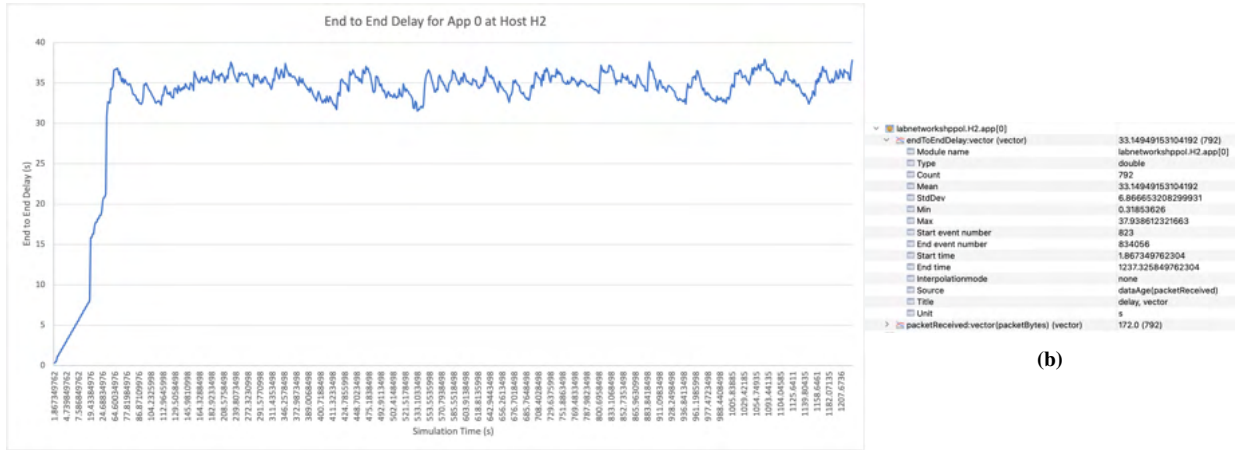
Figure 2.9: Weighted round robin with traffic policing and shaping

The filters.xml and OSPFConfig.xml files are same as in Fig. 1.6(a) and Fig. 1.6(b), respectively.

2.3 Results: Best effort (DropTail) queueing without policing and shaping

2.3.1 Results for App 0 at Host H2

End to End Delay



(a)

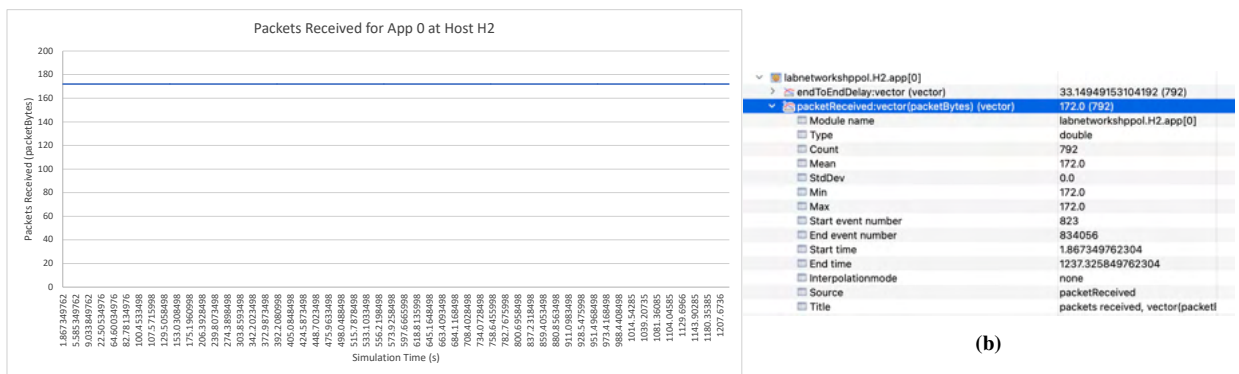
Figure 2.10: End to end delays for all 792 packets of app 0 received at host H2

Packets Received Out of Order

labnetworkshppol.H2.app[0]	
endToEndDelay:vector (vector)	33.14949153104192 (792)
outOfOrderPk:count (scalar)	0.0
Module name	labnetworkshppol.H2.app[0]
Type	double
Value	0.0
Interpolationmode	none
Source	outOfOrderPk
Title	packets received out of order, cc
outOfOrderPk:sum(packetBytes) (scalar)	0.0
packetReceived:count (scalar)	792.0

Figure 2.11: Packets Received Out of Order

Packets Received



(a)

Figure 2.12: Received packets vector for app 0 at host H2

Sent Packets

▼ Total sent (scalar)	0.0
Module name	labnetworkshppol.H2.app[0]
Type	double
Value	0.0

Figure 2.13: Packets Sent

2.3.2 Results for App 1 at Host H2

End to End Delay

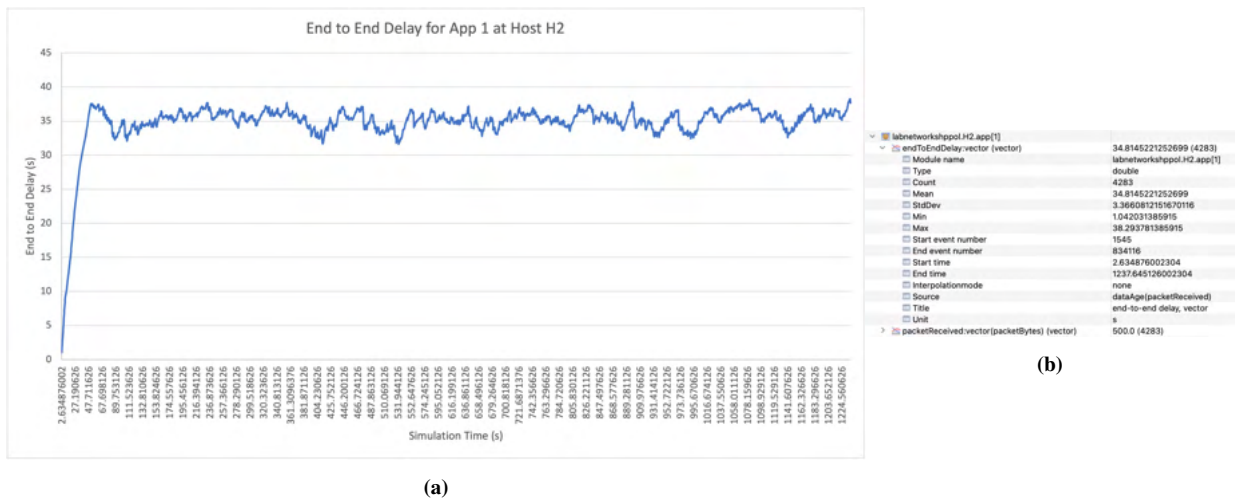


Figure 2.14: End to end delays for all 4283 packets of app 1 received at host H2

Received Packets

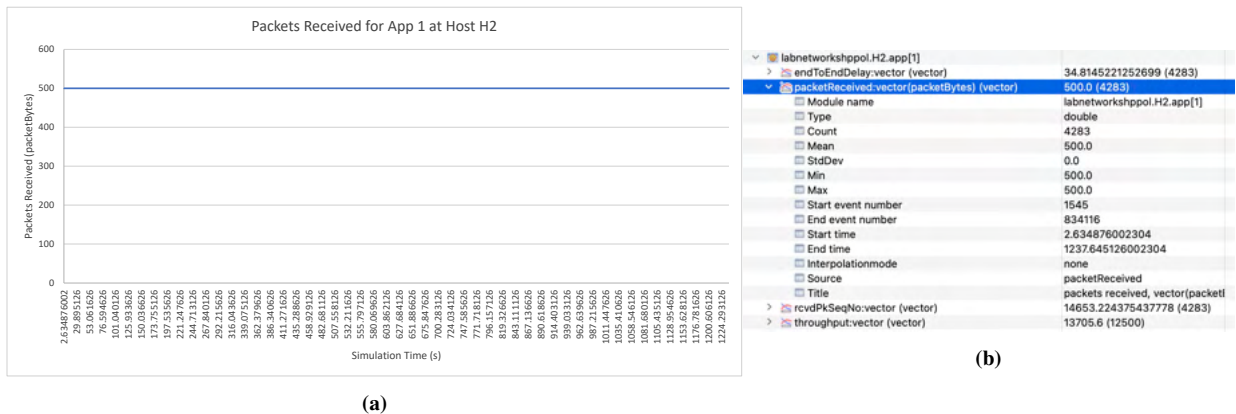
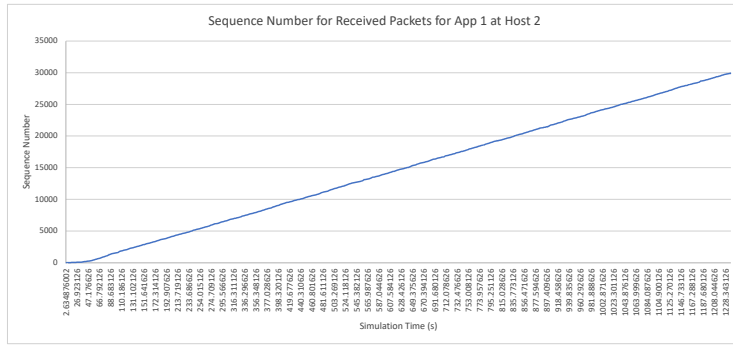


Figure 2.15: Received packets vector for app 1 at host H2

Sequence Number for Received Packets



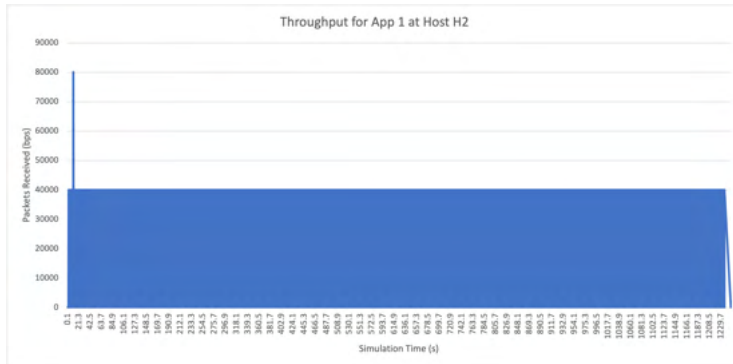
(a)

labnetworkshppol.H2.app[1]	
endToEndDelay:vector (vector)	34.8145221252699 (4283)
packetReceived:vector(packetBytes) (vector)	500.0 (4283)
rcvdPkSeqNo:vector (vector)	14653.224375437778 (4283)
Module name	labnetworkshppol.H2.app[1]
Type	double
Count	4283
Mean	14653.224375437778
StdDev	8888.268617727093
Min	0.0
Max	29960.0
Start event number	1545
End event number	834116
Start time	2.634876002304
End time	12.37645126002304
Interpolationmode	none
Source	appPkSeqNo(packetReceived)
Title	received packet sequence numb
throughput:vector (vector)	13705.6 (12500)

(b)

Figure 2.16: Sequence number vector of received packets for app 1 at host H2

Throughput Vector



(a)

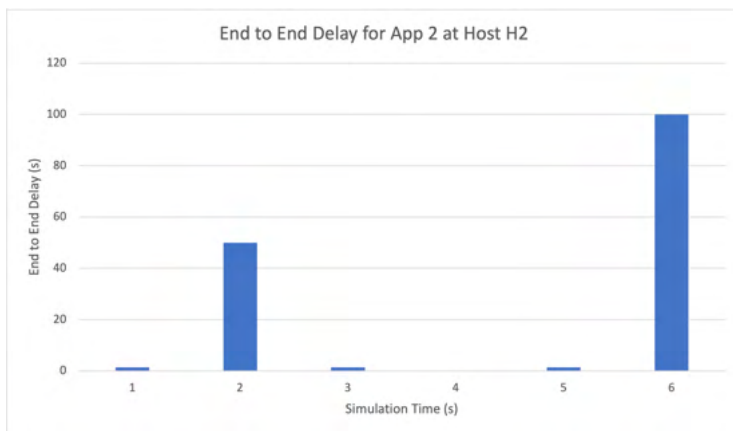
labnetworkshppol.H2.app[1]	
endToEndDelay:vector (vector)	34.8145221252699 (4283)
packetReceived:vector(packetBytes) (vector)	500.0 (4283)
rcvdPkSeqNo:vector (vector)	14653.224375437778 (4283)
throughput:vector (vector)	13705.6 (12500)
Module name	labnetworkshppol.H2.app[1]
Type	double
Count	12500
Mean	13705.6
StdDev	18991.192218250795
Min	0.0
Max	80000.0
Start event number	1545
End event number	834895
Start time	0.1
End time	1250
Source	throughput(packetReceived)
Title	throughput, vector
Unit	bps

(b)

Figure 2.17: Throughput vector for app 1 at host H2

2.3.3 Results for App 2 at Host H2

End to End Delay



(a)

labnetworkshppol.H2.app[2]	
endToEndDelay:vector (vector)	0.16526514 (1)
Module name	labnetworkshppol.H2.app[2]
Type	double
Count	1
Mean	0.16526514
StdDev	Infinity
Min	0.16526514
Max	0.16526514
Start event number	396
End event number	396
Start time	1.33431162
End time	1.33431162
Interpolationmode	none
Source	dataAge(packetReceived)
Title	end-to-end delay, vector
Unit	s
packetReceived:vector(packetBytes) (vector)	50.0 (1)

(b)

Figure 2.18: End to end delay vector for app 2 received at host H2

Received Packets

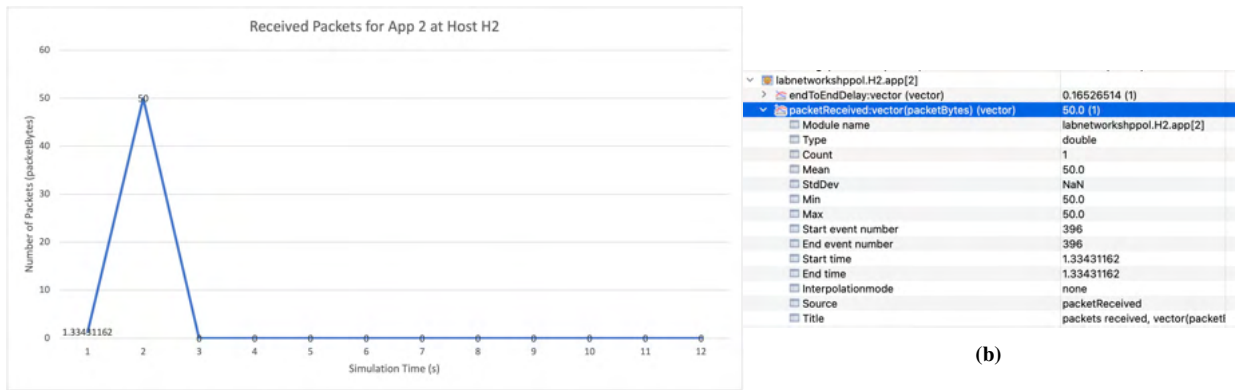


Figure 2.19: Received packets vector for app 2 at host H2

Sent Packets



Figure 2.20: Sent packets vector for app 2 at host H2

2.4 Results: Best effort (DropTail) queueing with policing and shaping

2.4.1 Results for App 0 at Host H2

End to End Delay

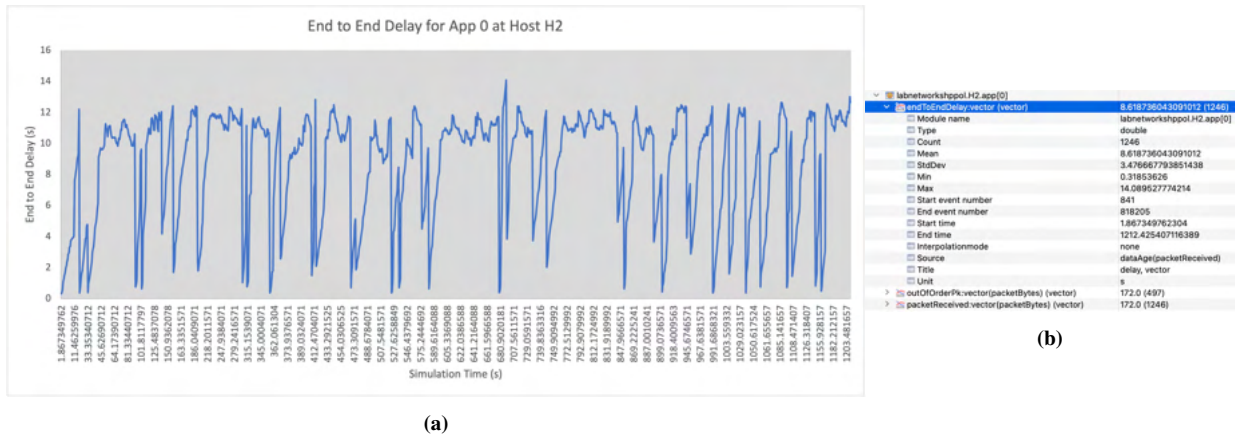


Figure 2.21: End to end delays for all 1246 packets of app 0 received at host H2

Received Out of Order Packets

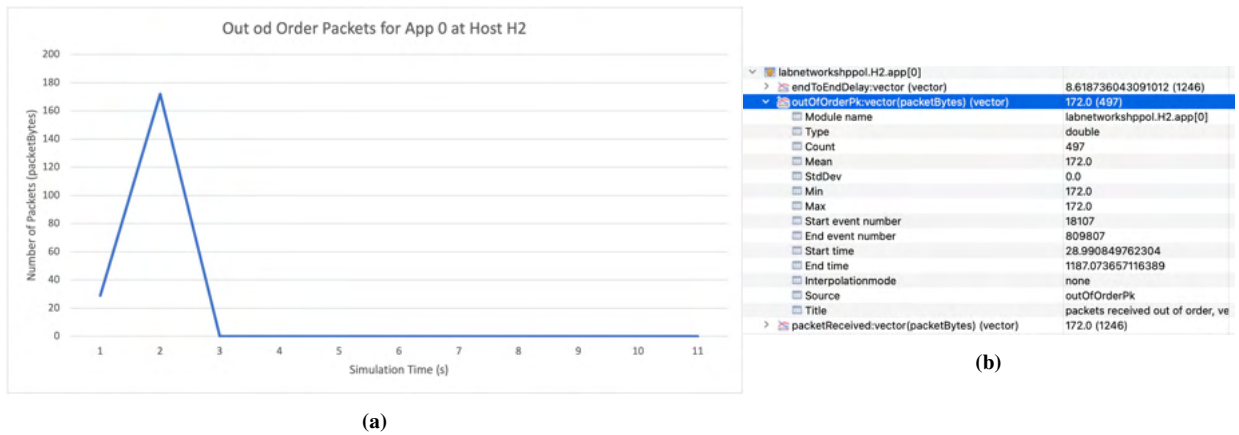


Figure 2.22: Received out of order packets vector for app 0 at host H2

Received Packets

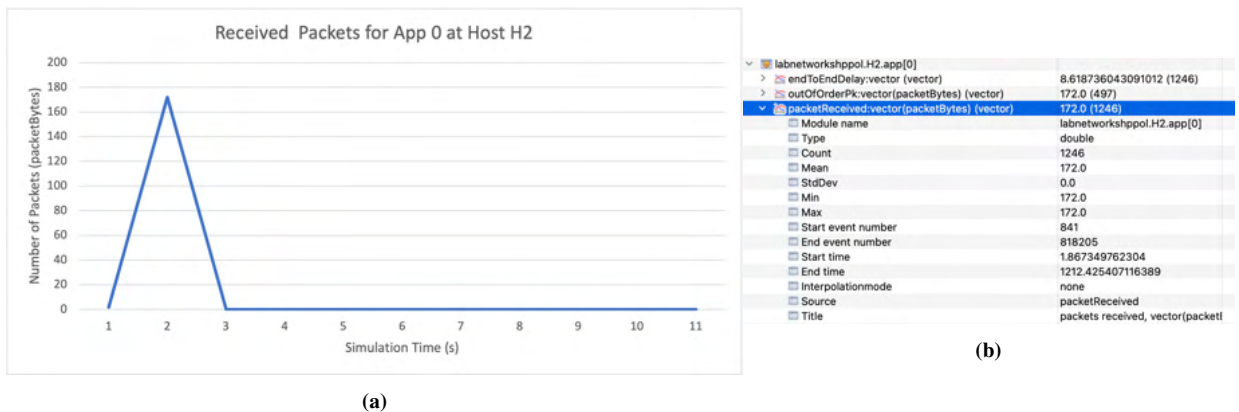


Figure 2.23: Received packets vector for app 0 at host H2

Sent Packets

packetSent:sum(packetBytes) (scalar)	0.0
Module name	labnetworkshppol.H2.app[0]
Type	double
Value	0.0
Interpolationmode	none
Source	packetSent
Title	packets sent, sum(packetBytes)
Total deleted (scalar)	0.0
Total received (scalar)	1246.0
Total sent (scalar)	0.0

Figure 2.24: Packets Sent

2.4.2 Results for App 1 at Host H2

End to End Delay

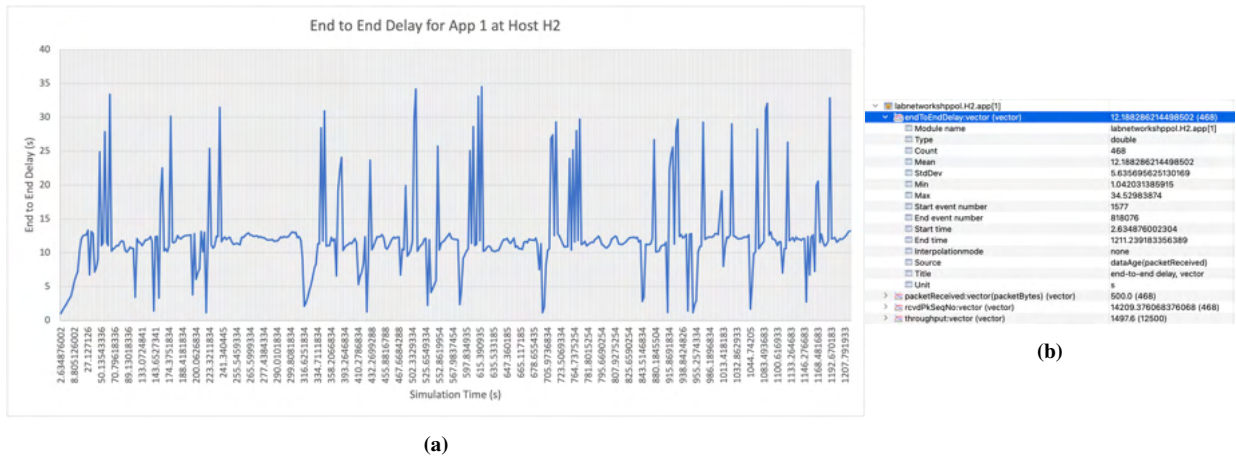


Figure 2.25: End to end delays for all 468 packets of app 1 received at host H2

Received Packets

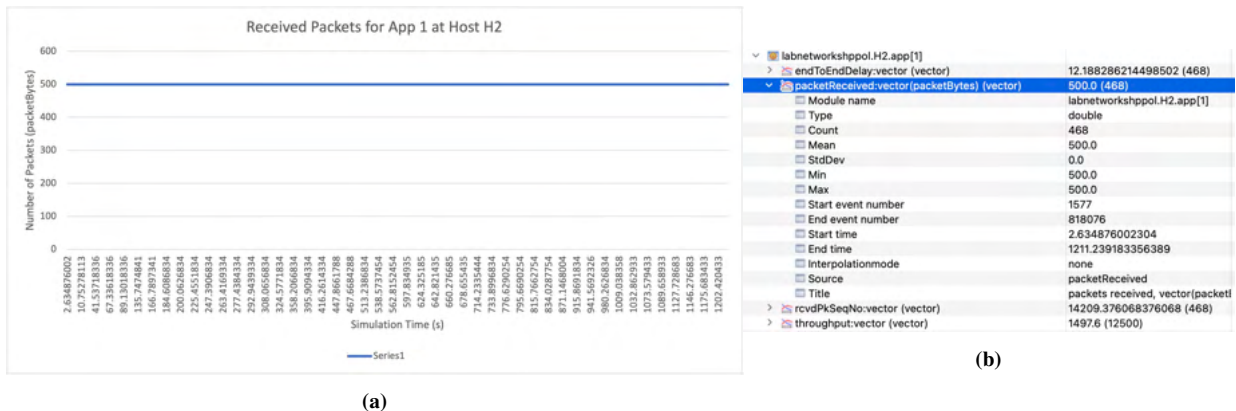
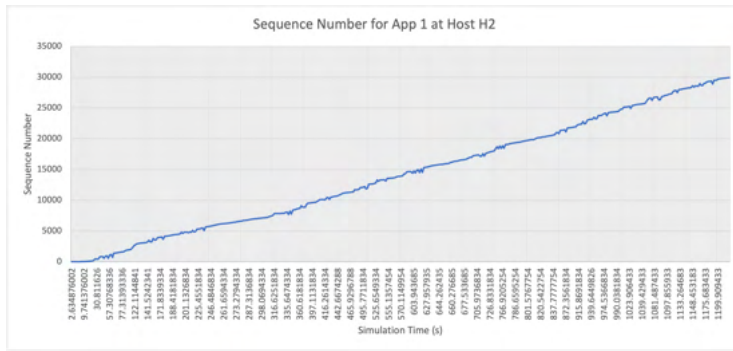


Figure 2.26: Received packets vector for app 1 at host H2

Sequence Number for Received Packets

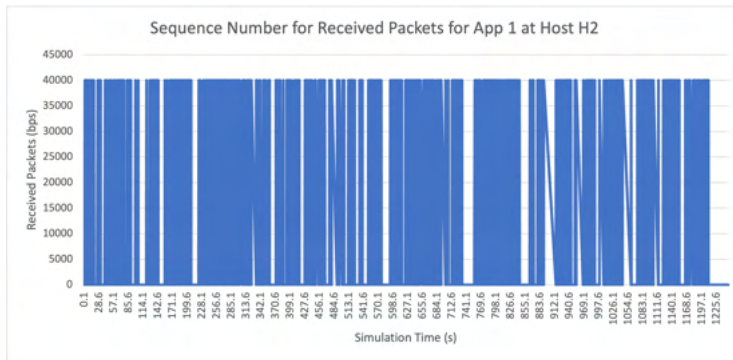


labnetworkshppol.H2.app[1]	
endToEndDelay:vector (vector)	12.188286214498502 (468)
packetReceived:vector(packetBytes) (vector)	500.0 (468)
rcvPkSeqNo:vector (vector)	14209.376068376068 (468)
Module name	labnetworkshppol.H2.app[1]
Type	double
Count	468
Mean	14209.376068376068
StdDev	8841.896568549846
Min	0.0
Max	29911.0
Start event number	1577
End event number	818076
Start time	2.634876002304
End time	1211.239183356389
Interpolationmode	none
Source	appPkSeqNo(packetReceived)
Title	received packet sequence numb
throughput:vector (vector)	1497.6 (12500)

(b)

Figure 2.27: Sequence number vector of received packets for app 1 at host H2

Throughput Vector



labnetworkshppol.H2.app[1]	
endToEndDelay:vector (vector)	12.188286214498502 (468)
packetReceived:vector(packetBytes) (vector)	500.0 (468)
rcvPkSeqNo:vector (vector)	14209.376068376068 (468)
throughput:vector (vector)	1497.6 (12500)
Module name	labnetworkshppol.H2.app[1]
Type	double
Count	12500
Mean	1497.6
StdDev	7593.800596842161
Min	0.0
Max	40000.0
Start event number	1577
End event number	819651
Start time	0.1
End time	1250
Source	throughput(packetReceived)
Title	throughput, vector
Unit	bps

(b)

Figure 2.28: Throughput vector for app 1 at host H2

2.4.3 Results for App 2 at Host H2

End to End Delay



labnetworkshppol.H2.app[2]	
endToEndDelay:vector (vector)	16.0273319886375 (2)
Module name	labnetworkshppol.H2.app[2]
Type	double
Count	2
Mean	16.0273319886375
StdDev	22.432350061783826
Min	0.16526514
Max	31.889398833275
Start event number	404
End event number	41014
Start time	1.33431162
End time	58.481398316389
Interpolationmode	none
Source	dataAge(packetReceived)
Title	end-to-end delay, vector
Unit	s
packetReceived:vector(packetBytes) (vector)	50.0 (2)
packetSent:vector(packetBytes) (vector)	100.0 (2)

(b)

(a)

Figure 2.29: End to end delay vector for app 2 received at host H2

Received Packets

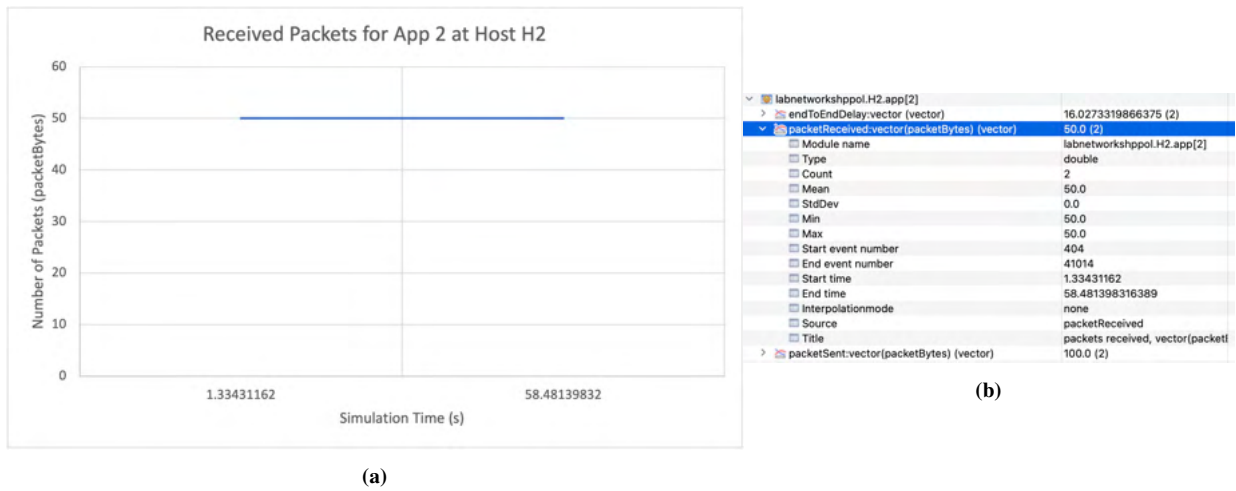


Figure 2.30: Received packets vector for app 2 at host H2

Sent Packets

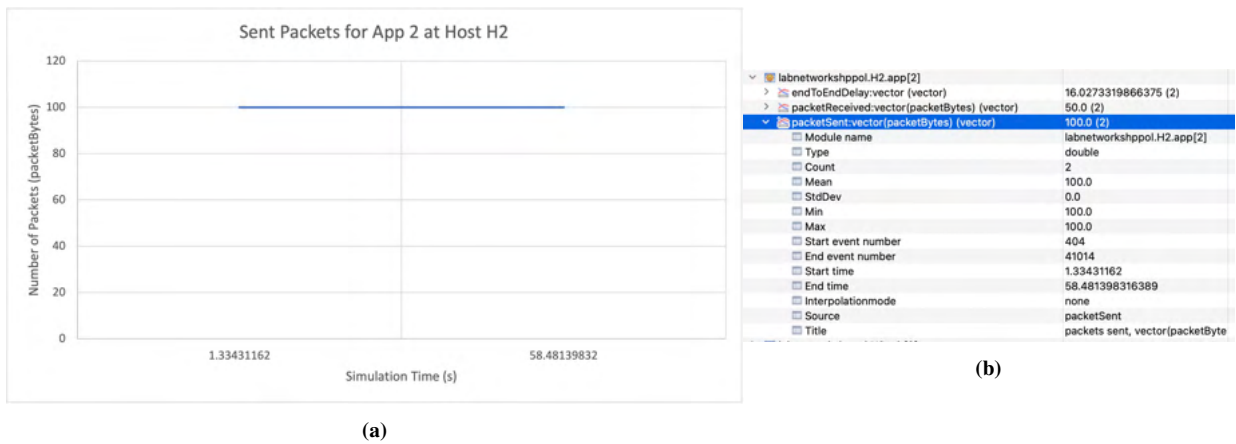


Figure 2.31: Sent packets vector for app 2 at host H2

2.5 Results: WRR queueing with policing and shaping

2.5.1 Results for App 0 at Host H2

End to End Delay

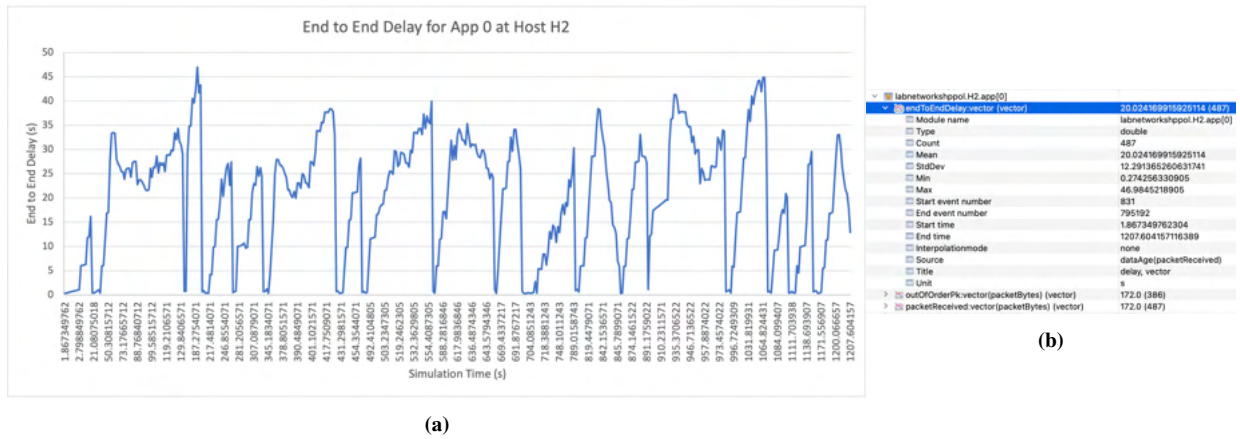


Figure 2.32: End to end delays for all 487 packets of app 0 received at host H2

Received Out of Order Packets

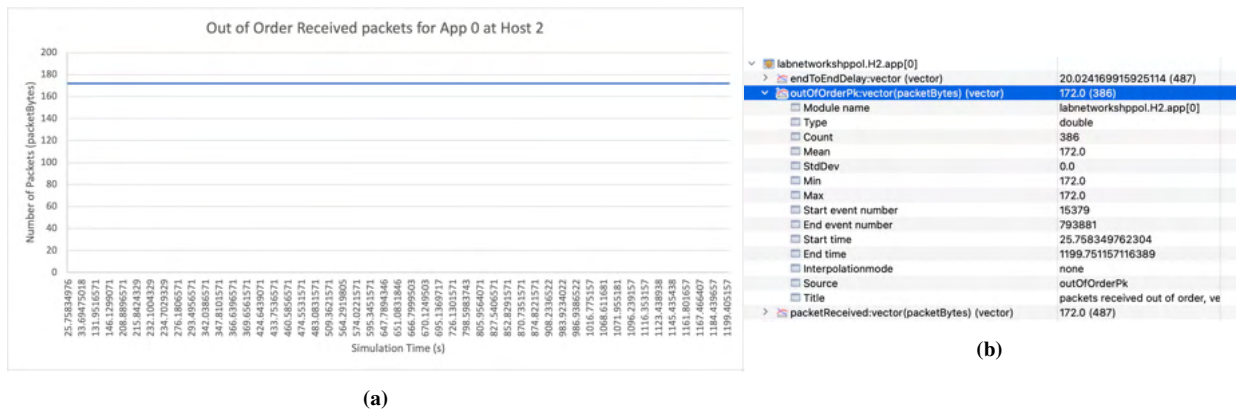


Figure 2.33: Received out of order packets vector for app 0 at host H2

Received Packets

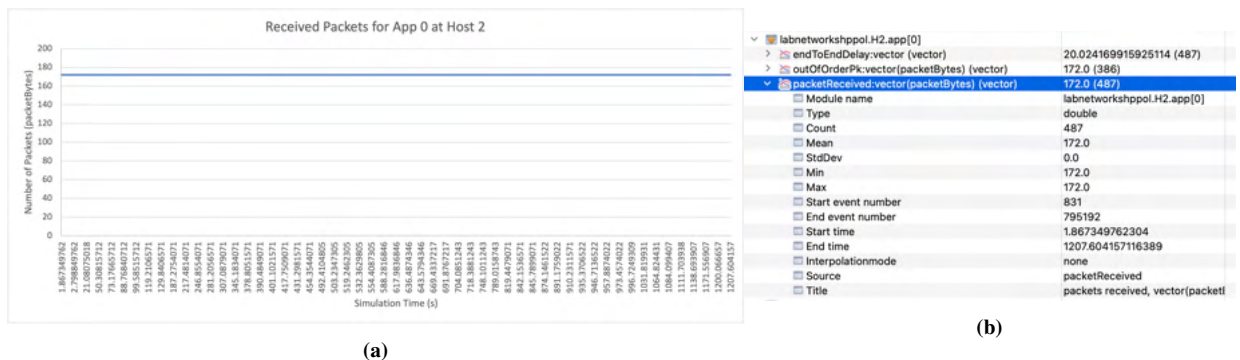


Figure 2.34: Received packets vector for app 0 at host H2

Sent Packets

> packetSent:count (scalar)	0.0
▼ packetSent:sum(packetBytes) (scalar)	0.0
Module name	labnetworkshppol.H2.app[0]
Type	double
Value	0.0
Interpolationmode	none
Source	packetSent
Title	packets sent, sum(packetBytes)
> Total deleted (scalar)	0.0
> Total received (scalar)	487.0
> Total sent (scalar)	0.0

Figure 2.35: Packets Sent

2.5.2 Results for App 1 at Host H2

End to End Delay

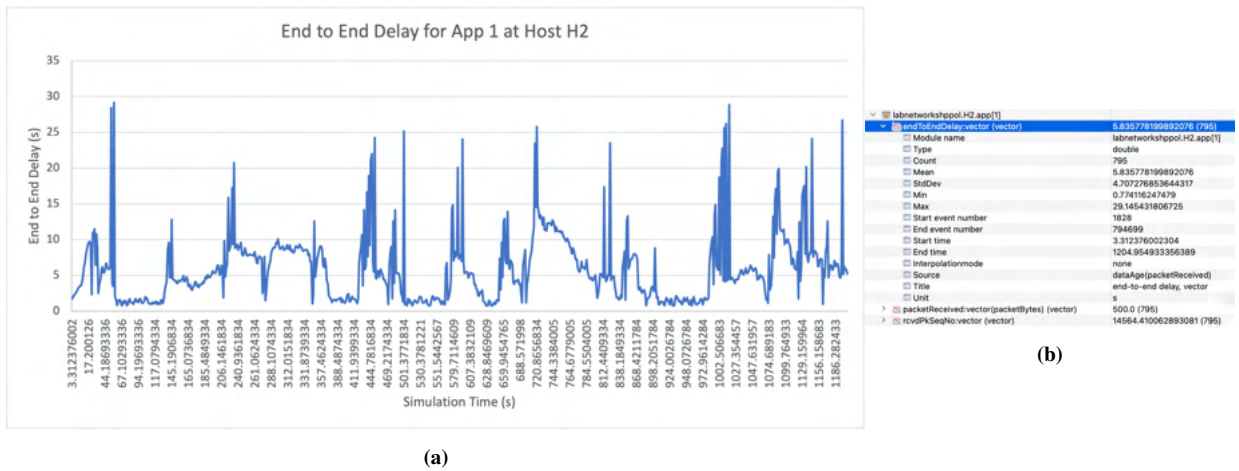


Figure 2.36: End to end delays for all 795 packets of app 1 received at host H2

Received Packets

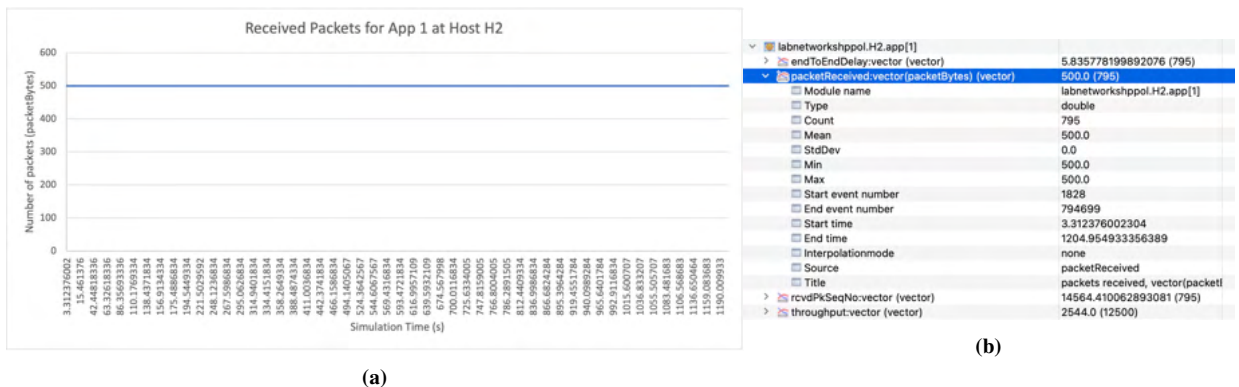


Figure 2.37: Received packets vector for app 1 at host H2

Sequence Number for Received Packets

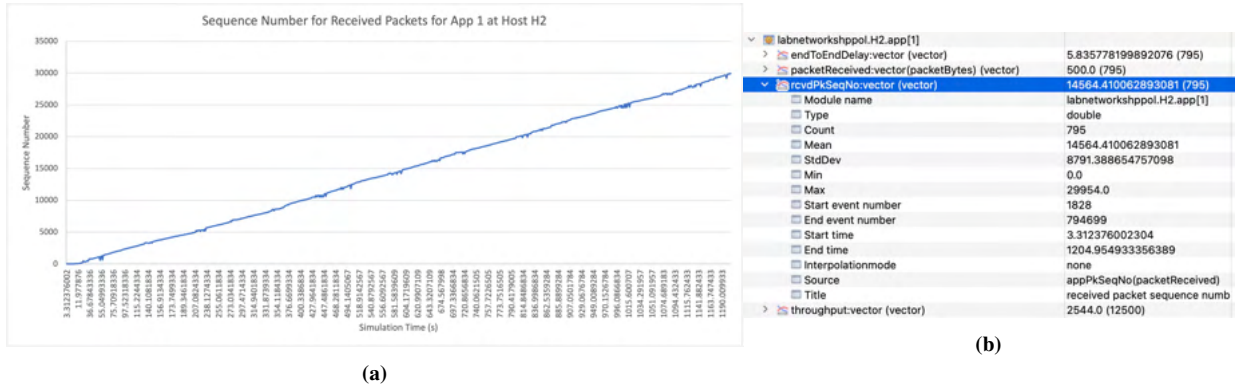


Figure 2.38: Sequence number vector of received packets for app 1 at host H2

Throughput Vector

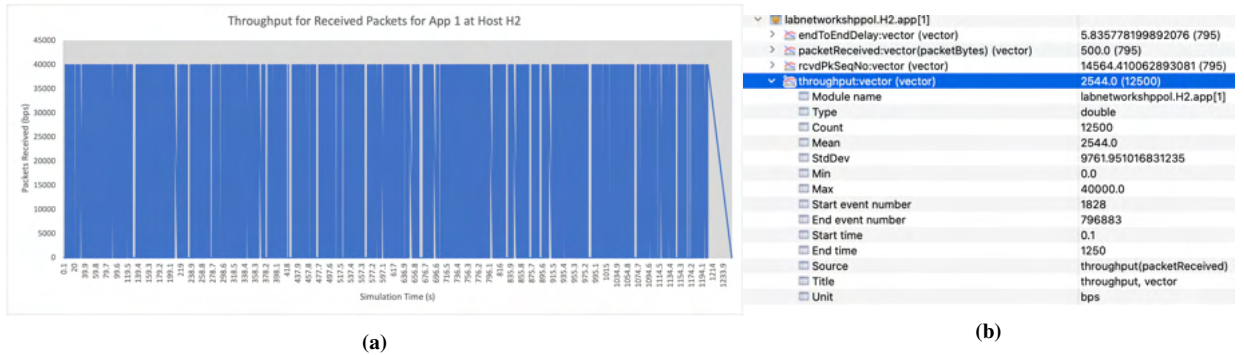


Figure 2.39: Throughput vector for app 1 at host H2

2.5.3 Results for App 2 at Host H2

End to End Delay



Figure 2.40: End to end delay vector for app 2 received at host H2

Received Packets

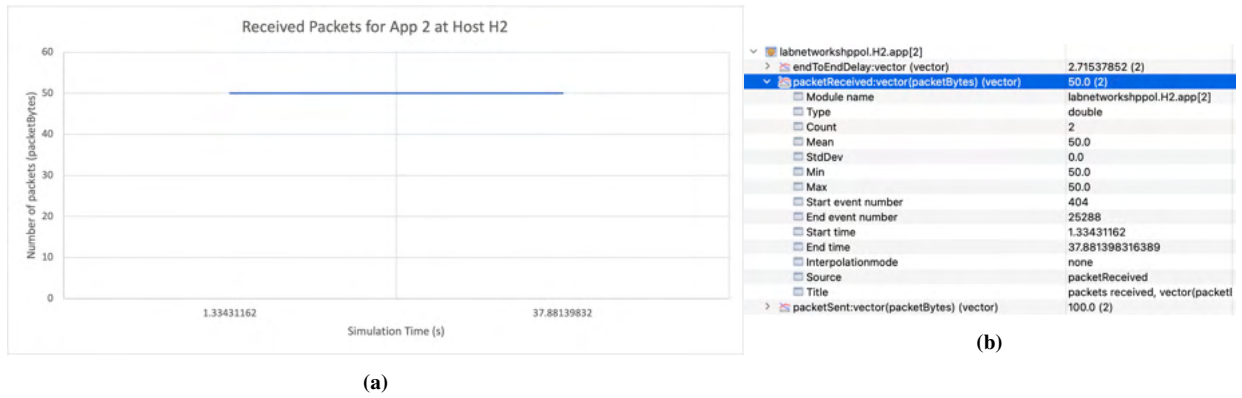


Figure 2.41: Received packets vector for app 2 at host H2

Sent Packets

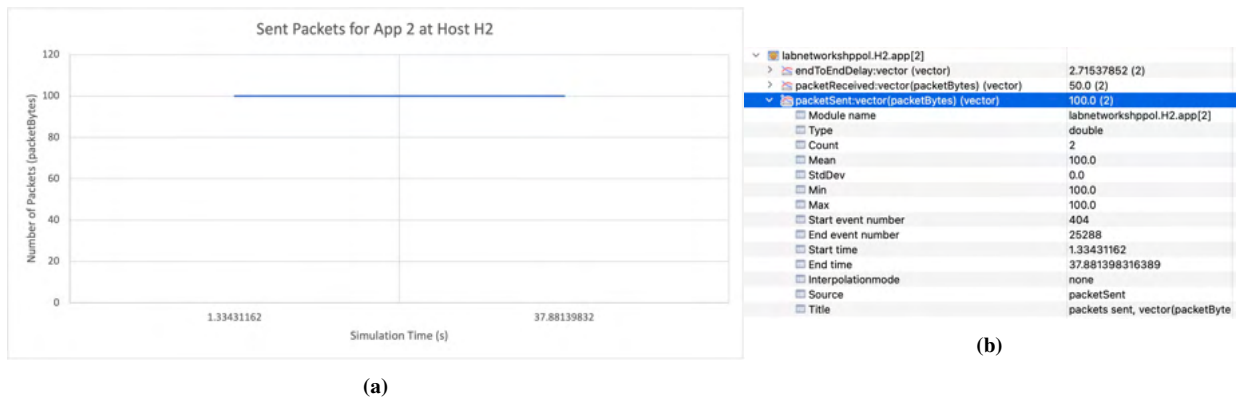


Figure 2.42: Sent packets vector for app 2 at host H2

3 Discussion

3.1 Comparison of Queueing Strategies in Experiment 1: Enable Queueing QoS

3.1.1 App 0 at Host H2

End to End Delay

Referring to the Fig. 1.7 and Fig. 1.17, the best effort DropTail queueing end to end delay count is for the 870 received packets which is the same in the case of WRR DiffServ queueing. In DropTail, it starts at 2.32s when the simulation begins but for DiffServ, it starts at 1.77s. The end time is same in both of the cases. The mean 46.21s with standard deviation of 12.69s in DropTail while in DiffServ, it stands for 39.70s with standard deviation of 14.26s. The WRR DiffServ proved to be the best for end to end delay scenario.

Packets Received Out of Order

The number of received packets which dropped or out of order is 120 out of 870 received packets in both cases as shown in Fig. 1.18 and Fig. 1.8, respectively.

Packets Received

The number of received packets is 870 with a mean of 172 with the start time at 1.77s and the end time at 1249.17s in both cases as shown in Fig. 1.9 and Fig. 1.19, respectively.

3.1.2 App 1 at Host H2

End to End Delay

Referring to the Fig. 1.10 and Fig. 1.20, the best effort DropTail queueing end to end delay count is for the 4237 received packets which is the same in the case of WRR DiffServ queueing. In DropTail, it starts at 2.32s when the simulation begins in both the cases. The end time is same in both of the cases. The mean 46.21s with standard deviation of 12.69s in DropTail as well as in DiffServ. The WRR DiffServ proved to be the same for end to end delay scenario.

Packets Received

The number of packets received are 4237 with the mean of 500 packetBytes in both cases as

shown in Fig. 1.11 and Fig. 1.21, respectively.

Sequence Number for the Packets Received

The sequence number for the 4237 packets received characterized by parameters mean and standard deviation which are 14600.11 and 8832.12 as shown in Fig. 1.12 and Fig. 1.22, respectively.

Throughput Vector

As shown in Fig. 1.13 and Fig. 1.23, for DropTail, 12500 packets have been received and the throughput vector stands with the mean of 13558.4 while in DiffServ, for the same 12500 count, the throughput vector is with the mean of 13321.6. In this case, DropTail proved to be better and advantageous as it can pass more number of packetBytes in the network.

3.1.3 App 2 at Host H2

End to End Delay

Referring to the Fig. 1.14 and Fig. 1.24, the best effort DropTail queueing end to end delay count is for the 1 received packets which is the same in the case of WRR DiffServ queueing. In both cases, it starts and ends at 1.22s. The mean 0.1147s in both scenarios.

Packets Received

As shown in Fig. 1.15 and Fig. 1.25, the count for the number of packets received are 1 with 50 packetBytes only, in both cases.

Packets Sent

As shown in Fig. 1.16 and Fig. 1.26, the count for the number of packets received are 1 with 100 packetBytes only, in both cases.

3.2 Comparison of Queueing Strategies in Experiment 2: Traffic shaping and policing

3.2.1 App 0 at Host H2

End to End Delay

Referring to the Fig. 2.10, for the best effort DropTail queueing without traffic shaping and

policing (Exp 1) end to end delay count is for the 792 received packets which is 1246 in the case of the best effort DropTail queueing with traffic shaping and policing (Exp 2) in Fig. 2.21, but for WRR DiffServ with traffic shaping and policing(Exp 3) in and Fig. 2.32, the end to end delay count is for the 487 received packets only.

In DropTail without and with traffic shaping and policing, it starts at 1.86s when the simulation begins which is the same for DiffServ with traffic shaping and policing. The end time is same in all three cases at 1.86s. The mean 33.14s in Exp 1 with standard deviation of 6.86s while in Exp 2, it stands for 8.61s with standard deviation of 3.47s and for Exp 3, this is 20.02s with standard deviation of 12.29s. So, with traffic shaping and policing, the round robin DiffServ is better when compared to the Drop tail queueing strategy, however, DropTail queueing method without traffic shaping and policing is the best among all three scenarios.

Packets Received Out of Order

The number of received packets which dropped or out of order are none in Exp 1 as shown in Fig. 2.11. For Exp 2 and Exp 3, the out of order packets are 497 and 386, as shown in Fig. 2.22 and Fig. 2.33, respectively.

Packets Received

The number of received packets is 792 with a mean of 172 with the start time at 1.86s and the end time at 1237.32s in Exp 1 as shown in Fig. 2.12, which for Exp 2 is 497 with a mean of 172, start time at 1.86s and end time at 1212.42s as shown in Fig. 2.23 and for Exp 3, 386 packets are received with a mean of 172, start time 1.86s and end time at 1207.60s as shown in Fig. 2.34, respectively. So, DropTail without traffic shaping and policing is the best among all three scenarios.

Packets Sent

The number of sent packets are none in Exp 1, Exp 2 and Exp 3 as shown in Fig. 2.13, Fig. 2.35, respectively.

3.2.2 App 1 at Host H2

End to End Delay

Referring to the Fig. 2.14, in the the best effort DropTail queueing without traffic shaping

and policing (Exp 1), the end to end delay count is for the 4283 received packets with a mean of 34.81s and standard deviation of 3.36s, and which starts as 2.63s and ends at 1237.64s. In Fig. 2.25 for Exp 2 with traffic shaping and policing, the count is 468 with a mean of 12.18s and standard deviation of 5.63, and which starts at 2.63s and ends at 1211.23s. In Fig. 2.36, for WRR with traffic shaping and policing (Exp 3), the count is 795 with a mean of 5.83s (which is the lowest) and standard deviation of 4.70s, and which starts at 3.31s and ends at 1204.95s.

Packets Received

The number of packets received are 4283 with a mean of 500 packetBytes in Exp 1 as shown in Fig. 2.15. For Exp 2, the packets received are 468 with a mean of 500 packetBytes as shown in Fig. 2.26, and for Exp 3 as shown in Fig. 2.37, the count for the received packets are 795 with a mean of 500 packetBytes.

Sequence Number for the Packets Received

For Exp 1 as shown in Fig. 2.16, the sequence number for the 4283 packets received are obtained with a mean of 14653.22 and standard deviation of 8888.26. In Fig. 2.27, the sequence number is given by a mean of 14209.37 with a standard deviation of 8841.89. For Exp 3, the sequence number is presented by a mean of 14564.41 and a standard deviation of 8791.38 as shown in Fig. 2.38. Clearly, the Exp 3 with shaping and policing is the best choice among all the three above mentioned scenarios.

Throughput Vector

For Exp 1 as shown in Fig. 2.17, for the total number of received packets which are 4283, the throughput is 12500bps with a mean of 13705.6bps. For Exp 2, the throughput vector is 12500bps with the mean of 1497.6bps where the total number of received packets are 468 as shown in Fig. 2.28, and for Exp 3 as shown in Fig. 2.39, the throughput vector is 12500bps with a mean of 2544bps for 795 received packets. Once again, with shaping and policing, round robin is better than the Droptail in Exp 2.

3.2.3 App 2 at Host H2

End to End Delay

Referring to the Fig. 2.18, the end to end delay vector for Exp1 is given with a mean of

0.16s but without a finite standard deviation only for the single count of packet received. For Exp 2 given in Fig. 2.29, the end to end delay vector is having a mean of 16.02s and a standard deviation of 22.43s. In Fig. 2.40, the end to end delay for Exp 3 is having a mean of 2.71s and standard deviation of 3.60s, which is available for 2 counts of received packets.

Packets Received

For Exp 1, as shown in Fig. 2.19, the count for the number of packets received are 1 with 50 packetBytes only. For Exp 2 and Exp 3, as shown in Fig. 2.30 and Fig. 2.41, respectively, the count for the number of packets received are 2 with 50 packetBytes only.

Packets Sent

For Exp 1, as shown in Fig. 2.20 , the count for the number of packets sent are 1 with 100 packetBytes. For Exp 2 and Exp 3, as shown in Fig. 2.31 and Fig. 2.42, respectively, the count for the number of packets received are 2 with 100 packetBytes.

4 Conclusion

In this report, in the first part, the best effort and weighted round robin (WRR) queueing methods have been used to enhance the performance of the network. The WRR (DiffServ) method has been proved to be the most effective one based on the numerical results and data from the simulation. The end to end delay is better with WRR whereas the throughput is better in DropTail best effort method. In the second part of the report, the traffic shaping and policing has been applied to the network with the same queueing strategies as in the first part of the report. Based on the results, the DropTail queueing method is proved to be better in the case if traffic shaping and policing has not been applied. With traffic shaping and policing, the WRR (DiffServ) has better performance compared to DropTail with traffic shaping and policing.

References

- [1] A. Agarwal. “Elec 6181 project phase 1” *The Department of Electrical and Computer Engineering, Gina Cody School of Engineering and Computer Science, Concordia University*. [Online]. Available: https://moodle.concordia.ca/moodle/pluginfile.php/5304298/mod_resource/content/1/ELEC6181%20Project%20Phase%20One.pdf
- [2] TechTarget. [Online]. Available: <https://www.techtarget.com/searchnetworking/answer/Whats-the-difference-between-an-edge-router-vs-core-router>
- [3] C. Press. “Cisco ip telephony flash cards: Weighted random early detection (wred),” *Cisco Press*,. [Online]. Available: <https://www.ciscopress.com/articles/article.asp?p=352991&seqNum=6>