Differentiated Services Simulator Manual

This is the differentiated services simulator manual.

1. Overview

This document is a guide for users to use this application to differentiated services in ns-3 in Strict Priority Queueing (SPQ) and Deficit Round Robin (DRR), respectively. This document assumes that the reader has been successfully installed “Ubuntu 18.04.1 LTS” Linux distribution, ns-3 and Jsoncpp library.

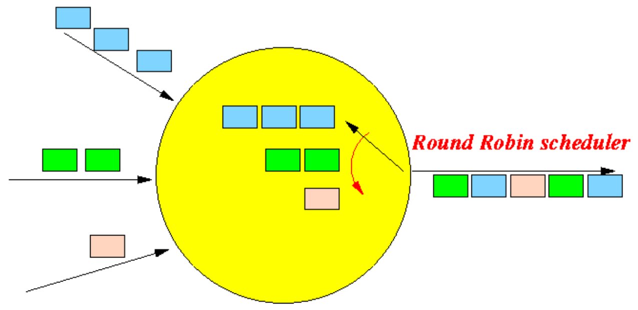
2. Background

This project follows the work led by Vahab Pournaghshband in *Differentiated Service Queuing Disciplines in NS-3*. This project aims at implementing a a selection of differentiated services: SPQ and DRR, which will be validated and verified. By observing the simulation, we expect that by implementing differentiated services, traffic with different attributes will be allocated different bandwidth accordingly.

2. Terminology

- **Ns-3.** ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.
- **PPP. Point-to-Point Protocol** (**PPP**) is a link layer communications protocol used to establish a direct connection between two nodes. It connects two routers directly without any host or any other networking device in between.
- **Node**. A node is a connection point that can receive, create, store or send data along distributed network routes.
- **Strict Priority Queueing (SPQ).** With strict priority queuing, the switch services the eight queues in order of their priority. The highest priority queue is serviced until it is empty, and then the lower priority queues are serviced sequentially until they are empty.

- **Deficit Round Robin(DRR).** Packets from different flows are stored in different queues. Queues are serviced in a Round-Robin manner.



3. Organization

Pursuant to the requirement above, our design below depicts the topology of 3 nodes network:

```
    10.1.1.0             10.1.3.0
n0 ------------- n1    ------------- n2
point-to-point        point-to-point
```

The IP addresses are assigned as follows:
Client IP : 10.1.1.1 (n0)
Server IP : 10.1.2.2 (n2)

4. Major Base Classes

This application is built on four base classes. Major functions are designed to the following classes:
DiffServ: provides basic functionalities required to simulate differentiated services
TrafficClass: This class defines the characteristics of a queue
Filter:This class is a collection of conditions as FilterElements, all of which should match in order to trigger match
FilterElement: This class is a base class for a primitive condition to match on

5. Getting Started

5.1. build

In order to run an ns-3 program, you need to build and run in the right directory. First, go to the directory ns-3-dev by:

```
$ cd ns-3-dev
```

Then you will configure the project by:

```
$ ./waf configure--enable-examples
```

The build system is now configured and you can build the debug versions of the ns-3 programs by simply typing:

```
$ ./waf
```

5.2. run

You will run scripts under shell script we provided. This allows the build system to ensure that the shared library paths are set correctly and that the libraries are available at run time. To run a program, simply use the ./ option.

Run the ns-3 program by typing the following:

```
$ ./step1.sh
$ ./step2.sh
```

5.3 configure file

We also made a configuration file for both the SPQ and DDR simulations.

1. spq.json can be configured to set the number of queues, the priority for each queue, and the filter type of each queue to match
2. ddr.json can be configured to increase the number of queues, the quantum given in each queue and the filter type of each queue to match

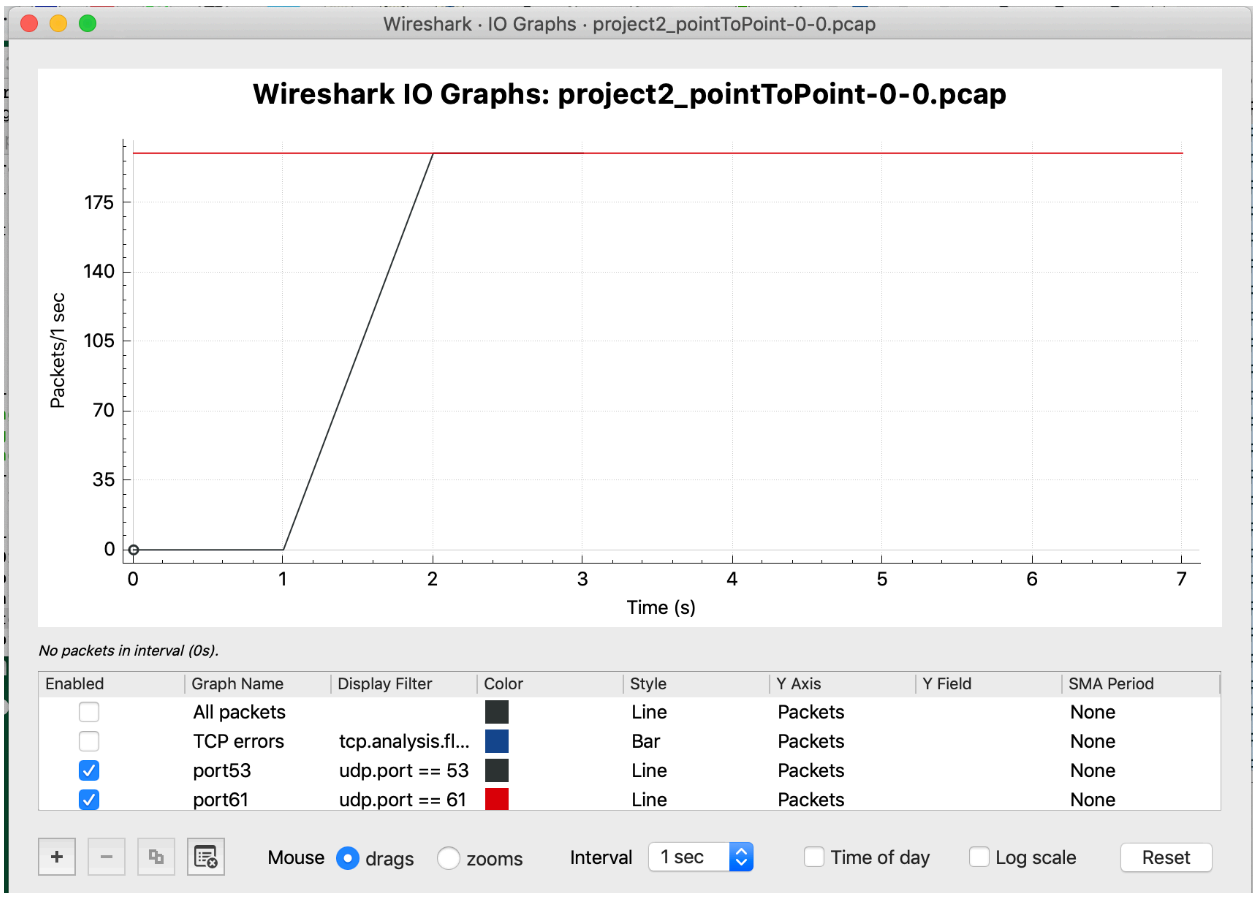
6. Simulation Validation

We included four pcap file to validate the intended result.

1. Pre SPQ.pcap This captures all packets immediately before the SPQ node, showing a number of only low priority packets first, followed by interleaving high and low priority packets).
2. Post SPQ.pcap This captures all packets immediately after the SPQ node, showing a number of only low priority packets first, followed by only high priority packets, followed by only low priority packets.
3. Pre DRR.pcap This captures all packets immediately before the DRR node, showing the interleaving three traffic streams close to a 123123123 pattern, if 1, 2, 3 represent the packets in each traffic stream.
4. Post DRR.pcap This captures all packets immediately after the DRR node, showing a pattern of packets closely aligned with the respective quanta (weights).

The simulation result is also depicted in the following figure:

FIGURE1. Throughput vs. time before the SPQ node

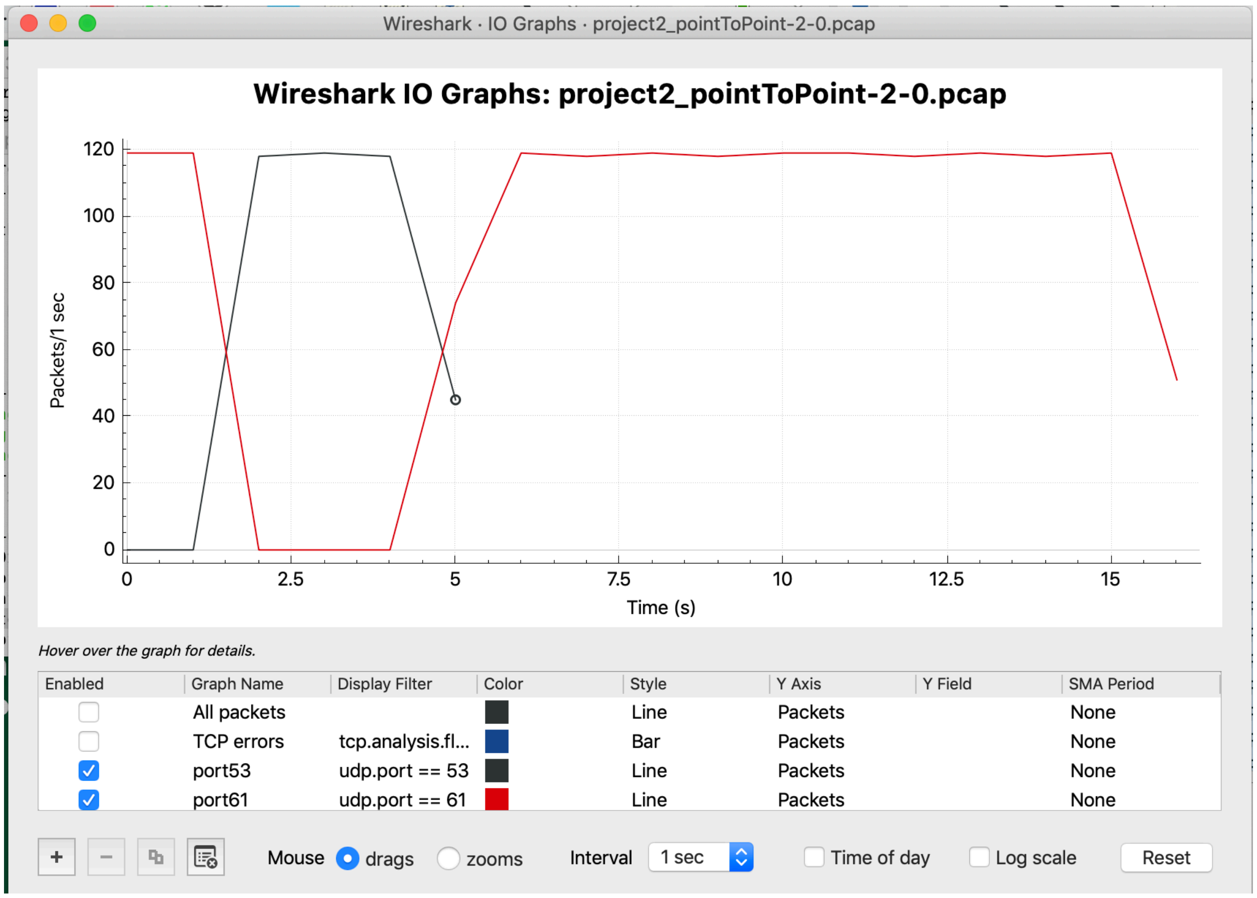

Wireshark · IO Graphs · project2_pointToPoint-0-0.pcap
Wireshark IO Graphs: project2_pointToPoint-0-0.pcap
Packets/1 sec
175
140
105
70
35
0
0
1
2
3
4
5
6
7
Time (s)

No packets in interval (0s).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	All packets		⬛	Line	Packets		None
☐	TCP errors	tcp.analysis.fl...	🟦	Bar	Packets		None
✓	port53	udp.port == 53	⬛	Line	Packets		None
✓	port61	udp.port == 61	🟥	Line	Packets		None

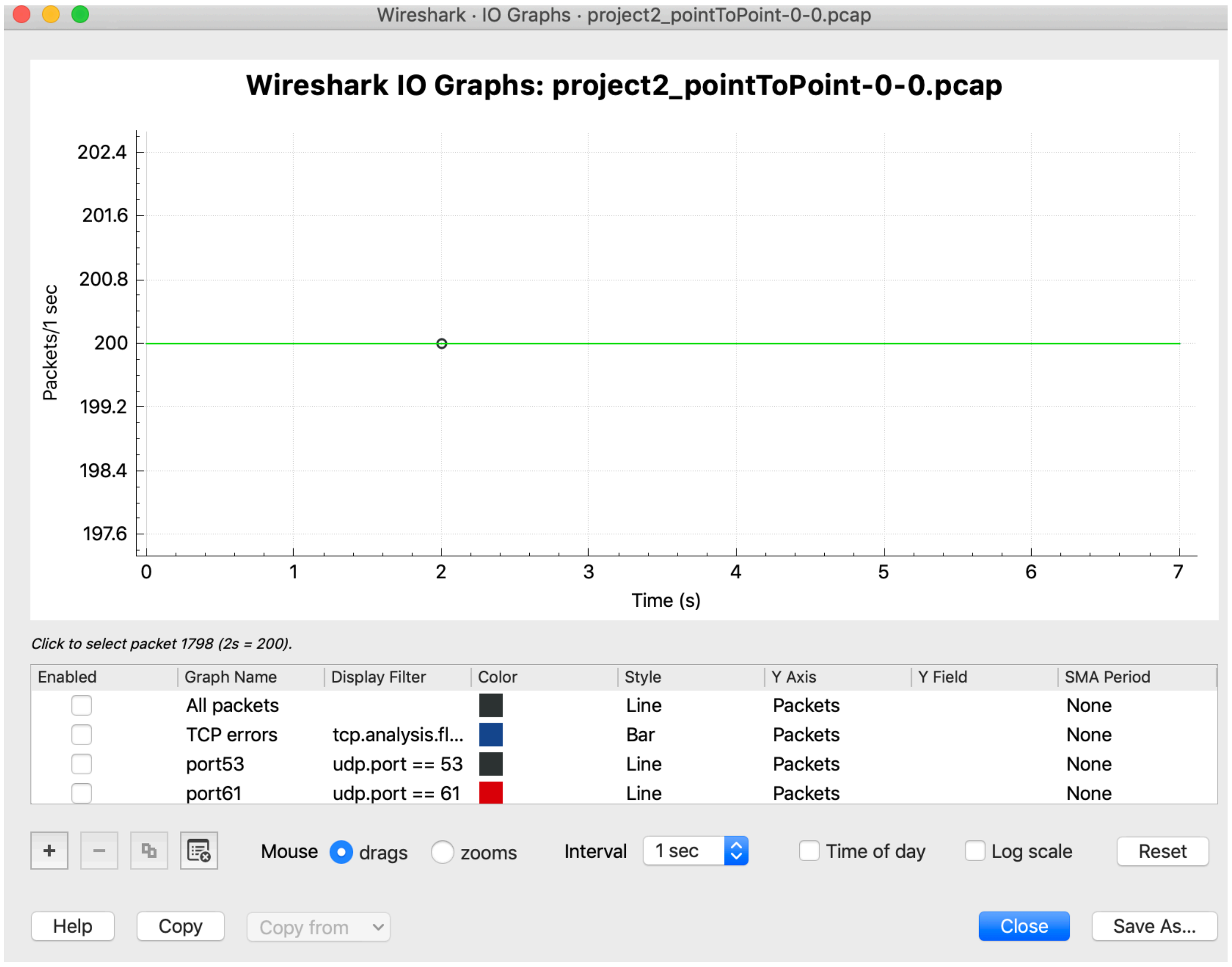
Mouse [✓ drags] zooms Interval 1 sec ☐ Time of day ☐ Log scale Reset

FIGURE2. Throughput vs. time after the SPQ node


Wireshark · IO Graphs · project2_pointToPoint-2-0.pcap
Wireshark IO Graphs: project2_pointToPoint-2-0.pcap
Packets/1 sec
Time (s)
Hover over the graph for details.
Mouse
drags
zooms
Interval
1 sec
Time of day
Log scale
Reset

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	All packets		⬛	Line	Packets		None
☐	TCP errors	tcp.analysis.fl...	🟦	Bar	Packets		None
✓	port53	udp.port == 53	⬛	Line	Packets		None
✓	port61	udp.port == 61	🟥	Line	Packets		None

FIGURE3. Throughput vs. time before the DRR node

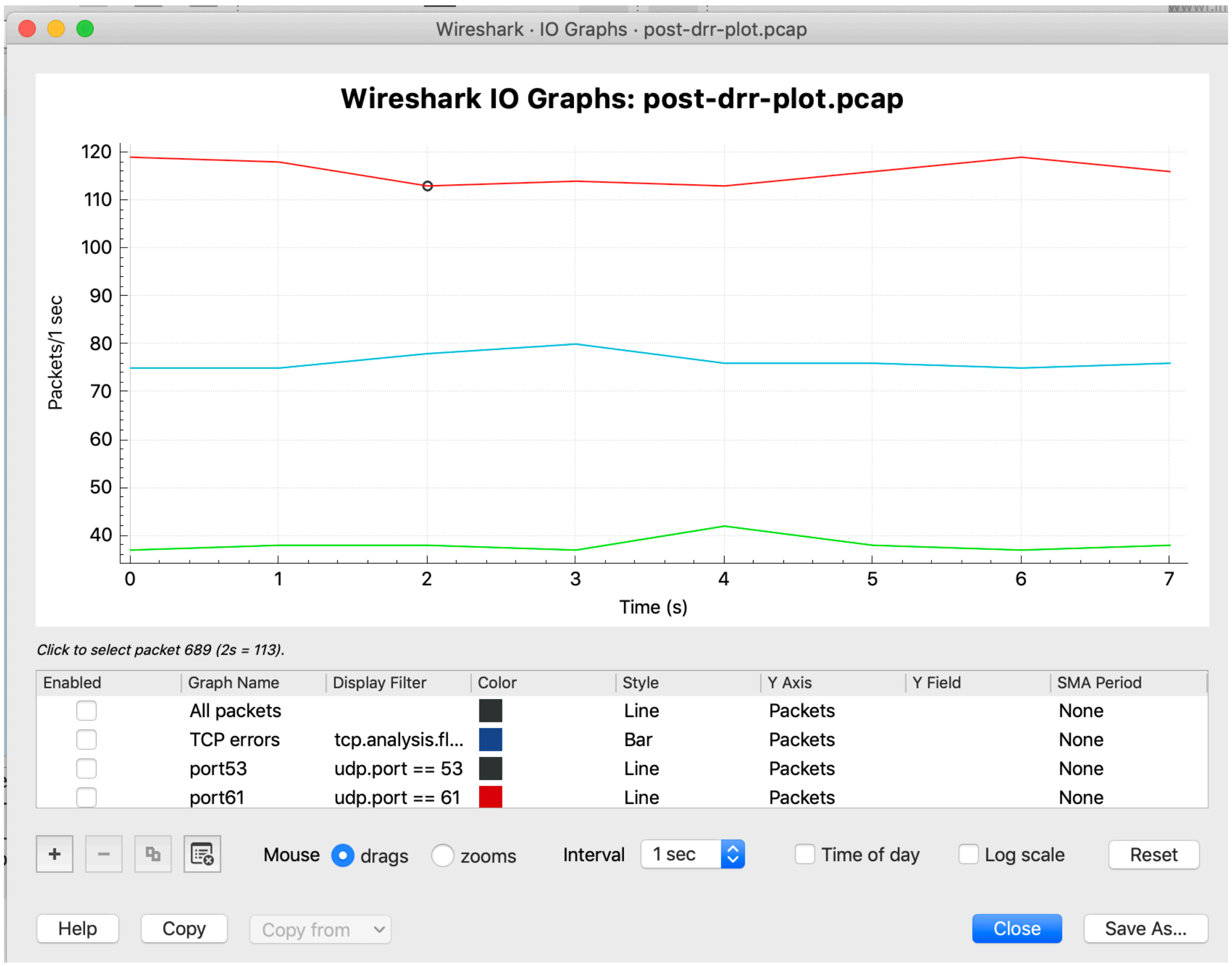


Click to select packet 1798 (2s = 200).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	All packets		⬛	Line	Packets		None
☐	TCP errors	tcp.analysis.fl...	🟦	Bar	Packets		None
☐	port53	udp.port == 53	⬛	Line	Packets		None
☐	port61	udp.port == 61	🟥	Line	Packets		None

Mouse [✓ drags] zooms

FIGURE4. Throughput vs. time after the DRR node


Wireshark · IO Graphs · post-drr-plot.pcap
Wireshark IO Graphs: post-drr-plot.pcap
Packets/1 sec
120
110
100
90
80
70
60
50
40
0
1
2
3
4
5
6
7
Time (s)

Click to select packet 689 (2s = 113).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	All packets		⬛	Line	Packets		None
☐	TCP errors	tcp.analysis.fl...	🟦	Bar	Packets		None
☐	port53	udp.port == 53	⬛	Line	Packets		None
☐	port61	udp.port == 61	🟥	Line	Packets		None

Mouse [✓ drags] zooms   Interval 1 sec   ☐ Time of day   ☐ Log scale