Advanced Networking 2020

Lab #1: TCP Congestion Control
V2
Total points: 20 pts

Assignment
Lab date: February 07, February 12, 2019
Submission date: 11:59PM February 13th 2019 CET

Author: Jamila Alsayed Kassem, Sara Shakeri, Peter Prjevara , Vincent Breider and Paola Grosso

Email: j.alsayedkassem@uva.nl, s.shakeri@uva.nl peter@os3.nl, vincent@os3.nl, p.grosso@uva.nl
University of Amsterdam

Introduction

This assignment helps to learn more about congestion control in the TCP protocol through emulation and simulation of various TCP implementations. In this lab you will work in pairs. Choose a colleague to work with.

Related documents

The material covered in the lecture gives you a good starting point. However, we do expect that you fill your knowledge gaps with information available on the internet and RFCs. Here is an incomplete list to get you started.

- ns-3 manual: `http://www.nsnam.org/doxygen/index.html`
- ns-3 tutorial: `http://www.nsnam.org/docs/release/3.27/tutorial/ns-3-tutorial.pdf`
- gnuplot reference card: `http://www.gnuplot.info/docs_4.0/gpcard.pdf`
- `google.com` - USE IT :-)

Submission

For lab 1, the submission consists of 3 items:

1. **A single report in PDF format**. The report should contain all answers with motivation and correctly labeled plots. **Give complete answers**: When asked to motivate or discuss your answer, we expect you to relate it to the theory (the tcp state machine etc), your figures, experiences during the lab, and some reasoning when you discover mismatches .

   `lab1-report-\${groupnumber}.pdf`

2. The **.tex source files** from which you generated the pdf:

   `lab1-source-\${groupnumber}.tar.gz`

3. An archive file (.tar.gz or .zip) for all source codes, simulation/emulator outputs and plots. The file name must be:

   `lab1-data-{groupnumber}.tar.gz`

Any other kind of submission will not be taken into account.

Task 1: TCP congestion simulation (10 points)

Task1 is about network simulation; you will use the NS3 simulator to simulate data transfers using congestion algorithms within different scenarios.

Requirements for Your Submission of Task1

- Use *gnuplot* to make your plots.

  Note: if you want to use *gnuplot* for plotting graphs from a data log, you can refer to example in the `plot-lab1-task1.gnu` script.

- Use the supplied tex template to write a small report.
- Provide plots and raw data logs.
- The file names (for the 3 scenarios) must be named (lowercase):

  `lab1-group{groupnumber}-task1-question{questionnumber}.[png,jpg,log]`

  e.g.

  `lab1-group6-task1-question1.1.log`

  or

  `lab1-group6-task1-question1.1-xrange-0-5.png`

  or

  `lab1-group6-task1-experimentA-question1.1.png`

Preparation

Hint: Virtualbox will not run if your XEN hypervisor is enabled. You either have to disable the XEN hypervisor via grub or re-load the kernel module. If you have UEFI enabled, you will have to enroll the key of Virtualbox in the MOK Manager.

Execute the following to install the Vagrant environment and to load the virtualbox kernel modules:

```
sudo apt install vagrant virtualbox wget

sudo modprobe vboxdrv vboxnetadp vboxnetflt vboxpci
```

Download and setup the Vagrant box and the AN tasks: The vagrant boxes are based on Ubuntu 16.04; you can become root using sudo and install packages if necessary.

```
wget http://derby.studlab.os3.nl/an2020-lab1.tar.gz
```

Extract the ns327.box file and run the following commands:

```
vagrant box add ns327.box --name ns327

tar xfvz an-lab1.tgz

cd an-lab1/task1

vagrant up

vagrant ssh
```

To test the simulator go to the ns3 installation directory inside the Vagrant box:

```
cd ns-allinone-3.27/ns-3.27/

./waf --run hello-simulator
```

You should now be ready to continue the Task1.

examples/tcp/tcp-variants-comparison.cc Read the code and try to understand the various options available when running the simulation.

You can run this simulation by issuing the command:

```
./waf --run tcp-variants-comparison --command-template="%s --tracing=1 --pcap_tracing=1 \
    --duration=100"
```

Note that by setting the *tracing* and *pcap_tracing* options to 1 (true) you will get a number of output files in your working directory in the Vagrant VM. Make sure you copy this data to your machine for further analysis.

Scenario 1: TcpNewReno

In this first scenario you will investigate the behaviour of the TCP New Reno implementation. To select the New Reno implementation you have to explicitly indicate this as command line option:

```
./waf --run tcp-variants-comparison --command-template="%s --tracing=1  --pcap_tracing=1 \
     --duration=400 --sack=true --prefix_name=TcpNewReno --transport_prot=TcpNewReno"
```

Hint: Inside the vagrant virtual machines there are /vagrant directories that are linked to the project directory on the host machine, an-lab1/task1, or an-lab1/task2. That will help you to share data between the host/Virtual machines.

Event	Time (s)	Current CWND (bytes)	Next CWND(bytes)	FlightSize	SSTH	New State
...						
...						
...						

Table 1: TCP Congestion Control

- Q1.1 Plot a graph showing SSTH versus time from 0.0s to 400.0 , then zoom in and plot again from 0.0s to 50s and discuss what you see.
- Q1.2 Plot a graph showing INFLIGHT versus time from 0.0s to 400.0 and discuss what you see.
- Q1.3 Plot a graph showing CWND versus time from 0.0s to 400.0s and discuss what you see.
- Q1.4 Find the points where the slow-start, congestion-avoidance, fast retransmit/fast recovery states begin. Then use that to fill the following table showing CWND changes (table 1). This will allow you to identify where state-change points occurred and fill in following fields in the table below: the time, CWND before and after change, Inflight, SSThreshold, the new state, and the event that caused the change. Provide also the initial states.

Scenario 2: TCP performance

The aim of this section is to simulate the TCP performance with varying parameters.

Experiment A

Replicate scenario 1 of the simulation with SACK Off: **Sack=0**.

- Q1.5 Answer again questions Q1.1-Q1.4 and compare the two results.

Experiment B

Replicate scenario 1 of the simulation and answer the questions: **Packet loss of 4%, Sack=0 and transfer duration of 400sec**.

- Q1.6 Answer again questions Q1.1-Q1.4 and compare the two results.

Experiment C

Replicate scenario 1 of the simulation: **Delay of 300 ms, Sack=true and transfer duration of 400sec**.

- Q1.7 Answer again questions Q1.1-Q1.4 and compare the two results.

Task 2: Emulation of TCP Congestion Window (10 pts)

Task2 is about network emulation; you will use Netem and the traffic control (*tc*) facilities on the linux host to emulate different network circumstances, change congestion algorithms and analyze them.

Requirements for Your Submission of Task2

- Use *gnuplot* to make your plots.
- Use the supplied tex template to write a small report.
- The file names (for the 3 scenarios) must be named (lowercase):

  `lab1-group${groupnumber}-task2-question${questionnumber}.[png,jpg,log]`

  e.g.

  `lab1-group6-task2-question1.1.log`

  or when a range of a plot is requested:

  `lab1-group6-task2-question1.1-xrange-0-5.png`

Preparation

Start the Vagrant Boxes for task2.

```
cd an-lab1/task2
```

```
vagrant up
```

try basic iperf and open two terminals and in the an-lab1/task2 directory.
On one terminal, start the iperf server in the sink box.

```
vagrant ssh sink
```

```
iperf -s
```

On the other terminal, start the iperf client in the source box.

```
vagrant ssh source
```

```
iperf -c sink
```

Now that the environment is set up, you need to familiarise yourself with the necessary tools to complete this lab:

1. "TCP probe": In our experiments you can used tcp_probe along with iperf to record the state of TCP connection in response to incoming packet. You can read the output of tcp_probe from `/proc/net/tcpprobe` using `dd if=/proc/net/tcpprobe ibs=128 obs=128`. The last output line may be truncated; remove this before plotting.

   Tcp probe can be loaded as a kernel module:

   ```
   modprobe tcp_probe port=5001 full=1
   ```

   The module generates the most interesting data at the sender side. In order to make the lab not affected by other factors, please remove the tcp probe module every time before starting a new emulation using `modprobe -r tcp_probe` and then add it again.

2. "IPerf(2)": Iperf is tool which you can use to generate the network traffic. The advantage of using iperf is that it allows us to tune various network parameters. Iperf reports bandwidth jitter and loss (In case of TCP transmission). Iperf has to be run at server and client side, also iperf has its own TCP version (it is not dependent on TCP version of the underlying system). This can be selected by -Z flag (e.g. Iperf -c -Z cubic).

   To make sure that the system you use is in sync with the experimental set up, you must select the same TCP congestion algorithm at system level as used with iperf. You can use the command "cat /proc/sys/net/ipv4/tcp_available_congestion_control" to know how many congestion algorithm the machine can support. Use the command (echo "cubic" > /proc/sys/net/ipv4/tcp_congestion_control) to change the congestion algorithm for the entire machine.

3. "Netem": Netem is a command line network emulator for testing network protocols. You can use this emulator in combination with traffic control (tc) tool to set network parameters like delay and loss by choosing the appropriate interface device.

   Possible command: "tc qdisc add dev eth1337 root handle 1:0 netem delay 100ms loss 2%".

4. "gnuplot": You can use the sample `plot-lab1-task2.gnu` to make a graph of time in seconds in x-axis versus segments (cwnd, sshtresh) on y-axis.

   Note: Before plotting the congestion window, please remove the last line of the data file you get. Because sometime the last line is not complete, it may cause error when plotting.

Scenario 1: Reno, low latency

Configure the machine to use the congestion algorithm of "TCP reno" and use "iperf" to emulate a TCP flow for 200 seconds. In this scenario, set the latency of this link to 30ms and the packet loss rate as 0%.

- Q2.1 Plot a graph showing the CWND and ssthresh versus time with all the data you get. These two metrics are in one graph.
- Q2.2 Briefly discuss the main changes in the graph, motivate your answer.

Scenario 2: Reno high latency

Use the congestion algorithm of "TCP reno" and use "iperf" to emulate a TCP flow for 200 seconds. Here, set the latency to 300ms and the packet loss rate as 0%.

- Q2.3 Plot a graph showing CWND versus time with all the data you get.
- Q2.4 Compare this graph with the one from Q2.1, show the difference between these two graphs and discuss why these occur.

Scenario 3: Cubic, low latency

Use the congestion algorithm of "TCP cubic" and use "iperf" to emulate a TCP flow for 200 seconds. Here, set the latency to 30ms and the packet loss rate as 0%.

- Q2.5 Plot a graph showing CWND and ssthresh versus time with all the data you get.
- Q2.6 Compare this graph with the graph of Q2.1 and discuss the major differences.

Scenario 4: Cubic high latency

Use the congestion algorithm of "TCP cubic" and use "iperf" to emulate a TCP flow for 300 seconds. Here, set the latency to 250ms and the packet loss rate as 0%.

- Q2.7 Plot a graph showing CWND versus time with all the data you get.
- Q2.8 Compare this graph with the one from Q2.3, show the difference between these two graphs and discuss why these occur.