S22 18731 HW2: Low-Rate TCP-Targeted DoS Attack

Instructor: Vyas Sekar

1 Introduction

In this project, your aim is to understand and replicate the low rate TCP DoS attack (a.k.a the shrew attack). The low rate TCP DoS attack consists of periodic, short, and carefully chosen duration bursts to exploit TCP's retransmission timeout. As a result, the retransmitted packets of other senders will be dropped. In contrast to other DoS attacks, low-rate TCP DoS attacks are particularly difficult to detect because of their carefully chosen bursts and durations.

2 Objective

The objectives of this assignment are to:

1. Familiarize yourself with Mininet and learn how to build simple topology using Mininet.
2. Understand and replicate how low rate TCP DoS attacks work.
3. Understand the attack parameters that play an important role for this attack.

It is critical that you read (Kuzmanovic and Knightly's SIGCOMM03 paper) to complete this assignment.

3 Submission

1. Your handin should include the following:
   (a) Your writeup: `andrewID.pdf`.
   (b) The code that initializes the topology and runs your experiment: `dumbbell.py`.
   (c) From part #1: `run_part1.sh`, `part1_task3_hl1_bw.png` and `part1_task3_bw.txt`.
   (d) From part #2: `run_part2.sh`, `part2_hl1_bw.png`, `part2_tcp_cwnd_iperf1.png`, `part2_a1_bw.png`, `shrew_client.sh`, and `shrew_server.sh`.
   (e) From part #3: `part3_task2_hl1_bw.png`, `part3_task2_tcp_cwnd_iperf1.png`, and `part3_task2_a1_bw.png`.
2. To submit your code, please create a repository and put files into your repository. To tell us your repository, please fill in this form.
3. Please provide access to the instructor and the course TAs, so that we can clone your repository using `git` (See the submission guidelines on Blackboard). *Warning: if you do not follow the instructions and we cannot access your repository, you may not get a credit.*

4 Homework Description

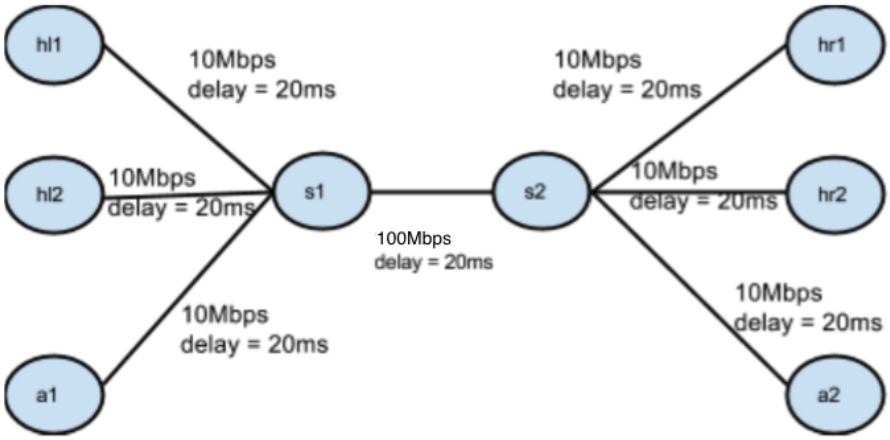
Please follow the following steps to get started with this homework:

1. Launch the Amazon Machine Image (AMI). The public AMI name is `18731_S22` (Make sure you search for the AMI in US West, Oregon). The username to log in to the AMI is `ubuntu`. Use your AWS key-pair to authenticate at login time. Consult this link for details on how to launch EC2 instance from a public AMI.
2. In the home directory, go to 18731 directory and run:
   `git clone https://sniperyyc@bitbucket.org/sniperyyc/hw2-18731-s22.git`

4.1 Part #1: Set Up (30 points)

4.1.1 Task #1: Setup the dumbbell topology (10 points)

You will setup the topology shown in the figure below.



Your topology should have the following nodes:

Hosts: `h1`, `h2`, `hr1`, and `hr2`.

Attackers: `a1` and `a2`.

Switches: `s1` and `s2`.

Specifically, you should do the following in this part:

1. Add code in dumbbell.py that:
   (a) Sets up this topology.
   (b) Tests if all nodes are reachable.

   This resource may help: Introduction to Mininet.
2. Run `sudo ./run_part1.sh` to construct the topology and test if all nodes are reachable. You should modify the supplied `run.sh` to write your `run_part1.sh`.
3. Type `exit` in Mininet console once done.

The output of running `sudo ./run_part1.sh` should look like:

```
mininet@mininetvm:~/HW3$ sudo ./run_part1.sh
start shrew experiment
net.ipv4.tcp_congestion_control = reno
net.ipv4.tcp_min_tso_segs = 1
starting mininet ....
a1 a1eth0:s1eth4
a2 a2eth0:s2eth4
hl1 hl1eth0:s1eth2
hl2 hl2eth0:s1eth3
hr1 hr1eth0:s2eth2
hr2 hr2eth0:s2eth3
*** Ping:  testing ping reachability
a1 -> a2 hl1 hl2 hr1 hr2
a2 -> a1 hl1 hl2 hr1 hr2
hl1 -> a1 a2 hl2 hr1 hr2
hl2 -> a1 a2 hl1 hr1 hr2
hr1 -> a1 a2 hl1 hl2 hr2
hr2 -> a1 a2 hl1 hl2 hr1
*** Results:  0% dropped (30/30 received)
mininet > exit
cleaning up...
ping:  no process found end
```

4.1.2 Task #2: Set up longlived TCP connection (10 points)

Now add the code in `dumbbell.py` to setup two `iperf` servers on the left and two `iperf` clients on the right (i.e., `hl1` and `hl2` are `iperf` servers and `hr1` and `hr2` `iperf` clients).

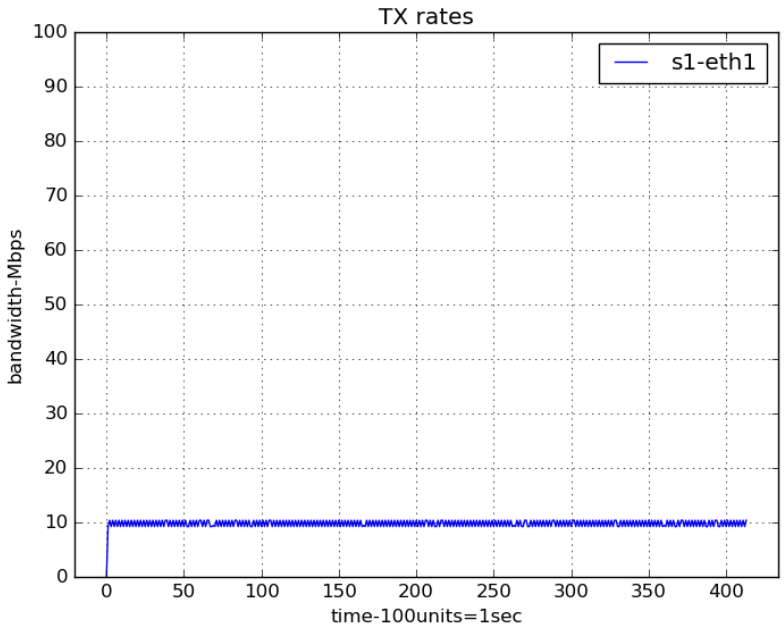
`iperf` servers: `hl1`, `hl2`.
`iperf` clients: `hr1`, `hr2`.

`hl1` connects to `hr1` via port 5001 and `hl2` connect to `hr2` via port 5002 (`p` option in `iperf`).

4.1.3 Task #3: Plot throughout and congestion window before attack (10 points)

1. Run `sudo ./run_part1.sh` to start the network.
2. On a different terminal, run sudo `./monitor.sh` <exp_name> where <exp_name>= `part1_task3` to monitor the throughput in the network. (i.e., You should ssh again to get a different terminal).
   (a) After the desired seconds press return to stop monitoring.
   (b) Run `./plot_figures.sh` <exp_name> to generate figures.
   (c) To view the figures, use `scp` to copy the figures to your local machine.

The throughput for `h1` (`part1_task3_h1_bw.png`) should look like (around 5-10Mbps):



4.2 Part #2: Implement the Shrew Attack (40 pts)

4.2.1 Task #1: Implement the attack (30 points)

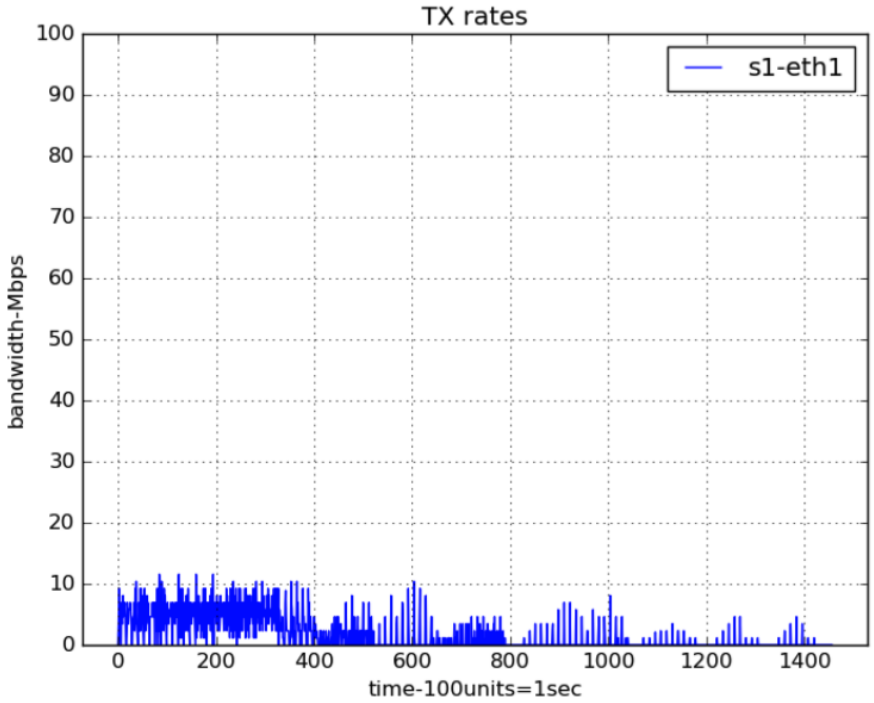
In this part 2, you will write code that simulates low-rate DoS attacks that are run on `a1` and `a2`. We let attacker use UDP traffic instead of TCP (you can use `u` flag in `iperf`).

You also need to make changes to your `dumbbell.py` to change appropriate parameters (Hint: set max queue length to 20. Note that the default value from previous part was 1,000. To launch a low-rate DoS attack, you need to reduce the queue length in order to overflow the buffer).

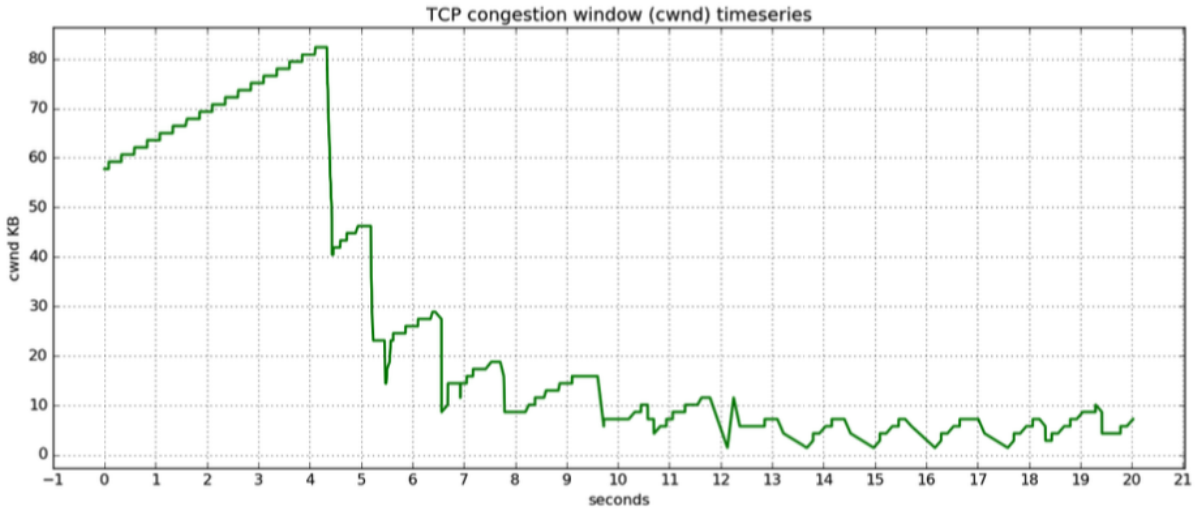
1. From the paper, you can see that a suggested lower bound on RTO is 1s. We have modified the Linux kernel of the provided AMI to support a minRTO of 1s (Nothing to do here).
2. Write `shrew_server.sh` for `a1` and `shrew_client.sh` for `a2`.
   (a) `shrew_server.sh` sets up the `iperf` server.
   (b) `shrew_client.sh` implements the attack.
3. To run the attack:
   (a) Run `sudo ./run_part2.sh`.
   (b) `sudo ./monitor.sh` <exp_name> where <exp_name>="part2" (in new terminal).
   (c) In Mininet (where you ran `./run.sh`) run: `a1 sudo ./shrew_server.sh`.
   (d) In Mininet run: `a2 sudo ./shrew_client.sh`.
   (e) After desired period of time, press enter to stop monitoring.
   (f) `sudo ./plot_figures.sh` <exp_name>. View figures using `scp`.

4. Plot throughput and congestion window. You should have gotten figures from running `./plot_figures.sh` command. You should also see reduced throughput from the first burst.

You should see around 50% to 60% reduction in throughput for `h1`. I.e., The plot in `part2_h1_bw.png`) should look like this (for this sample output, the first burst attack started around 500ms):



The congestion window (`part2_tcp_cwnd_iperf1.png`) should look like:



Notes:

1. Your plots don't need to be identical to the ones shown here, but they should show similar trends.
2. We will check if you are implementing low-rate DoS (not regular DoS) correctly.

4.2.2 Task #2: Attack details (10 points)

What are the parameters used for each of the following and why? (max 1-2 sentences per point).

- Attack period.
- Burst duration.
- Burst rate.

Please answer in your writeup.

4.3 Part #3: Understand Attack Parameters (30 pts)

The goal of this part is to gain understanding of the attack parameters essential for a low-rate DoS attack to work. You will mathematically reason about why a certain value of attack parameter is needed and get to tweak a parameter to see its impact on the attack you implemented in Part #2.

4.3.1 Task #1: Understanding the attack period (10 points)

Why did you choose that attack period from part #2? (Trial and error is not a valid answer. Hint: Equation 2 of the paper).

Please answer in your writeup.

4.3.2 Task #2: Increase buffer size (20 points)

Now, in your dumbbell code, increase max queue size (buffer size) to 1000. Other attack parameters should be the same as part 2. When launching the attack, your <exp_name>= "part3_task2".

Plot the throughput and the congestion window. Please answer in your writeup: Is the attack successful, explain why? If not, explain why?