HW2 Shrew Attack

Hao Zhang
Haozhan2
Network Security

Part #1 Set Up

1. Setup dumbbell topology

   I first setup the topology as described in the handout with code below. Note that for each host added, the bandwidth, delay and max_queue_size is parsed through argument, which I set through run_part1.sh.

```
class DumbbellTopo(Topo):
    "Dumbbell topology for Shrew experiment"
    def build(self, n=6, bw_net=100, delay='20ms', bw_host=10, maxq=None):
    #TODO: Add your code to create topology

        hl1 = self.addHost( 'hl1' )
        hl2 = self.addHost( 'hl2' )
        a1 = self.addHost( 'a1' )

        hr1 = self.addHost( 'hr1' )
        hr2 = self.addHost( 'hr2' )
        a2 = self.addHost( 'a2' )

        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')

        # Add links
        self.addLink( hl1, s1, bw=bw_host, delay = delay, max_queue_size = maxq)
        self.addLink( hl2, s1, bw=bw_host, delay = delay, max_queue_size = maxq)
        self.addLink( a1, s1, bw=bw_host, delay = delay, max_queue_size = maxq)

        self.addLink( hr1, s2, bw=bw_host, delay = delay, max_queue_size = maxq)
        self.addLink( hr2, s2, bw=bw_host, delay = delay, max_queue_size = maxq)
        self.addLink( a2, s2, bw=bw_host, delay = delay, max_queue_size = maxq)

        self.addLink( s1, s2,bw=bw_net, delay = delay )
```

Once the topology is successfully created, I start the network and test reachability between each nodes with net.Pingall().

```
topo = DumbbellTopo(n=args.n, bw_net=args.bw_net,
                    delay='%sms' % (args.delay),
                    bw_host=args.bw_host, maxq=int(args.maxq))

net = Mininet(topo=topo, host=CPULimitedHost, link=TCLink,
                  autoPinCpus=True)
net.start()
dumpNodeConnections(net.hosts)

#TODO: Add your code to test reachability of hosts
net.pingAll()
```

The output on terminal is shown as below. As you can see, each node is able to reach every other node.

2. Setup longlived TCP

I initially write this two lines of codes in dumbbell.py.

```
'''hl1, hr1 = net.get('hl1','hr1')
iperf((hl1,hr1), port=5001)

hl2, hr2 = net.get('hl2','hr2')
iperf((hl2,hr2), port=5002)
'''
```

Running them in terminal will show below iperf statistics. As the bandwidth for host-switch is 10 Mbits/s, they are what we expect. However, for task 3, net.iperf doesn't seem to create traffic that can be properly captured by monitor.sh. Therefore, I switch it to cmd in task 3.

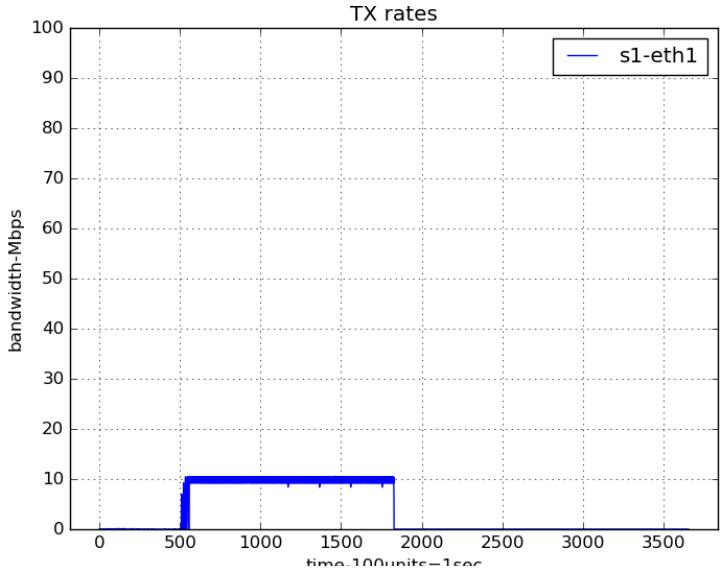
```
*** Results: 0% dropped (30/30 received)
*** Iperf: testing TCP bandwidth between hl1 and hr1
*** Results: ['9.20 Mbits/sec', '15.0 Mbits/sec']
*** Iperf: testing TCP bandwidth between hl2 and hr2
*** Results: ['9.18 Mbits/sec', '14.7 Mbits/sec']
```

3. Plot throughout and congestion window before attack

As foreshadowed by task 2, I switch the net.iperf to iperf in dumbbell.py as below. (a1 and a2 are there for part 2). With this code, hr1 and hr2 will constantly send traffic to hl1 and hl2 in order to measure the network performance. Note that & is used so the upper command won't block the lower command (polling mode).

```
hl1,hl2,hr1,hr2,a1,a2 = net.get('hl1', 'hl2','hr1', 'hr2','a1','a2')
hl1.cmdPrint('iperf -s -p 5001 -t 1000 &')
hl2.cmdPrint('iperf -s -p 5002 -t 1000 &')
a1.cmdPrint('iperf -s -p 5003 -t 1000 &')

hr1.cmdPrint('iperf -c 10.0.0.3 -p 5001 -i 1 -t 1000 &')
hr2.cmdPrint('iperf -c 10.0.0.4 -p 5002 -i 1 -t 1000 &')
a2.cmdPrint('iperf -c 10.0.0.1 -p 5003 -i 1 -t 1000 &')
```

After using monitor.sh and plot_figure.sh. I got this figure below. This is what we expect, as the link between s1 and hl1 has bandwidth of 10Mbps. Only 10Mbps of traffic can reach hl1, so that's what we see in the graph, hl1 is getting bandwidth of 10 Mbps.



Part #2 Implement the Shrew Attack

Task#1 Implement the attack

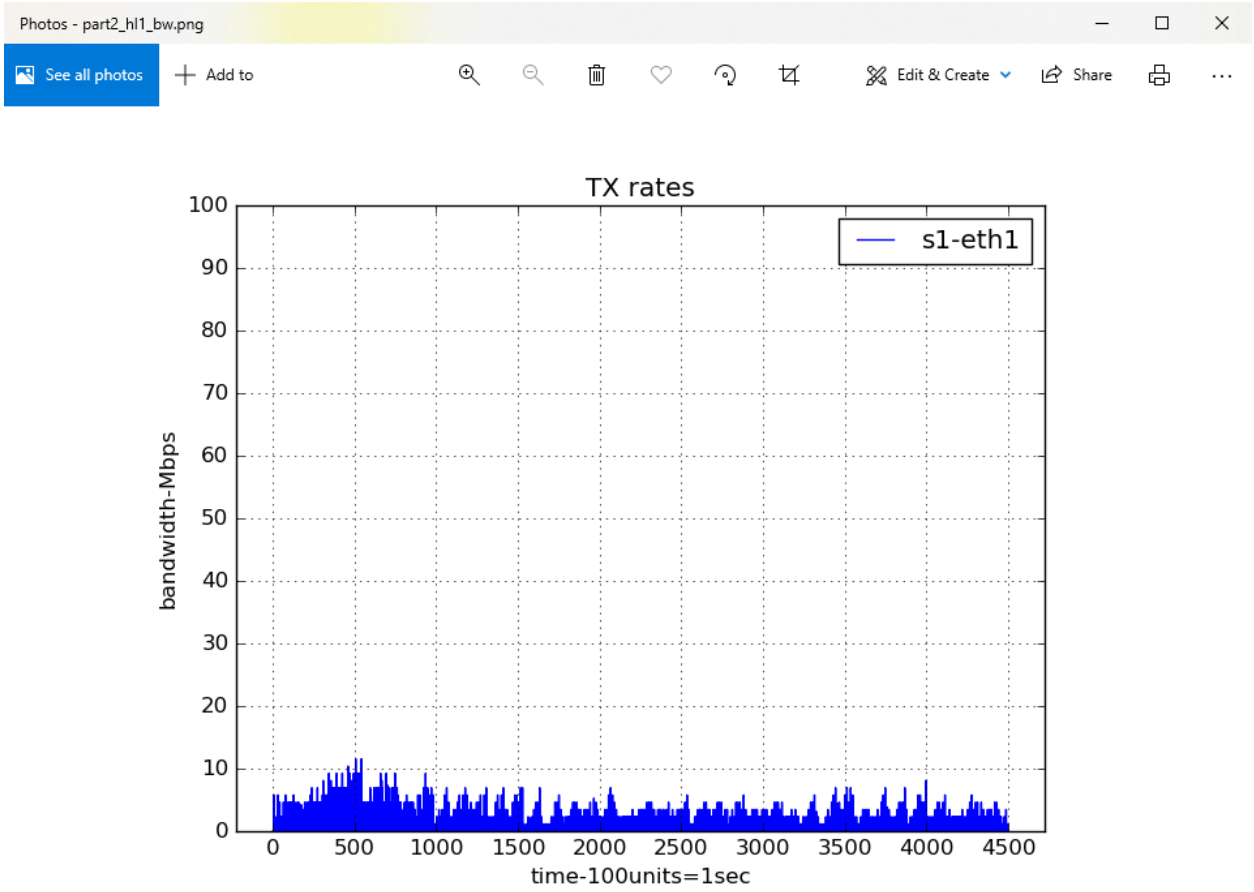
The attack is carried out from a2 to a1 using iperf. A1 runs a iperf server (udp) and A2 runs a iperf client. A1 would run bash script as below. It is listening on port 5006 waiting for a2 to connect.

```
echo set iperf-server

iperf -s -u -p 5006 -t 400

killall -9 iperf ping
echo "end Server"
```

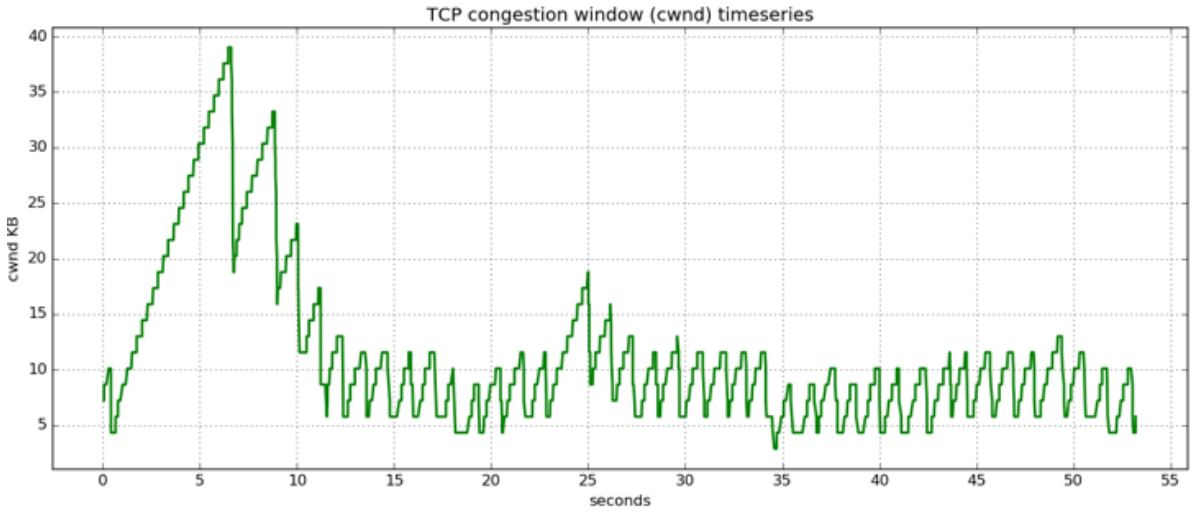
A2 will launch the attack with code as below. It will launch an attack that lasts 0.25s and sleep for 0.75s (a period of 1s). This attack will go over and over for 200 times. It will target a1 port 5006 and the burst rate is 10MB.
Note, for this attack to happen, max_queue_size for the link between switch 1 and switch 2 is set to 20.

```
echo running iperf-client

for i in {1..200}
do
    iperf -u -c 10.0.0.1 -b 10M -p 5006 -t 0.25
    sleep 0.75
done


killall -9 iperf ping
echo "end Client"
```

The throughput on hl1 is shown below. Compared to part 1 which has its bandwidth fixed at 10, the bandwidth now is constantly below half of its original throughput, which means the attack is effective (hl1 can no longer communicate normally with hr1).



The congestion window also reflect similar truth that it is generally kept below 10kB of congestion window after attack.



Task #2 Attack details

Some of the parameters are set for this shrew attack.

The attack period is set to 1s (0.25 burst duration + 0.75s of sleep in the code). This is calculated with formula listed in shrew attack paper.

$$\rho(T) = \frac{\lceil \frac{minRTO}{T} \rceil T - minRTO}{\lceil \frac{minRTO}{T} \rceil T}. \qquad (2)$$

I originally calculated T as 2, but when I tested it out for attack, it did not restrain hl1 to limited throughput. Therefore, I set T to 1. Now, If we plug in minRTO as 1 and T as 1, we will find throughput as 0, since the theory could slightly deviate from the reality, we still see some traffic as shown in task 1 of part 2.

Burst duration: 0.25s. As hinted in the paper, duration l' should be greater than RTT so the original packet between hl1 and hr1 will be dropped. RTT in the topology is 120ms. I wish to make it safe, so I set it around 2RTT.

Burst rate: 10MB. a2's host link to switch 2 is limited to 10MB, so any traffic greater than 10 MB is pointless. I used 10MB as it gives greater chance of filling the queue of s2, and therefore drop the packet for hl1.

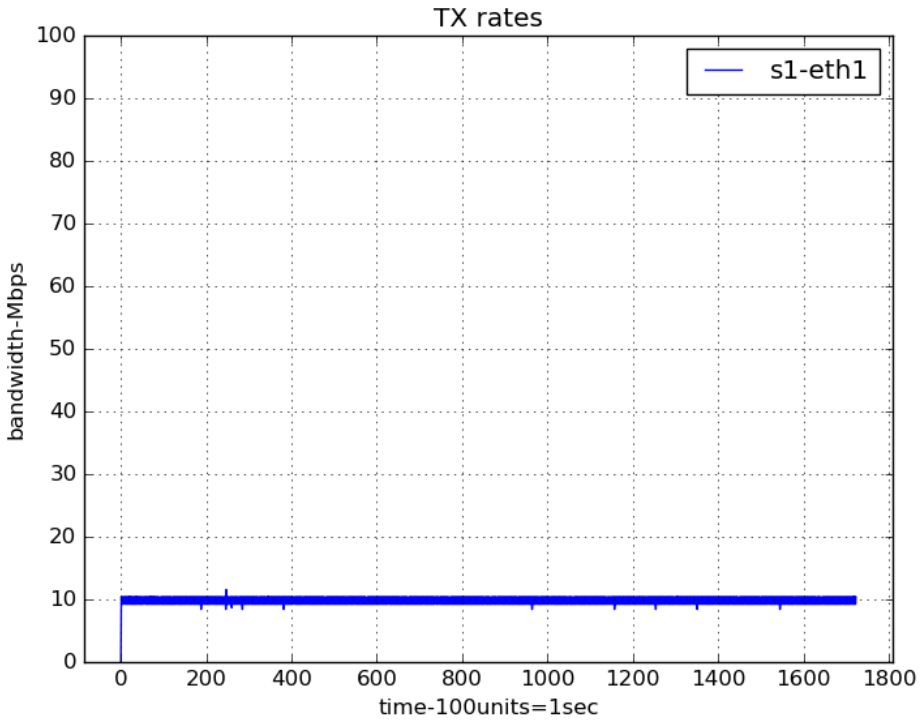
Part #3 Understand Attack Parameters

Task #1 Understanding the attack period

As discussed in piazza, I have incorporated it in part 2 task 2.

Task #2 Increase Buffer Size

Increasing queue size to 1000, the throughput hl1 is as below.



We sees that hl1 is not affected by shrew attack at all. The reasoning is as follow. A2 can at most send out 10MB of flooding. Suppose hr1 is also sending out 10MB of traffic. The bandwidth between s1 and s2 is 100MB, far from being drained. The reason that the attack is successful in part 2 is that the queue size for s2 is only 20, so if packets from a2 can preoccupied the queues, then packets from hr1 has to be dropped. Now that the switch has enough queue size, packets for hr1 will no longer be dropped. Hence hr1 will safely transmit its packet to hl1 and no loss will be detected.

For the same reason, hl1 believes the channel is not congested at all and keeps increasing its congestion window as shown below.