Network Simulation

Cody Heffner

22 Jan. 2015

1 Preface

This report details the experiment I ran and the results obtained as specified by the Network Simulation Lab in the BYU CS 460 class taught by Dr. Zappala. The project specifications can be found here.

The experiment requires heavy use of a network simulator to test different network scenarios. The network simulator I used is Dr. Zappala's Bene, written in Python. All my simulation examples shown will be tailored towards this simulator.

2 Summary

The goal of the experiment was to test various network scenarios by sending packets across networks of diverse bandwidth and distances, then observing the delays incurred by the transmission, propogation, and queueing of those packets in the network. The next section describes the experiment I ran with a simple two-node network and one bi-directional link set to various bandwidths and lengths. The following section reports the experiment I ran with a three-node network and two bi-directional links. The section after that describes the portion of the experiment that was used to validate queueing theory with regards to an M/D/1 queue on a network. The final section summarizes the experiment as a whole and discusses some things I did wrong during the experiment that gave me deeper insight into how the internet works.

3 Two Nodes

The network I created for this part of the experiment was a simple network consisting of two nodes and one bi-directional link. The following scenarios were tested:

1. One packet of length 1000 bytes sent from node n1 to node n2 at time 0. The network's bandwidth was 1Mbps with a propagation delay of 1 second.

2. One packet of length 1000 bytes sent from node n1 to node n2 at time 0. The network's bandwidth was 100bps with a propagation delay of 10 ms.

3. Three packets of length 1000 bytes sent from node n1 to node n2 at time 0, then one more packet sent at time 2. The network's bandwidth was 1Mbps with a propagation delay of 10 ms.

The following two code blocks show how I set up the first scenario with one packet sent on a 1Mbps link and a propagation delay of 1. (Network setup for the other two scenarios are nearly exactly identical so I have omitted the code from this report.) The first block of code is a Python snippet interacting with the Bene simulator. Line 3 builds a Network object as described in the file 2n1.txt, which is shown in the second code block below. Lines 6 and 7 of sim.py simply retrieve nodes for later use. Lines 8 and 9 add a forwarding entry in its forwarding tables for the other node. For example on line 8 the function *add_forwarding_entry* takes the parameters *address* and *link*. *Address* can be thought of as the receiving

IP address n2 has designated to receive packets from n1; thus, the code needs to retrieve that address from n2's attributes. The parameter *link* represents the physical line going out from n1 to n2. In this case it is the first and only link enumerated within n1's *link* array. The forwarding entry is added to n1's forwarding tables.

```
1 # sim.py
2     # setup network
3     net = Network('networks/2n1.txt')
4
5     # setup routes
6     n1 = net.get_node('n1')
7     n2 = net.get_node('n2')
8     n1.add_forwarding_entry(address=n2.get_address('n1'),link=n1.links[0])
9     n2.add_forwarding_entry(address=n1.get_address('n2'),link=n2.links[0])
10
11    # setup app
12    d = DelayHandler()
13    net.nodes['n2'].add_protocol(protocol="delay",handler=d)
```

The file 2n1.txt is relatively self-explanatory, but I'll go over it quickly. The first three lines are ignored. Lines 4 and 5 establish nodes and links between nodes. The first word represents a node to be created, and all following words on that same line represent a connection between the first word and each following word. Lines 8 and 9 configure links. The first two words target the start and end node of a link, and the last two words set that link to have a 1Mbps bandwidth and 1000ms (1 second) propagation delay.

```
1 # 2n1.txt
2     # n1 -- n2
3     #
4     n1 n2
5     n2 n1
6
7     # link configuration
8     n1 n2 1Mbps 1000ms
9     n2 n1 1Mbps 1000ms
```

The simulator simulates three types of delays that packets experience as they travel: transmission, propagation, and queueing.

Transmission delay is calculated as the result of the size of the packet in bits divided by the bandwidth of the link it is being transmitted onto. In scenario 1, transmission delay is the result of (8,000 bits / 1,000,000 bps) = 0.008s.

Propagation delay is simply a factor of the distance and speed the packet has to travel, but in this experiment, the propagation delays are given. In scenario 1, propagation delay = 1s

Queueing delay is how long a packet has to wait after it is fully received by a node. The cause of queueing delay is solely a factor of how many packets are in line at a node when a packet needs to be sent. In scenario 1, only one packet ever exists, so there will be no queueing delay.

The total delay is the sum of all three of those delays.

The following table shows each packet's identifier, the time it was created, and the time it was received.

ID	Time Created	Time Receieved
Scenario 1:		
1	0s	1.008s
Scenario 2:		
1	0s	80.01s
Scenario 3:		
1	0s	0.018s
2	0s	0.026s
3	0s	0.034s
4	2s	2.018s

Raw Data

The simulator can be shown as being accurate by comparing theoretical results with simulator results. The following tables compare the delay the packets experience in each situation as calculated by hand and the delay the simulator reported the packets experienced. In any scenarios with multiple packets, I will show the queueing delay as experienced by the packet with the largest queueing delay.

Delay Type	Calculated	Simulated
Transmission	0.008s	0.008s
Propagation	1s	1s
Queue	0s	0s
Total	1.008s	1.008s

Scenario 1

Delay Type	Calculated	Simulated
Transmission	80s	80s
Propagation	0.01s	0.01s
Queue	0s	0s
Total	80.01s	80.01s

Scenario 2

Delay Type	Calculated	Simulated
Transmission	0.008s	0.008s
Propagation	0.01s	0.01s
Queue	0.016s	0.016
Total	0.034s	0.034s

Scenario 3

In scenario 3, node n1 sends three packets back to back. N1 needs to place the third packet in a queue behind the other two while it transmits the first two. Therefore, the largest queueing delay will be experienced by the third packet and will be equal to two times the transmission delay for any packet.

As the three tables above show, the simulator is precise and accurate, finding each of the simulated delay values are equal to the calculated delay values.

4 Three Nodes

For the three-node portion of the experiment, I set up three separate networks. The following list enumerates the three scenarios for which I needed separate networks.

/beginenumerate

/item Two Fast Links, Part A: In a network consisting of three nodes A, B, and C, perform the transfer of a 1 MB file divided into 1000 1 kB packets from node A to C. The links include a link from A to B with 100 ms propagation delay and 1 Mbps bandwidth, and a link from B to C otherwise identical to the first link.

/item Two Fast Links, Part B: In a network consisting of three nodes A, B, and C, perform the transfer of a 1 MB file divided into 1000 1 kB packets from node A to C. The links include a link from A to B with 100 ms propagation delay and 1 Gbps bandwidth, and a link from B to C otherwise identical to the first link.

/item One Fast Link and One Slow Link: In a network consisting of three nodes A, B, and C, perform the transfer of a 1 MB file divided into 1000 1 kB packets from node A to C. The links include a link from A to B with 100 ms propagation delay and 1 Mbps bandwidth, and a link from B to C with 100 ms propagation delay and 256 Kbps bandwidth.

/endenumerate

By inspection, all these scenarios are nearly identical. I will only show the raw code for the first network. The following two code blocks show the .txt file describing the network to Bene and the Python file that executes the experiment.

```
# 3n1a.txt
    # n1 -- n2 -- n3
    #
    n1 n2
    n2 n1 n3
    n3 n2

    # link configuration
    n1 n2 1Mbps 100ms
    n2 n1 1Mbps 100ms
    n2 n3 1Mbps 100ms
    n3 n2 1Mbps 100ms
```

```
# sim.py
    # setup network
    net = Network('networks/3n1a.txt')

    # setup routes
    n1 = net.get_node('n1')
    n2 = net.get_node('n2')
    n3 = net.get_node('n3')
    n1.add_forwarding_entry(address=n2.get_address('n1'),link=n1.links[0])
    n1.add_forwarding_entry(address=n3.get_address('n2'),link=n1.links[0])
    n2.add_forwarding_entry(address=n1.get_address('n2'),link=n2.links[0])
    n2.add_forwarding_entry(address=n3.get_address('n2'),link=n2.links[1])
    n3.add_forwarding_entry(address=n2.get_address('n3'),link=n3.links[0])
    n3.add_forwarding_entry(address=n1.get_address('n2'),link=n3.links[0])

    # setup app
    d = DelayHandler()
    net.nodes['n1'].add_protocol(protocol="delay",handler=d)
    net.nodes['n2'].add_protocol(protocol="delay",handler=d)
    net.nodes['n3'].add_protocol(protocol="delay",handler=d)
```

Upon close inspection of sim.py, the network needs to be set up in a slightly different way than the two-node network. This network needed six different forwarding table entries compared with two entries in the two-node network. This is because there needs to be a forwarding table entry from *each* node to *every other* node. As you can see from the code, a forwarding table entry is added from n1 to n2 and to n3, then from n2 to n1 and to n3, then finally from n3 to n2 and to n1. The most interesting cases are from n1 to n3 and back. N1 doesn't have a direct link to n3, so it must go through n2. Thus, the forwarding table entry from n1 to n3 needs to specify that it needs to go to the address n3 has dedicated to receive from n2, and it needs to run on the first (and only) link n1 has - the link to n2.

5 Queueing Theory

6 Conclusion