ECE 358: Assignment 6

Ariel Weingarten - 20366892
Alexander Maguire - 20396195

July 7, 2014

1 Question One

For each packet sent, there is 40 bytes of overhead. 20 bytes come from the TCP header and the other 20 bytes come from the IP header. The MSS is 1000 bytes and does not include headers. Therefore each data packet sent consists of 1040 bytes. For each successfully received packet, an ACK packet is sent back. All of the data required for an ACK is contained in the header, so each ACK packet is 40 bytes.
It follows that for every 1000 bytes of application data sent there are 1080 bytes sent.

$$D_{efficiency} = \frac{1000}{1080} \approx 0.926$$

2 Question Two

Yes. Exponential backoff is a reative procedure for mitigating congestion, it does not prevent congestion. Consider the following example. You are sending a 10 GB file from Host A to Host B over a 100 MB link. Using only exponential backoff will result in packets being sent as fast as possible. Only when a congestion point is reached do you reap any benefit from exponential backoff. Dropped packets are not rapidly resent, which would add to the congestion. Though there is no policy to stop sending new packets (packets that have not been sent before). This means that the congestion will continue to worsen until all packets have had one attempted delivery and exponential backoff will start to manage the congestion levels.

3 Question Three

Above is a graphical representation of what is happening to our cwnd. Let W be the size of the window in bytes, MSS be the maximum segment size, and RTT be round trip time. For each successfully delivered packet, the cwnd increases by 1. This means that it takes $\frac{W/2}{MSS}$ RTTs to go from a cwnd size of $\frac{W/2}{MSS}$ to $\frac{W}{MSS}$. The total number of packets sent in one of these cycles (each "saw tooth") is equal to the area under the curve. This is easily approximated due to the geometry of the curve.

$$N_{packets} = \frac{W/2}{MSS} \cdot \frac{W/2}{MSS} + \frac{1}{2} \cdot \frac{W/2}{MSS} \cdot \frac{W/2}{MSS}$$

$$N_{packets} = \frac{3 \cdot W^2}{8 \cdot MSS^2}$$

$$Throughput_{packets/second} = \frac{data_{percycle}}{time_{percycle}}$$

$$Throughput_{packets/second} = \frac{\frac{3 \cdot W^2}{8 \cdot MSS^2}}{\frac{W \cdot RTT}{2 \cdot MSS}} = \frac{3 \cdot W}{4 \cdot RTT}$$

$$Throughput_{bytes/second} = Throughput_{packets/second} \cdot MSS$$

The above calculations do not take into account packet lost, but will be very helpful nonetheless. With a packet loss ratio of L, we know that the total number of continous, successfully sent packets sent will be equal to $\frac{1}{L}$ We can plug this into our previous equation for the number of packets per cycle.

$$\frac{1}{L} = \frac{3 \cdot W^2}{8 \cdot MSS^2}$$

Solving for W, we find that:

$$W = \sqrt{\frac{8}{3 \cdot L}} \cdot MSS$$

Plugging in this value of W into our Throughput equation we confirm that:

$$Throughput_{bytes/second} = \frac{3 \cdot \sqrt{\frac{8}{3 \cdot L}} \cdot MSS}{4 \cdot RTT} \approx \frac{1.22 \cdot MSS}{RTT \cdot \sqrt{L}}$$

4 Question Four

No. Addidtive increase gives a slope of 1 and so it follows that additive decrease gives a slow of -1. No progress towards equilibrium can be made without $slope < -1$.

5 Question Five

5.1 $4S/R > S/R + RTT > 2S/R$

$$\text{Let} P(x) = xR/S + RTT$$

$$\begin{aligned} t &= RTT + P(1) + P(2) + P(4) + P(8) \\ t &= 16S/R + 4RTT \end{aligned}$$

$$\begin{aligned} S/R + RTT &> 2S/R \\ RTT &> S/R \\ \therefore \quad t &> 20S/R \end{aligned}$$

$$\begin{aligned} 4S/R &> S/R + RTT \\ 3S/R &> RTT \\ \therefore \quad 28S/R &> t \end{aligned}$$

$$28S/R > t > 20S/R$$

5.2 $S/R + RTT > 4S/R$

$$t = 16S/R + 4RTT$$

$$\begin{aligned} S/R + RTT &> 4S/R \\ RTT &> 3S/R \\ \therefore \quad t &> 28S/R \end{aligned}$$

5.3 $S/R > RTT$

$$t = 16S/R + 4RTT$$

$$\begin{aligned} S/R &> RTT \\ \therefore \quad t &> 20RTT \end{aligned}$$

6 Question Six

6.1 Part A

Since there are 32 bits in an IP address, given a mask of 25 bits, we are left with 7. Since $2^7 >= 100$, this subnet will suffice for the department.

The given mask allows the last 7 bits to vary for the department, providing addresses in the range 129.97.44.0 → 129.97.44.127.

6.2 Part B

Because each group wants a separate subnet, each requires a $\lceil \log_2(h) \rceil$ bits where h is the number of hosts requested.

Group R1 thus requires 7 bits, R2 requires 5 and R3 requires 4. However, the total assignable space is only 7 bits, and thus group R1 must necessarily claim all of it – leaving no subnet available for R2 or R3.

6.3 Part C

R1 now needs 46 addresses – 45 for the publicly-addressable hosts, and one for NAT to address the other 25. In total they now need a mask of $\lceil \log_2 46 \rceil = 6bits$. In order to accommodate them, the university can assign the subnet 129.97.44.0/6, corresponding to the addresses 129.97.44.0 → 129.97.44.63.

To R2, the university now grants 129.97.44.64/5: giving the address range 129.97.44.64 → 129.96.44.95.

R3 correspondingly, is assigned 129.97.44.96/4; ie. the range 129.97.44.96 → 129.97.44.111.

7 Question Seven

7.1 Part a

Does this request go anywhere?

uhhh:

? → 129.97.3.79

aa:aa:aa:aa:aa:aa → ff:ff:ff:ff:ff:ff

7.2 Part b

C finds D:

129.97.3.5 → 129.97.3.10

cc:cc:cc:cc:cc:cc → ff:ff:ff:ff:ff:ff

D replies:

129.97.3.10 → 129.97.3.5

dd:dd:dd:dd:dd:dd → cc:cc:cc:cc:cc:cc

7.3 Part c

B finds R1:

129.97.2.2 → 129.97.3.67

bb:bb:bb:bb:bb:bb → ff:ff:ff:ff:ff:ff

R1 replies:

129.97.3.67 → 129.97.2.2

01:01:01:01:01:01 → bb:bb:bb:bb:bb:bb

B sends message via R1:

129.97.2.2 → 129.97.3.5

bb:bb:bb:bb:bb:bb → 01:01:01:01:01:01

R1 finds R2:

129.97.3.67 → 129.97.3.72

01:01:01:01:01:01 → ff:ff:ff:ff:ff:ff

R2 replies:

129.97.3.72 → 129.97.3.67

02:02:02:02:02:02 → 01:01:01:01:01:01

R1 sends B's message to R2:

129.97.2.2 → 129.97.3.5

01:01:01:01:01:01 → 02:02:02:02:02:02

R2 finds C:

129.97.3.72 → 129.97.3.5

02:02:02:02:02:02 → ff:ff:ff:ff:ff:ff

C replies:

129.97.3.5 → 129.97.3.72

cc:cc:cc:cc:cc:cc → 02:02:02:02:02:02

R2 sends B's message to C:

129.97.2.2 → 129.97.3.5

02:02:02:02:02:02 → cc:cc:cc:cc:cc:cc

7.4 Part d

D finds R2:

129.97.3.10 → 129.97.3.1

dd:dd:dd:dd:dd:dd → ff:ff:ff:ff:ff:ff

R2 replies:

129.97.3.1 → 129.97.3.10

02:02:02:02:02:02 → dd:dd:dd:dd:dd:dd

D sends message via R2:

129.97.3.10 → 67.195.160.76

dd:dd:dd:dd:dd:dd → 02:02:02:02:02:02

R2 finds R1:

129.97.3.72 → 129.97.3.67

02:02:02:02:02:02 → ff:ff:ff:ff:ff:ff

R1 replies:

129.97.3.67 → 129.97.3.72

01:01:01:01:01:01 → 02:02:02:02:02:02

R2 forwards message via R1:

129.97.3.10 → 67.195.160.76

02:02:02:02:02:02 → 01:01:01:01:01:01

R1 finds internet:

public ip → 74.11.22.14

01:01:01:01:01:01 → ff:ff:ff:ff:ff:ff

Internet replies:

74.11.22.14 → public ip

in:in:in:in:in:in → 01:01:01:01:01:01

R1 forwards message via internet:

129.97.3.10 → 67.195.160.76

01:01:01:01:01:01 → in:in:in:in:in:in