15-441:Networking – Solutions Homework 4

Spring 2006

Out: Friday 4/28/06 - Due: Friday 5/5/06, 5pm

Problem 1: TCP throughput

Chapter 6, question 16

Answer:

(a)
In slow start, the size of the window doubles every RTT. At the end of the ith RTT, the window size is $2^i KB$. It will take 10 RTTs before the send window has reached $2^{10} KB = 1MB$.

(b)
After 10 RTTs, 1023KB has been transferred, and the window size is now 1MB. 9217KB remains to be transferred. This requires 9 RTTs for the first 9MB, and 1 more for the last 1KB, for a total of 20.

(c)
It takes 2.0 seconds ($20RTs * 100msRTT$) to send the file. The effective throughput is (10MB / 2s) = 5MBps = 41.9Mbps. This is only 4.2% of the available link bandwidth.

Problem 2: TCP congestion control

Chapter 6, question 28

Answer:

Slow start is active up to about 0.5 sec on startup. At that time a packet is sent that is lost; this loss results in a coarse-grained timeout at T=1.9.

At that point slow start is again invoked, but this time TCP changes to the linear-increase phase of congestion avoidance before the congestion window gets large enough to trigger losses. The exact transition time is difficult to see in the diagram; it occurs sometime around T=2.4.

At T=5.3 another packet is sent that is lost. This time the loss is detected at T=5.5 by fast retransmit; this TCP feature is the one not present in Figure 6.11 of the text, as all lost packets there result in timeouts. Because the congestion window size then drops to 1, we can infer that fast recovery was not in effect; instead, slow start opens the congestion window to half its

previous value and then linear increase takes over. The transition between these two phases is shown more sharply here, at T=5.7.

Problem 3: Multimedia problem

Chapter 9, problem 33

Answer: Each receiver gets to use 5% of 1/1000 of 20 KByte/sec, or 1 byte/sec (strictly speaking 1.024 Bytes/second). This translates into one RTCP packet every 84 sec. At 10K recipients, the rate drops to one packet per 840 seconds, or 14 minutes.

Problem 4: Security problem

Part A: Consider slides 20 and 21 of lecture 24. Based on the belief rules summarized on slide 20, summarize which beliefs B has after receiving each of the packets sent to it in the protocol described on slide 21. Provide a brief argument for each belief B acquires. Your argument may refer to beliefs B had before the beginning of the protocol if you wish. There is no need to provide a justification for beliefs such as "S has not been taken over by an attacker."

Answer:

Initially, B believes

1. A & S know A's key, which is otherwise not known
2. B & S know B's key, which is otherwise not known
3. S follows the protocol outlined, e.g., S does not spuriously encrypt numbers and send them to B.

When B receives "I am A", B should not believe that claim. B sends a nonce to the party alleging to be A, and should immediately assume that everybody in the world learns this nonce value.

When B receives $\{n\}k_A$, this should not cause B to take on any new beliefs. B has no way to check whether or not the received bits contain the nonce or some other value, encrypted with A's key or some other key. B packages together A's identity with the bits received from the party claiming to be A, encrypts the package with its key, and sends the result to the authentication server. B does this based on the initial assumption that only the authentication server knows B's key and can decrypt the packet.

When B receives the next packet, B must *not* trust that it comes from the authentication server. However, if a decryption with B's key reveals the nonce B sent to the A claimant, B can come to an important set of beliefs.

Barring lucky guesses, if the packet decrypts to n, then it was encrypted by somebody who knew B's key, meaning S.

Furthermore, if S is following the protocol, S sent that packet to B as a result of using B's key to decrypt the package and then A's key to extract B's nonce.

That is, B believes the packet is from somebody who

1. knows B's key,
2. knows A's key, and
3. received a packet from somebody who knows A's key.

B believes S believes the A claimant knows A's key, so B believes the A claimant is A.

Part B: Alice claims that the protocol provides protection against replay attacks. Bob claims that it does or does not, depending on "whether the protocol is used carefully." Chris claims it provides no protection against replay attacks. Who is right? Explain.

Answer: Bob is right–the protocol provides replay protection if B never uses the same nonce twice (or does so extremely rarely). B could carefully increment the nonce value every time it is used, or could use nonces chosen by a *cryptographically random* number source (*not* the output of a "random number generator").

Problem 5: QoS problem

Three flows A, B, and C arrive at a router with a WFQ scheduling policy. Assume packets from each flow arrive often enough that the router always has at least one packet queued for each flow. Flow A has reserved $\frac{2}{3}$ of the throughput on the outgoing link. Flow B has reserved $\frac{1}{4}$ of the throughput on the outgoing link. Flow C has reserved $\frac{1}{12}$ of the throughput on the outgoing link. All packets are the same size.

The WFQ scheduler in this problem is roughly modeled after a "general processor sharing" model. At the start of each packet time slot, each flow receives credit proportional to its weight (not unlike a token bucket, except that there are multiple flows). Next, the flow with the most credit is chosen to send a packet in that slot. If two or more flows have the same amount of credit, the router prefers A, then B, then C. Of course, each time the router sends a packet on behalf of a flow, that flow's credit is reduced appropriately (again similar to a token bucket). Note that, unlike a token bucket, a flow's credit value may be negative.

Describe in full the order in which packets leave the router. Briefly explain your reasoning.

Answer: he table below show the credits at the beginning of each time step, what flow gets to send a packet, and the credits at the end of each time step.

Time step	Credit Before	Sending Flow	Credit After
1	A(2/3) B(1/4) C (1/12)	A	A(-1/3) B(1/4) C (1/12)
2	A(1/3) B(1/2) C (1/6)	B	A(1/3) B(-1/2) C (1/6)
3	A(1) B(-1/4) C (1/4)	A	A(0) B(-1/4) C (1/4)
4	A(2/3) B(0) C (1/3)	A	A(-1/3) B(0) C (1/3)
5	A(1/3) B(1/4) C (5/12)	C	A(1/3) B(1/4) C (-7/12)
6	A(1) B(1/2) C (-1/2)	A	A(0) B(1/2) C (-1/2)
7	A(2/3) B(3/4) C (-5/12)	B	A(2/3) B(-1/4) C (-5/12)
8	A(4/3) B(0) C (-1/3)	A	A(1/3) B(0) C (-1/3)
9	A(1) B(1/4) C (-1/4)	A	A(0) B(1/4) C (-1/4)
10	A(2/3) B(1/2) C (-1/6)	A	A(-1/3) B(1/2) C (-1/6)
11	A(1/3) B(3/4) C (-1/12)	B	A(1/3) B(-1/4) C (-1/12)
12	A(1) B(0) C (0)	A	A(0) B(0) C (0)

So the packets leave the system in this order

A, B, A, A, C, A, B, A, A, A, B, A. This sequence of 12 packets then repeats itself forever.